UNIVERSITÀ DEGLI STUDI DI MILANO

Ph.D in Computer Science – XXXI cycle
Department of Computer Science “Giovanni Degli Antoni”
Settore scientifico disciplinare: INF/01

Design and Synthesis of High Density Integrated Circuits

Ph.D Thesis of:
Luca Frontini

Tutor: Dr. Gabriella Trucco
Co-tutor: Prof. Valentina Ciriani

Ph.D school coordinator: Prof. Paolo Boldi

Academic year 2017–2018

Abstract

Gordon E. Moore, a co-founder of Fairchild Semiconductor, and later of Intel, predicted that after 1980 the complexity of an Integrated Circuit (IC) would be expected to double every two years. The prevision made by Moore held for decades, for this reason it is also called "Moore's law".

The trend in ICs is driven by a reduction of area and power consumption. Today scaled CMOS technologies are the main solution for digital processing. However, the interconnection scaling is not optimal. At every new technology node, the number of metal layers and their thickness increases, exploiting the vertical direction. The reduction of the minimum distance between interconnections and the growth in vertical dimension increase the parasitic capacitance and consequently the dynamic power consumption. Moreover, due to the non-optimal scaling of the interconnections, signal routing is becoming more and more challenging at every technology node advancement. Very scaled technologies make possible to reach a great transistor density. However, the design must comply to strict rules for metal interconnections.

The aim of this thesis is to find possible solutions to the disadvantages of scaled CMOS technologies. This goal is obtained in two different ways: using ad-hoc design techniques on today CMOS technologies and finding new approaches to logic synthesis of nanocrossbars, that are an emerging post-CMOS technology. The two approaches used corresponds to the two parts of this thesis.

The first part presents the design of an Associative Memory (AM) focusing the attention on develop design and logic synthesis techniques to reduce power consumption. The field of applicability of AMs is real-time pattern-recognition tasks. The possible uses range from scientific calculations to image processing for intelligent autonomous devices to image reconstruction for electro-medical apparatuses. In particular AMs are used in High-Energy Physics (HEP) experiments to detect particle tracks. HEP experiments generate a huge amount of data, but it is necessary to select and save only the most interesting tracks. Being the data compared in parallel, AMs are synchronous ICs that have a very peaked power consumption, and therefore it is necessary to minimize the power consumption. This AM is designed within the projects IMPART and HTT in 28 nm CMOS technology, using a fully-CMOS approach. The logic is based on the propagation of a "kill signal" that, if one of the bits in a word is not matching, inhibits the switching of the following cells. Thanks to this feature, the designed AM array consumes less than 0.7 fJ/bit. A prototype has been fabricated and it has proven to be functional.

The final chip will be installed in the data acquisition chain of ATLAS experiment on HL-LHC at CERN.

In the future nanocrossbars are expected to reduce device dimensions and interconnection complexity with respect to CMOS. Logic functions are obtained with switching lattices of four-terminal switches. The research activity on nanocrossbars is done within the project NANOxCOMP.

To improve synthesis are used some algorithmic approaches based on Boolean function decomposition and regularities, in particular P-circuits, EXOR-Projected Sums of Products (EP-SOP), Dimension-reducible (D-RED) functions and autosymmetric functions. The decomposed functions are implemented into lattices using internal and external decomposition methods. Experimental results show that this approaches reduce the complexity of the single synthesis problem and leads, in average, to a reduction of lattice area and synthesis time. Lattices are made of self-assembled structures and they have a non-negligible defectivity ratio. To cope with this limitation, are presented some techniques to reduce sensitivity to defects.

Contents

1 Introduction **1**
- 1.1 CMOS technology . . . 4
- 1.2 Switching lattices . . . 5
- 1.3 Thesis Organization and Overview . . . 6
  - 1.3.1 Part I: Scaled CMOS Technology . . . 6
  - 1.3.2 Part II: Switching Lattices . . . 7

I Scaled CMOS Technology **9**

2 Scaled fabrication processes **11**
- 2.1 Technologies and materials . . . 12
  - 2.1.1 High-κ dielectrics . . . 12
  - 2.1.2 Regular Patterns . . . 13
- 2.2 Scaling issues . . . 14
  - 2.2.1 Variability . . . 14
  - 2.2.2 Interconnection capacitances . . . 14
  - 2.2.3 Digital Switching Noise . . . 16

3 Associative Memory ICs **19**
- 3.1 AM chip road-map . . . 21
- 3.2 AM chip architecture and functionality . . . 23
  - Write mode . . . 23
  - Compare mode . . . 23
  - 3.2.1 Variable resolution . . . 24
  - 3.2.2 AM chip required specifications . . . 25
- 3.3 KOXORAM Associative Memory Cell for AM07 . . . 25
  - 3.3.1 Schematic Diagrams and Layout . . . 28
  - 3.3.2 KOXORAM cell working modes . . . 28
  - 3.3.3 Clockless logic . . . 28
  - 3.3.4 Simulations . . . 30

3.3.5 Measurements 30
3.4 Improvements for AM08 32
3.4.1 Hi performance technology 32
3.4.2 KOXORAM+ 32
3.4.3 SRAM metastability 33
Butterfly diagram 33
SRAM design 34
3.5 Quorum circuit 34
3.5.1 Simulation Results 39
3.5.2 Design Verification 41
3.6 Digitally Controlled Oscillator 41
3.6.1 Circuit Structure 43
3.6.2 Layout Design 46
3.6.3 Simulations 48

II Switching Lattices 51

4 Technology Description 53
4.1 Boolean function implementation 54
4.2 Synthesis methods 56
4.2.1 Altun-Riedel 56
4.2.2 Gange-Søndergaard-Stuckey 58

5 Decomposition Methods 59
5.1 P-circuits and EP-SOP forms 59
5.1.1 P-circuits 61
5.1.2 EXOR-Projected Sums of Products 63
5.2 D-reducible Boolean functions 67
5.3 Autosymmetric functions 69
5.4 Internal Composition 72
5.4.1 P-Circuits 73
5.4.2 EXOR-Projected-Sums Of Products 77
5.4.3 D-Reducible functions 77
5.4.4 Experimental Results 82
P-circuits and EP-SOP 82
D-Reducible functions 87
5.5 External Composition 89
5.5.1 Autosymmetric functions 91
5.5.2 P-Circuits 92

5.5.3 D-Reducible functions 93
5.6 Experimental results 94

6 Defect Tolerance **101**
6.1 Defect Injection Methodology 101
6.2 Metrics used for Sensitivity Analysis 102
6.2.1 Benchmarks and Simulations 102
6.3 Mitigation by Defect Avoidance 104

7 Conclusions **107**
7.1 Concluding remarks 107
7.2 Future developments 109
7.3 Publications 109

Appendices

A AM chip specifications **111**
A.1 Main requirements 111
Aggressive goals 112
Full Custom simulation Corners 112
Interdisciplinary Applications 112
A.1.1 Cores 113

Bibliography **116**

List of Terms **127**

Chapter 1

Introduction

From 1952 to 1956, Willard V. Quine [1, 2] and Edward J. McCluskey [3] developed a new algorithm to minimize Boolean functions, called Quine-McCluskey algorithm.

At that time, logic circuits were built using discrete components. Quine-McCluskey algorithm aims at minimizing the number of literals, and so the devices, needed to synthesize a Boolean function. The minimization of device number was a strong requirement to build digital circuits because it permits to reduce the cost related to the components and to mitigate the problem of heat dissipation.

In 1958, Jack S. Kilby showed the first working Integrated Circuit (IC) [4] to Texas Instruments managers, for the first time electronic components were integrated onto a single substrate. The technology used to build ICs was the first step to extend the use of transistors into mass-produced electronic circuits and microprocessors, indeed in a few years the number of component inside a single IC grew up to more than one hundred integrated devices.

Gordon E. Moore, a co-founder of Fairchild Semiconductor, and later of Intel, wrote in an article published in 1965 [5]: "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (Figure 1.1). Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least ten years".

Later, in 1975, Moore revised his forecast [6] predicting that after 1980 the rate of increase of complexity can be expected to double every two years instead of one, as Figure 1.2 shows. He wrote that it was due to a reduction of "circuit and device cleverness" that are particular production and design techniques, that permit to improve device density.

The prevision made by Moore held for decades, for this reason the plot in Figure 1.2b is also called "Moore's law".

The increasing density of ICs was obtained also using a new device: the Metal-

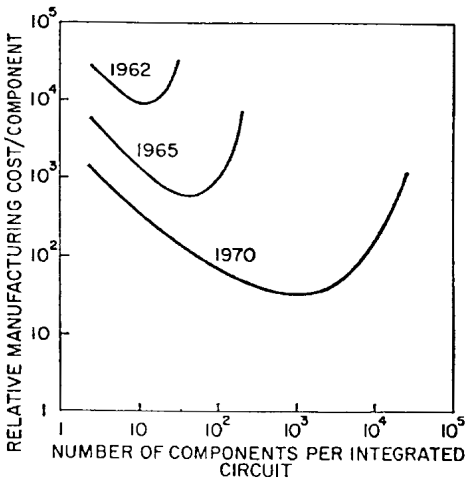
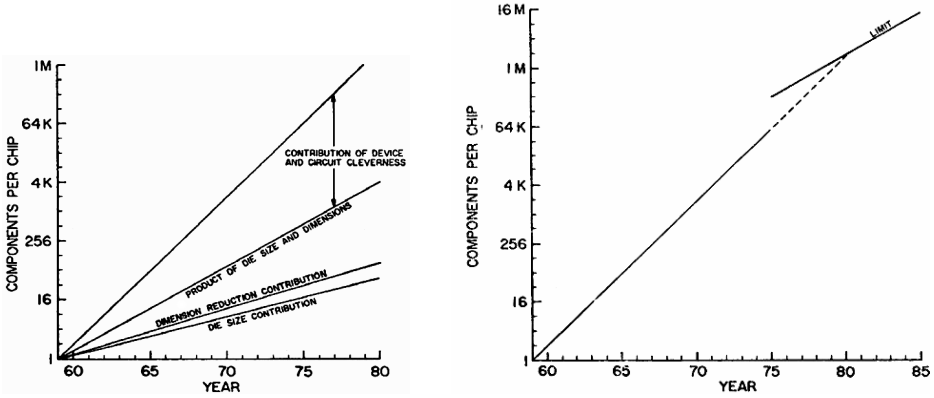


Figure 1.1: Plot of relative manufacturing cost per component, published by G. Moore in [5].



(a) (b)

Figure 1.2: (a): approximate component count for complex integrated circuits vs. year of introduction. (b): projection of the complexity curve reflecting the limit on increased density through invention. Published by G. Moore in [6].

Oxide-Semiconductor Field-Effect Transistor (MOSFET). The MOSFET was discovered by Julius E. Lilienfeld in 1925 and integrated into an IC by D. Kahng and Martin M. Atalla at Bell Labs in 1959. The MOSFET was described by Moore in 1965 as a promising device for future digital ICs [7] because a MOSFET can scale better than a Bipolar Junction Transistor (BJT) indeed MOSFET construction is planar and can be easier integrated into fabrication processes.

With the implementation of Complementary Metal-Oxide Semiconductor (CMOS) technology, that uses p-MOS and n-MOS transistors instead of resistors as pull-up and pull-down, the BJT was almost completely substituted by MOS in digital ICs. CMOS logic has less static power than BJT logic and pull-up and pull-down resistors, are not necessary. This aspects led to the diffusion of CMOS because they permit to fabricate low power consuming ICs that also present a higher device density.

MOS transistor size has been scaled for decades in an almost regular way, according to the Moore's law [8, 9]. The trend in ICs is driven by a reduction of area and power consumption. Today scaled CMOS technologies are the mainstream solution for digital processing.

However, the interconnection scaling is not optimal [11, 12]. At every new technology node, the number of metal layers and their thickness increases, exploiting the vertical direction, as Figure 1.3 shows. The reduction of the minimum distance between interconnections and the growth in vertical dimension increase the parasitic capacitance and consequently the dynamic power consumption. Moreover, due to the non-optimal scaling of the interconnections, signal routing is becoming more and more challenging at every technology node advancement, and it is limiting IC performances.

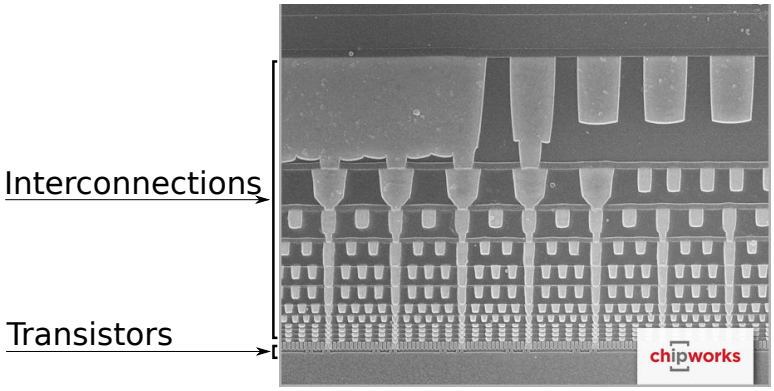


Figure 1.3: Cross section of Intel $14\,\mathrm{nm}$ CMOS technology [10].

1.1 CMOS technology

Very scaled technologies make possible to reach a great transistor density. However, the design must comply to strict rules for metal interconnections, and the reduction of interconnections is more important than reduction of the number of components, to optimize area and power consumption.

Today, to implement complex production-grade logic ICs it is necessary to deal with the disadvantages of scaled CMOS technologies and mitigate them with ad-hoc design techniques.

One of the applications where scaled CMOS technologies are used are Content Addressable Memories (CAMs). CAMs write data at an address and then, during comparison tasks, find addresses that match data. CAMs can be used for sparse database search, cache or routing table.

Associative Memories (AMs) are a particular type of CAMs [13] because AMs can handle also segmented data. The field of applicability of AMs is real-time pattern-recognition tasks. The possible uses range from scientific calculations to image processing for intelligent autonomous devices to image reconstruction for electro-medical apparatuses.

In particular, AMs are used in High-Energy Physics (HEP) experiments to detect particle tracks; the track of a particle is the path that a charged particle follows as it transverses the various layers of the detector. HEP experiments generate a huge amount of data, but not all the data are stored offline. The decision to store the data, or not, is taken online by the trigger system of the experiment. For best results in identifying interesting data and rejecting non-interesting ones, track identification and reconstruction is often needed early in the trigger chain, when the data rate is still high. In order to identify the tracks, AMs are used in the acquisition chain to find a correlation between the input data coming from the detector pixels and a set of pre-stored data [14, 15], for this reason they have to respect strict requirements of timing and power consumption. Being the data compared in parallel, AMs are synchronous ICs that have a very peaked power consumption, with peaks aligned with the rising edge of the clock.

There are four critical factors, described above, for AM and more in general for CMOS ICs:

1. signal routing and interconnections are critical due to scaling;
2. reduction of interconnections is more important than reduction of components;
3. AM power consumption is peaked and need to be minimized;
4. AM has to satisfy the specifications given by the trigger system.

This thesis describes the research activities within the projects IMPART[1] [16] and HTT[2] [17]. The final chip will be implemented in the data acquisition chain of ATLAS ("A Toroidal LHC ApparatuS") [18] experiment in the High Luminosity Large Hadron Collider (HL-LHC) at Conseil Européen pour la Recherche Nucléaire (CERN).

1.2 Switching lattices

In future, the Moore's law will stop to be valid [11]. The only way to improve speed, area, and power consumption will be to go over CMOS, by adopting new technologies. One of the emerging post-CMOS technologies are nanocrossbars [19].

Nanocrossbars arrays are fabricated with relatively cheap bottom-up nano-fabrication techniques rather than using purely lithography based conventional production [20, 21, 22]. Due to the novel manufacturing techniques, fabric yields to be in regular and dense form [23]. Because of their structure and technology, they are area and power efficient but they have a non-negligible defectivity ratio. Logic functions are obtained with arrays of crossbar-type switches. This structure is expected to reduce device dimensions and interconnection complexity with respect to CMOS.

Logic synthesis on nanocrossbars is one of the most important steps that are necessary to utilize this technology instead of CMOS [24]. Nanocrossbars requires different tools with respect to CMOS, due to the different type of switches. In literature there are described two possible algorithms to synthesize Switching lattices. The first one, developed by Altun and Riedel [25] produces lattices with a size that grows linearly with the number of products of the starting Boolean function, it runs in time that grows polynomially in the number of ISOP. The second algorithm, developed by Gange, Søndergaard and Stuckey [26], uses a SAT-solver, it provides minimum area lattices, but it uses a considerable amount of time and resources.

As described above there are some interesting nanocrossbars peculiarities:

1. they are an emerging post-CMOS technology;
2. they are area and power efficient;
3. the optimum synthesis requires a considerable amount of time and resources;

[1] Innovative Multi-chip system for multi-purpose PAttern Recognition Tasks (IMPART) is a project that consists in developing a cutting edge pattern recognition device for fast image analysis and future trigger processors for HEP experiments.

[2] Hardware Tracking for the Trigger (HTT) is a Hardware-based Tracking for the ATLAS Trigger, based on custom-designed AM ASICs for pattern recognition and FPGAs for track reconstruction and fitting.

4. they have a non-negligible defectivity ratio;

This thesis describes the work done within the project NANOxCOMP[3], that aims to develop a complete synthesis methodology for nanoscale switching crossbars that leads to the design and construction of an emerging computer [27].

1.3 Thesis Organization and Overview

Moore's law is holding for decades, but its end is approaching. The major problems of today scaled CMOS technologies are interconnection capacitance and routing complexity.

Therefore the aim of this thesis is to to find possible solutions to the disadvantages of scaled CMOS technologies.

This results are obtained using two different approaches: I) designing a commercial-grade AM IC finding solutions to mitigate power consumption and routing problems; II) developing nanocrossbars synthesis tools to improve the integration process. In this way, it is possible mitigate current CMOS issues and, in the mean time, find synthesis techniques for nanocrossbars to provide a proper synthesis tool set when the fabrication processes will be suitable for mass-adoption.

1.3.1 Part I: Scaled CMOS Technology

This part is focused on scaled CMOS technologies, that today are the main solution for digital processing. Then are analyzed their peculiarities to find design and synthesis approaches that can be useful to design digital circuits, in particular AMs.

Chapter 2 analyzes scaled CMOS fabrication processes. The description helps to identify the characteristics that can be exploited for the synthesis and the design, such as the great device density, and the issues that is possible to mitigate, for instance the non optimal interconnection scaling and the particular layout rules related to fabrication processes.

Chapter 3 is focused on AM ICs. First are described some possible applications of AMs. This description is useful to understand some design peculiarities and the required specifications of timing and power consumption. Then are described the AM chip architecture and the function of each logic block. The specifications of the AM chip are published in [28]. Then it describes the design of an AM cell that reduces power consumption with respect to previous solutions, using a logic

[3]NANOxCOMP has received funding from the European Union's H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 691178.

synthesis approach aiming at reducing interconnections instead of the number of transistors. The cell is designed in 28 nm CMOS technology, using a fully-CMOS approach. The designed AM cell permits, during comparison, to send a "kill" signal to the cascaded cell to inhibit further switching if one of the input bits is not matching the pre-stored data. Thanks to this feature, the CAM array requires less than 0.7 fJ/bit per comparison and reaches a comparison frequency of $184\,\mathrm{MHz}$. Than are describes the layout design, the simulations that have been done to test the SRAM metastability and the design techniques that used to avoid it.

Than are described the two blocks that will be included into the next test chip to further improve the memory density and reduce the power consumption of the AMs: the quorum circuit an the Digitally Controlled Oscillator(DCO). The quorum circuit orders the outputs of the memory arrays and has a logic based on a divide and conquer sorting algorithm. It will substitute the former standard cell circuit occupying less silicon area. The DCO uses logic gates as delay elements to mitigate the mismatch of scaled CMOS technology, it is used to spread the AM clock to reduce the current peak due to AMs. The AM cells and the test of AM07 are published in: [29, 30, 31]. The population count circuit is published in [32, 33]. The DCO is published in [34]

1.3.2 Part II: Switching Lattices

Switching Lattices are an emerging technology that can solve some issues of scaled CMOS technology. The synthesis of minimal Switching lattices is time and resource intensive and can fail to synthesize the lattice in a reasonable amount of time. For this reason, this part shows some methods aimed at reducing computing time and lattice area, and describes some techniques for mitigating lattice faults.

In **Chapter 4** describes the synthesis tools that today are used for switching lattices. The description permits to show the peculiarities of each method.

In **Chapter 5** studies an algorithmic approach to switching lattices based on Boolean functions decomposition methods and regularities: P-circuits, EXOR-Projected Sums of Products (EP-SOP), Dimension-reducible (D-RED) functions and autosymmetric functions. It shows two different approaches to implement the decomposed functions into lattices: internal and external decomposition. Internal decomposition implements all the sub-lattices into one single lattice. External decomposition uses a two level approach and requires additional inverters and routing.

A large set of experimental results is presented showing in average an area and synthesis time gain. The external decomposition method can obtain lattices that

have a total area smaller than the optimum single lattice. Internal decomposition results on P-circuit are published in [35] and D-red are published in [36]. The article that compare the decomposition methods is [37]. External decomposition results on autosymmetric functions are published in [38]. The comparison between external decomposition is published in [39].

In **Chapter 6** enunciates a metric for sensitivity analysis of switching lattices that takes in account SA0 and SA1 faults. Then, it shows several methodologies that exploit the regularities of synthesis algorithms aimed at mitigating the lattice defects. Part of this chapter is published in [40].

Part I

Scaled CMOS Technology

Chapter 2

Scaled fabrication processes

Microelectronic device industries invest a lot of resources to find novel technologies to increase device density and circuit speed minimizing power consumption. This is done optimizing the CMOS production process changing the fabrication procedures and the materials used to build the devices [9].

The classic scaling (shown in Figure 2.1), described in [12] by R. Dennard in 1975, cannot be strictly applied to production nodes under $130\,\mathrm{nm}$ [41], some technology improvements are needed. Often below this threshold there were done previsions of scaling problems due to technology limitations, materials or productive processes, but then technology improvements and fabrication process modifications succeeded into overcoming these limits.

The increase of device density and the adoption of new materials and fabrication procedures requires some modification on design and synthesis of digital circuits, in particular it is necessary to take into account the power consumption.

Device or circuit parameter	Symbol	Scaling factor
Device dimensions	t_{ox}, L, W	$1/k$
Doping concentration	N_a	k
Voltage	V	$1/k$
Current	I	$1/k$
Capacitance	$\varepsilon A/t_{ox}$	$1/k$
Delay time	VC/I	$1/k$
Power dissipation	VI	$1/k^2$
Power density	VI/A	1

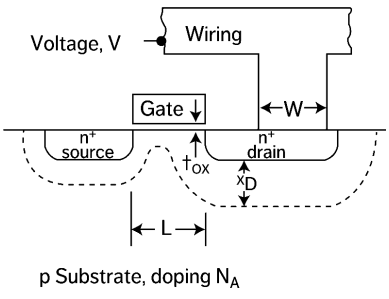
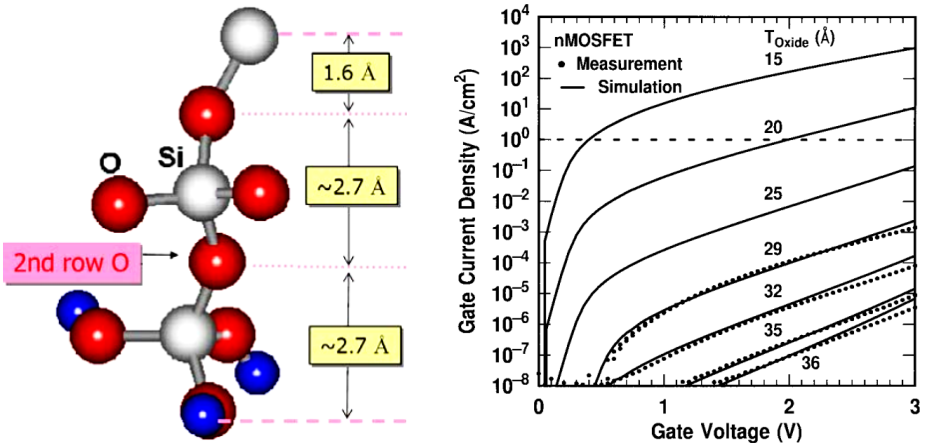


Figure 2.1: Traditional MOSFET scaling as described by Robert Dennard, published by Bohr and Young in [9]



(a) (b)

Figure 2.2: (a) Graph of the oxide leakage current versus gate voltage per different oxide thickness. The oxide leakage current increases exponentially as the oxide thickness reduced linearly [43]. (b) Bonding structure of SiO_2 indicating the minimum thickness of the bulk oxide is about 7 Å [42].

2.1 Technologies and materials

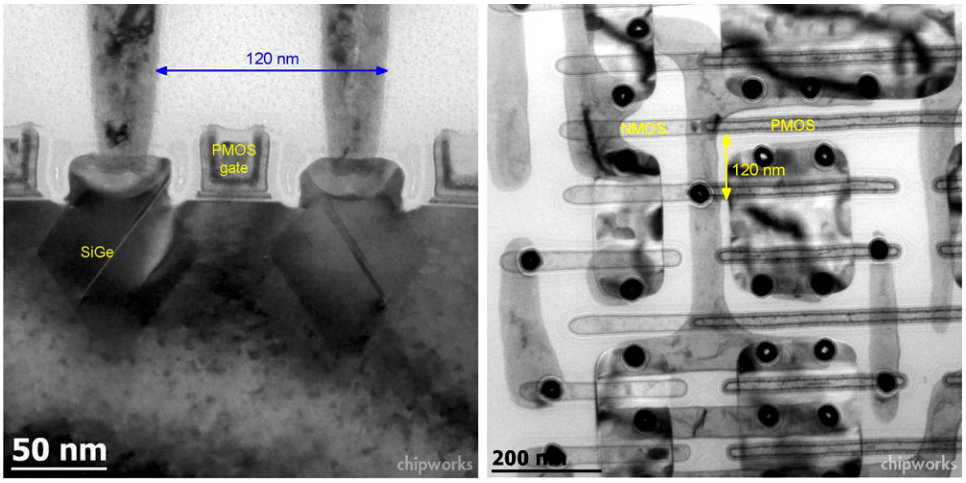
In this section we describe the principal technologies used by scaled CMOS technologies.

2.1.1 High-κ dielectrics

For decades silicon oxide (SiO_2) was used to build the gate oxide of MOSFETs. The downsizing of MOS follows the rules shown in Figure 2.1: all the voltages and dimensions are reduced by a factor k. To have an electric field of the scaled transistor equal to the large device, the charge density, and so the oxide capacity (C_{ox}), have to be up-scaled by k.

The approach used approximately until $64\,\mathrm{nm}$ process was to the reduce oxide thickness. In $28\,\mathrm{nm}$ and smaller technology nodes the needed thickness would be near the minimum thickness of SiO_2, 7 Å [42], that corresponds to two layers of SiO_2 atoms, as Figure 2.2a shows. Moreover the oxide leakage current increases exponentially as the oxide thickness reduces linearly, as Figure 2.2b shows [43]. For this reason MOSFET with thin oxides are not suitable for low power circuits.

Either the physical and the technological constraints can be circumvented by replacing the ultra-thin SiO_2 with physically thicker high-κ materials. But the most promising high-κ material, the hafnium oxide HfO_2, presents problems due to the interface between silicon and insulator, to provide better results it is used the hafnium-silicon oxinitride (HfSiON). HfSiON is obtained by atomic layer deposition to improve the oxide thickness uniformity. Figure 2.3a shows a PMOS



(a) (b)

Figure 2.3: (a) Altera 5SGXEA7K2F40C2 Stratix V 28 nm HP PMOS, TEM photograph [45]. (b) Xilinx XC7K325T Kintex-7 TSMC 28 nm HPL, Plan View TEM photography [45], different types of metals are used to fabricate PMOS and NMOS gates

in 28 nm TSMC technology, that uses high-κ dielectrics. HfSiON is not compatible with polysilicon gate, for this reason it is used a metal gate electrode, made by titanium nitride (TiN) for P-MOS and a titanium aluminum nitride (TiAlN) for N-MOS gate [44]. In Figure 2.3b it is possible to notice the two different types of material used to fabricate MOS gates.

2.1.2 Regular Patterns

Gates are fabricated using a gate-last approach, called also *damascene*. This fabrication procedure guarantee a precise gate alignment and make possible to use different types of metals for PMOS and NMOS. Gate-last approach uses sacrificial polysilicon gate structures that are substituted with the MOS metal gate.

The fabrication process is shown in Figure 2.4, from top-left to bottom-right:

- Deposition of Inter Layer Dielectric (ILD) over the sacrificial gate structures.
- Chemical Mechanical Polishing (CMP) to planarize the wafer.
- The sacrificial polysilicon is removed using a chemical attack.
- It is deposed a TiN gate for PMOS and a TiAlN gate for NMOS.

CMP is a critical polarization procedure because it is required to have a minimum material density to provide a good planarization without damaging the structures below the planarization level. For this reason it is necessary to use dummy

gate structures to have a good planarization. Dummy gates, as Figure 2.5 shows, are gate structures that does not intersect active area.

Transistor gates are build using phase shift masks [46, 47, 48]. In this way it is possible to create structures that are smaller than the wavelength of the light source used for the photolithography. Dummy gates are useful also for phase shift lithography because this techniques exploit light interference phenomena to create the gates and the most external gates of the interference figure may present some defects.

2.2 Scaling issues

Very scaled transistors permit to have great device density, but the process shrinking can lead to variability problems. Moreover interconnections do not scale as well as transistors increasing routing complexity and interconnection capacitances.

2.2.1 Variability

Scaled technologies presents a non-negligible process variability. The two main sources of variability are local mismatch, that involves devices of the same chip, and global variation, that involves devices on different silicon wafers. For example in scaled technologies the local mismatch can reach 40% in resistors and the variation among different wafers can cause a great difference in circuit working speed and power consumption.

It is possible to partially simulate process variability using Montecarlo simulations on the device parameters. Silicon foundries gives the models of the typical-case and worst-case corner that permit to simulate the parameters variability due to global mismatch.

2.2.2 Interconnection capacitances

Interconnections scaling is not optimal as for transistors. To provide a good connectivity the number of metal layers increases at every technology node exploiting more and more the vertical dimension, as Figure 1.3 shows. Furthermore at every technology node, to increase the metal routing density, the metals become thicker to maintain the needed cross-section area and prevent electromigration. For this reasons, as Figure 2.6 shows, the distance between the metals of the same layer, s, is smaller than the distance between metals of different layers, d. Furthermore the thickness of the metals, t, is almost the double of its minimum width w.

The dielectric used to fill the spaces between the metals is the same, so the parasitic capacitance between metals of the same layer can be more than double

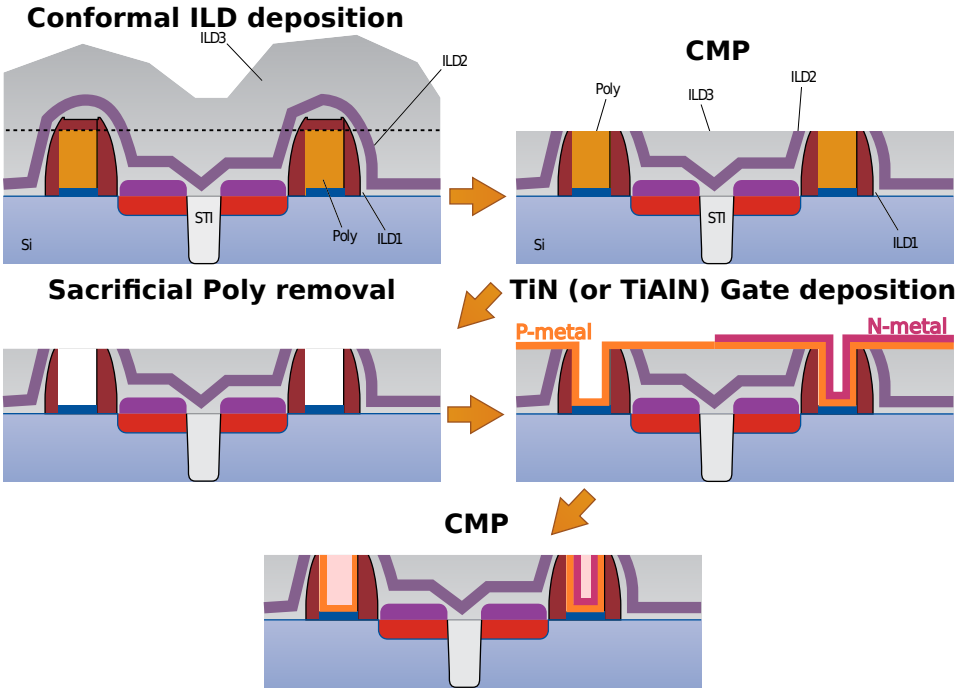


Figure 2.4: Gate-last fabrication process.

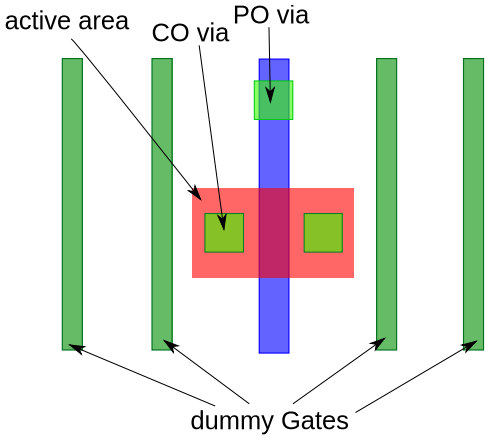


Figure 2.5: Drawing of a 28 nm transistor, showing the dummy gates.

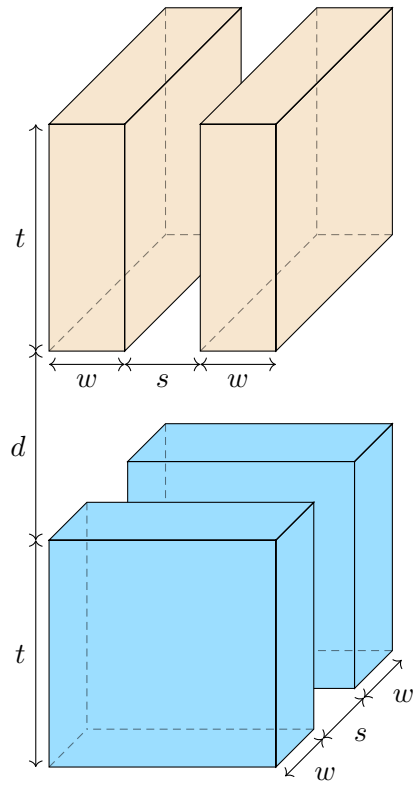


Figure 2.6: In scaled technologies d is more than two times s, and $t \approx 2w$. For this reason the parasitic capacitances due to same metal layer are bigger than capacitance due to superposition of different metal layers.

than the capacitance between different metal layers.

2.2.3 Digital Switching Noise

CMOS scaling leads to an exponential growth in device number, complexity and speed. This can cause unwanted interactions. Indeed in mixed signal System-on-Chip the switching of the digital circuits can influence the analog parts of the chip. The noise generated by digital switching activity propagates through parasitic elements due to interconnections and substrate affecting the system performance [49, 50, 51].

AM chips are massively parallel digital circuits. At each clock rising edge the comparison between stored and input data causes high current spikes. The current consumption produced an IR drop on the supply voltage of the memory; moreover, parasitic inductances and capacitances on the printed circuit board can cause oscillations that sometimes affected the functionality of the chip. This problem is usually

mitigated adding decoupling capacitors to the chip package power pins [52]. For the AM09 chip we propose also a complementary approach based on spreading the current peak providing eight clocks with slightly different phases using a Phase Locked Loop (PLL) based on a Digitally Controlled Oscillator (DCO), as explained in Section 3.6.

Chapter 3

Associative Memory ICs

By starting from late 80s [13], Associative Memories (AMs) was designed for High Energy Physics (HEP) applications. A common problem in HEP applications (in particular at ATLAS experiment) is the identification of particle tracks [17].

ATLAS experiment at Large Hadron Collider (LHC) in Phase-II, will produce up to $700\,\mathrm{GB/s}$ raw tracking data. Since a limited amount of events can be transferred to a storage system of the PC farms (for subsequent off-line processing), it is necessary to use an hardware trigger system to filter the relevant data for the event physics analysis. In addition, the luminosity of the collider will strongly increase in the next future. While in the past the trigger system do not take as input the track reconstruction, in view of the Phase-II upgrade a new system trigger system which performs the track reconstruction is needed. In this way, the system efficiency will reach about 90%.

The capability of performing real-time identification of tracks at LHC leads to a very high background rejection. For example, many overlapping proton-proton collisions can be separated using tracks to reconstruct the primary vertexes. Due to the complex structure of events, the trigger system must base its decision on a few tracks among several tracks.

Track reconstruction is performed by means of complex processors designed to achieves challenging constraints. For the Phase-I of LHC (up to 2020) the processor that the ATLAS community uses for this job is called Fast TracKer (FTK) [53]. FTK is composed by 16384 AMchip (version 6) and can perform 31 Ecomp/bit/s. For the next Phase-II of LHC (up to 2026) the process will by called Hardware for the Track Trigger (HTT). HTT will be composed by 13824 AMchip (version 9) and will can compare bit-wise 0.2 Zcomp/bit/s. The systems elaborate the information in two steps:

1. **pattern recognition:** the hit data coming from eight detector layers (over 12 detector layers) is clustered in low-resolution centroids that can be used for a

bit-wise comparison thanks to the AMs;

2. **track refining:** the found roads (output of the AMs) merged with the data coming from the whole 12 detector layers are used to refine the trajectories of the particles by mean of the $\tilde{\chi}^2$ calculation.

The AM chip performs the first task exploiting parallelism to the maximum level.

Theoretically, CPUs could provide the expected results (in terms of efficiency and resolution) but they require very large computing power and costs to keep up with the event rate of $250\,\mathrm{MHz}$ and the increase of luminosity (up to $2 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$) of Phase II upgrade [54], which will result in more than 400 overlapping proton-proton collisions in the same time-bin of the detector clock which identifies an event. Due to the event complexity increase and to the huge amount of data, CPU's tracking capability (with a reasonable number of CPUs per system) will not be sufficient to perform these tasks in future very-high luminosity runs. Trivially, if we consider to use one flop operation to run four word bit-wise comparison, even the most powerful supercomputer existing in the world (called IBM Summit) should not be able to run this analysis. IBM Summit capability is 122.3 Pflops/s. The HTT capability is 0.2 Zcomp/bit/s (2.7 Ecomp/word/s – 22.3x factor). However, the HTT is a specific supercomputer able to run just only bit-wise comparison.

Currently, a dedicate system based on AM is the unique solution that can match all the power, cost, and performance constraints [55].

The AM is a Application specific integrated circuit (ASIC) for pattern recognition based on a smart (CAM) architecture because the chip can store partial matches as they are found and use these partial matches to find correlations among data received at different times (each input bus is independent). In an AM each pattern is stored in a single memory location, like in the commercial CAMs, but the total number of available bits can be organized in N independent words of M bits each. Each word refers to a particular item to be identified in a flux of data that is distributed to one or to a group of multiple words that occupy a particular position in the pattern.

AM has be successfully used for other applications such as image processing [56, 57] and genome sequencing [58].

During my P.h.D. I had the opportunity to work on the design of AM. For this task, I had the responsibility to coordinate the group of Full Custom (FC) blocks. In particular, I am the main designer of CAM cells, and some other important full custom blocks (i.e., quorum). With my contribution the future AMchip (AM08 and AM09) can met the power budget.

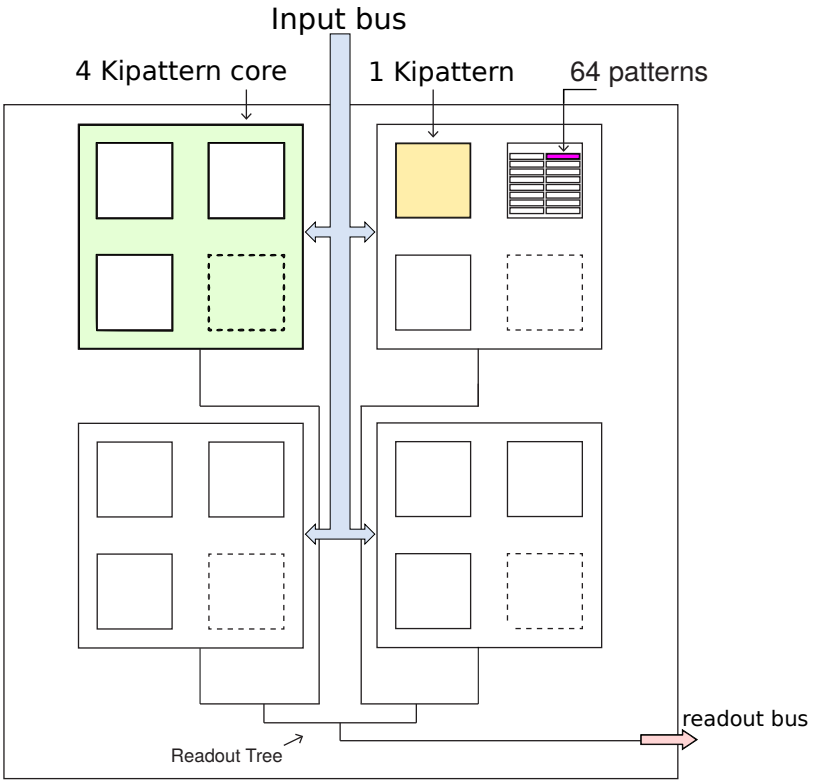


Figure 3.1: Simplified block diagram of the general architecture of the AM chips.

3.1 AM chip road-map

The silicon area for the AM06 [59] and the AM09 (the chips that we designed for a large volume production) is $154\,\mathrm{mm}^2$. Such large chips costs few millions of dollars, for this reason in the road-map there are two smaller chips to test the circuits and the features of the bigger chip. The large chip is designed mainly for HEP applications, but the smaller test chip provides features that can be used for multidisciplinary applications.

After the successfully project (AM06) for the Phase-1 at LHC, the prototype chips (AM07 and AM08) have an area of $10\,\mathrm{mm}^2$ and they contain 16×1024 patterns. The production chips (AM09) have an area of $154\,\mathrm{mm}^2$ and it contain 384×1024 patterns.

Smaller chips can be produced using a Multi Project Wafer (MPW) service to reduce costs, while big-area chips have to be produced using a Multi Layer Mask (MLM) service still to reduce costs. Therefore MLM is recommended for projects that require low or medium production volume. In our case we have to produce about 20 kchips.

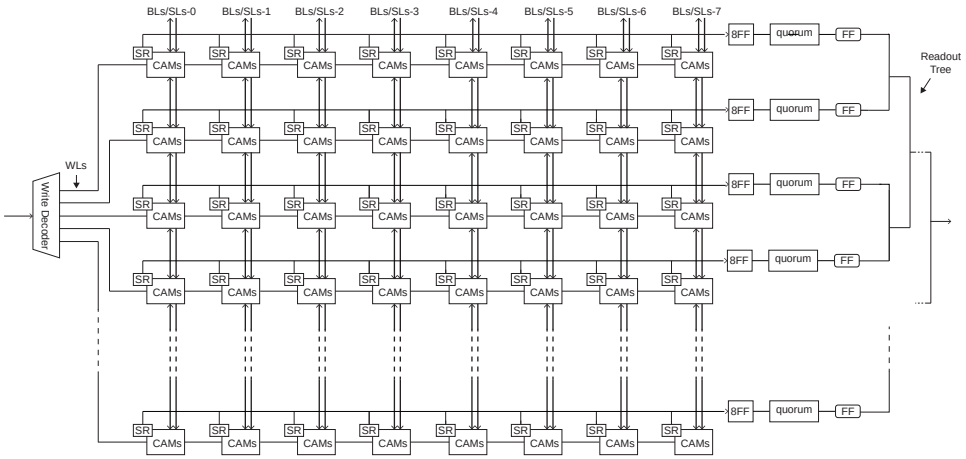


Figure 3.2: 64 pattern base memory block.

That is follow now is the time schedule and the aims of each prototype or chip:

- **AM07 prototype**: submitted in December 2016. It contains two different types of AM cells: KOXORAM and DOXORAM (an improved version of XORAM [60]). The goal of this test-chip was to test the memory cells and find which one consumes less power. The measurement confirmed the simulations and KOXORAM is proven to be the less consuming cell among the two. Designed in High Power Low-leakage (HPL) 28 nm technology.

- **AM08 prototype**: will be submitted in May 2019. It will be used to characterize, the new technology, the FC blocks (i.e, quorum, KOXORAM+ cell), the I/O circuits and protocols. This chip must have the same functionality of the AM09 ASIC, but smaller memory size; It will be designed in High Power and Computing (HPC) 28 nm technology.

- **AM09pre pre-production**: full area ASIC to be fabricated with a full-mask set or MLM pilot run. Production corner wafers will be created; It will be designed in HPC 28 nm technology.

- **AM09 production**: full area ASIC with refinements for the mass production. It will be designed in HPC 28 nm technology.

The AM09-pre and the AM09 chips will be developed on the AM08 extending the memory area, therefore the specifications of AM08 versions must be compatible with the specifications of AM09.

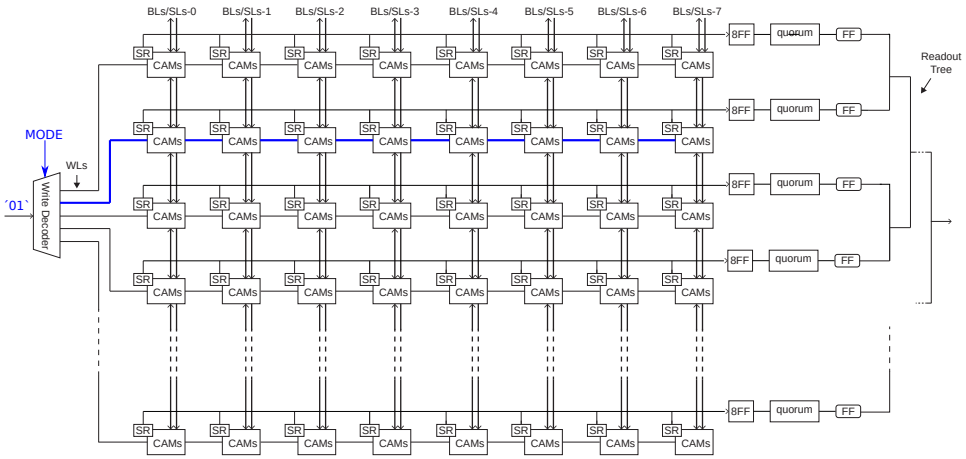


Figure 3.3: Write mode operation in AM08

3.2 AM chip architecture and functionality

The future AM will be designed with a core approach. AM08 will be composed by 4 elementary core. AM09 will be composed by 96 elementary core. Each core will contains 4×1024 patterns. Figure 3.1 shows the simplified block diagram of the general architecture of the AM08 chips.

More details can be found in [28].

The memory inside AM chips is composed by repeating multiple times a 64 patterns elementary block.

Figure 3.2 shows the functional diagram of 64-rows base block. The memory matrix consist of 64 patterns. Each line contains a *pattern*, that is composed by eight CAMs blocks. Each CAM block consists of a 18 bit *word* (each bit correspond to a CAM cell), and a Set Reset (SR) latch.

Write mode

The contents of the pre-defined pattern bank are propagated through the BLs. The address corresponding to a row of the CAM array are fed thanks to the write decoder that generate a one-hot signal for the WL bus (Figure 3.3). When the selected Write Line (WL) corresponding to the designated address is turned on, the pattern is written in the CAM segments at the selected address.

Compare mode

The information of hits coming from the detector is compared in parallel with the 18 bit word stored in each CAM block. The information is compared in parallel bus

by bus and row by row. If a 18-bit word find a bit-wise correspondence a SR latch is set to high logic value.

For each row, the word of matches (the eight correpondences found) is registered in eight Delay Flip-Flops (D-FFs) or High-Enable Latches located at the input of the quorum circuit using a dedicated pulse (READ_EV). Normally, this pulse is asserted when all the hits information has been fed into the AM. The pattern-match signal is calculated inside the quorum logic considering the number of matching words and other control signals such as the required matching threshold.

The pattern match outputs are read out with the readout clock and fed to the Road Out Tree (ROT) based on the Fischer's Tree [61]). The ROT reads out addresses of matched patterns as well as encoded 8-bit information of matched word (so called HITMAP) according to a priority embedded in the internal combinational logic. Several hierarchical level of ROT with different input bits are used in the AM to collect all the results.

Once an address/row is read out, the ROT generate a feed-back signal able to assert a DFF located at the output of the quorum. If the output of this DFF is '1' the match output will be disabled. With this strategy, the ROT can proceed to the next elaboration step.

3.2.1 Variable resolution

The AM can benefit of this nice feature called variable resolution: A large AM cell bank stores all patterns of interest, for a given input resolution. The AM extracts pattern addresses, when a sufficiently high number of words have matched the incoming data. For most practical problems the set of all patterns with full resolution is extremely large. The AM approach consists in performing pattern matching at reduced resolution first, with a resolution adequate to simplify and reduce the amount of data, and then refining the match inside a Field Programmable Gate Array (FPGA).

A don't care (DC) bit is used to increase the pattern recognition efficiency at different resolutions. It is possible to use patterns of variable shape. As a result of variable resolution, the number of fakes and the bank size decrease, and the efficiency remains high. Hence, for a given efficiency, the number of patterns required will decrease and a smaller pattern bank leads to a lower power consumption.

Figure 3.4 shows a qualitative example of the detection of four different particle trajectories. For a given efficiency, the fixed resolution approach requires to store three patterns, while with 1 or 2 DC bit it is necessary to store a single pattern. In addition, for the 2 DC case it is possible to further improve the tracking efficiency. Simulations demonstrated that an AM system with one single DC can be as effective as a three times larger AM system with fixed resolution [62].

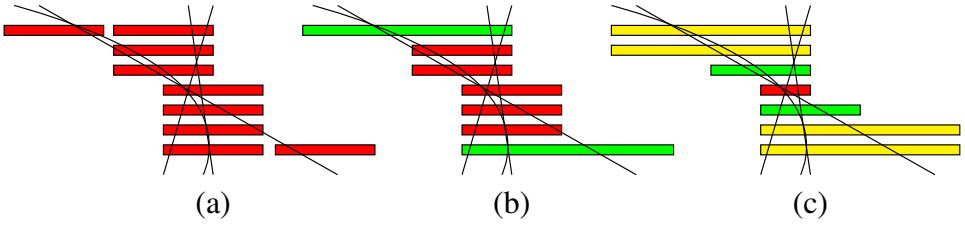


Figure 3.4: Diagram illustrating the multi-bit implementation of variable resolution words. Red rectangles are words without DC, green rectangles are words with one DC, yellow rectangles are words with two DC. (a) shows fixed resolution words. (b) shows words with 1 DC-bit. (c) shows words with 2-bit variable resolution, the pixels in this diagram have different scale with respect to the others.

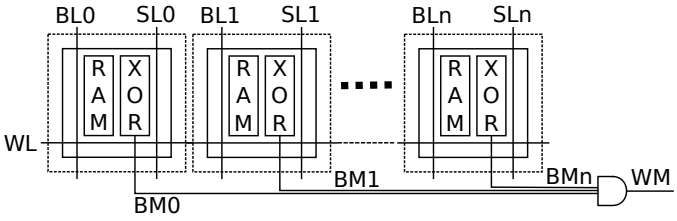


Figure 3.5: Structure of an associative memory

3.2.2 AM chip required specifications

The AM chip CAM related specifications and required features are listed in Appendix A, more detailed specifications can be found in [28].

3.3 KOXORAM Associative Memory Cell for AM07

The Figure 3.5 shows the conventional structure of a conventional CAM block. The elementary cell has the capability to store one bit of information (in "write" mode, when the WL signal is high, each cell stores the bit fed on the BLs), and to compare the memory content with the input data (in "compare" mode the RAM contents is compared with the bit at the SL input), providing at the output one bit which indicates whether the input bit is equal to the stored bit (the "bit match" signal BM). When all the bits in a stored word match the input data, then a "word match" signal (WM) is produced at the output.

Since the bit-wise comparison operation can be performed with an XOR logic gate, we have invented in the past a cell called "Xo(R)aM", which is made of a 6T SRAM cell and a pass-transistor XOR gate [60].

The new chip for the future upgrade of the ATLAS experiment will take advantage of the scaled features of the 28 nm CMOS technology, to integrate more

devices in the same silicon area [31]. The next generation associative memory chip is expected to contain about 1.5 billion transistors. However, the increased interconnection density in scaled technologies leads to higher parasitic capacitances, which may become the major source of dynamic power dissipation. To overcome this limitation, in the new design we adopted two complementary approaches.

1. The new associative memory cell must have a reduced switching activity, i.e., it may be disabled when its output is not relevant for the word matching.

2. The input signals must be fed through lines with low parasitic capacitances.

The two design aspects will be discussed in the following subsections.

A first approach to limit the power consumption is based on the reduction of switching activity of the internal nodes. Since the "word match" signal is active only when all the bits in a word are matching, it is obvious that a single non-matching bit will produce a non-matching output; in this case, all the cells operating on subsequent bits can be disabled to save power.

Cells can be disabled by propagating a "kill" signal. When a cell receives a positive "kill" signal from the previous one, it does not perform any comparison and it just propagates the "kill" signal to the next cell. If the input "kill" signal is not active, then the previous bits matched, and the cell compares the input data with the stored bit; if they do not match, the cell generates a positive "kill" signal at the output. Thus, the word matching occurs when the last cell is not providing a positive "kill" signal at its output.

This cell is called KOXORAM (Kill-Out XORAM), because it either performs an XOR logic operation or propagated the "kill" signal at the output, depending of the results of previous cells.

Since in fully CMOS technology the logic cells are inverting, we have designed two cells: one with positive "kill" at the input (NKill_In) and negative "kill" at the output (Kill_Out), and the other with negative "kill" at the input (Kill_In) and positive "kill" at the output (NKill_Out). The two versions of the cells can be cascaded, as shown in Figure 3.6. To limit the delay introduced by the propagation of the "kill" signal, in our design we have split the 18 bit word into two parts, each containing 9 bits; if the contents of both parts are matching the input data, the two

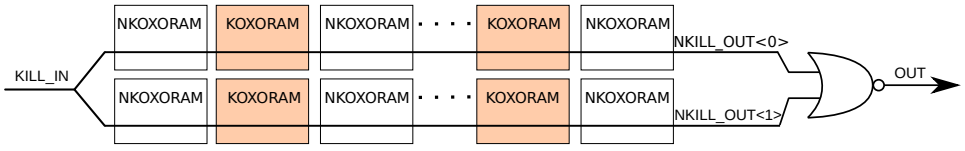


Figure 3.6: Connection of KOXORAM cells operating on the bits of a word

Kill_Out signals are 0, and a NOR gate provides the matching signal at the output (Figure 3.6).

Tables 3.1 and 3.2 show the Karnaugh's maps of the two cells (KOXORAM and NKOXORAM).

Table 3.1: Karnaugh map of the NKOXORAM cell

NKill_Out		(SL, A)			
		00	01	11	10
Kill_In	0	1	1	1	1
	1	0	1	0	1

Table 3.2: Karnaugh map of the KOXORAM cell

Kill_Out		(SL, A)			
		00	01	11	10
NKill_In	0	1	0	1	0
	1	0	0	0	0

The dynamic power due to switching activity is the major source of power consumption. Therefore, a special care has been adopted in the design of interconnections for input data.

First of all, different input lines are used: BLs carry the input data to be written into the memory during the writing. SLs propagate the data coming from the input bus to be compared with the data stored inside the CAM. In this way, the search lines are not affected by the parasitic capacitances of the BLs and the switches controlled by write lines (WL). Moreover, the SL signals are routed at a larger distance each other, and they are sufficiently spaced from other signals in the same metal layer, to reduce their capacitance.

BLs and WLs are much less critical, because the memory is written only at the beginning of the operation.

Finally, the aspect ratio of memory cells is chosen in such a way to reduce the cell height. As SLs are propagated vertically, height reduction helps in reducing interconnection capacitance. To reduce further the length of SLs, two cells are merged together. The same contact in the SL is used to drive the signal to four MOS gates (two gates for the MOS transistors in line 0, and two gates for the MOS transistors in line 1).

3.3.1 Schematic Diagrams and Layout

Figures 3.7 and 3.8 illustrate the schematic diagrams of the KOXORAM and of the NKOXORAM cells, respectively.

Figure 3.9 shows the layout of four neighboring cells. The cell height is $0.91\,\mu\mathrm{m}$, and the width is $6.24\,\mu\mathrm{m}$ ($3.12\,\mu\mathrm{m}$ for each block of two cells).

The total decoupled capacitance associated to the SLs is $0.27\,\mathrm{fF}$ for two cells ($0.2\,\mathrm{fF}$ due to gate capacitances, and $0.07\,\mathrm{fF}$ due to metal-metal capacitances).

3.3.2 KOXORAM cell working modes

The CAM cells have three different modes, as described in Table 3.3:

- **compare**, it is used during compare procedure;
- **always match**, it is used to implement the variable resolution;
- **never match**, it is used to guarantee the setup and hold timings and during test procedures

In the KOXORAM cell this working modes are given by the synthesized logic function. In NKOXORAM cell it is necessary to modify the input connections inverting the SL and SLN signals and memorizing an inverted data swapping the BL with the BLN signal, as shown in Figure 3.10.

3.3.3 Clockless logic

The CAM cell comparison procedure is completely clockless and without an enable signal. The setup and hold times are guarantee using the never-match working mode on one CAM cell as described in Figure 3.11. Precisely, during the setup and hold times the SL<17> and SLN<17> signals are fixed to '1' to prevent the kill signal propagation, and so the final matching, instead during comparison the value of all SLs and SLNs correspond to the input values coming from the detector.

Table 3.3: (N)KOXORAM cell working modes and signals at the interface between the CAM memory block and the standard cells logic

Working mode	A	SL	SLN	Kill_Out	NKill_Out
compare (match)	1/0	1/0	0/1	0	1
compare (not match)	0/1	1/0	0/1	1	0
don't care	–	1	1	0	1
never match	–	0	0	1	0

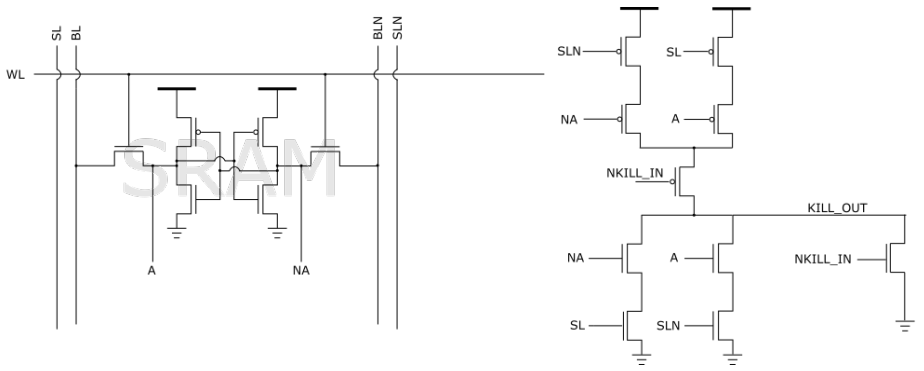


Figure 3.7: Schematic diagram of the KOXORAM cell

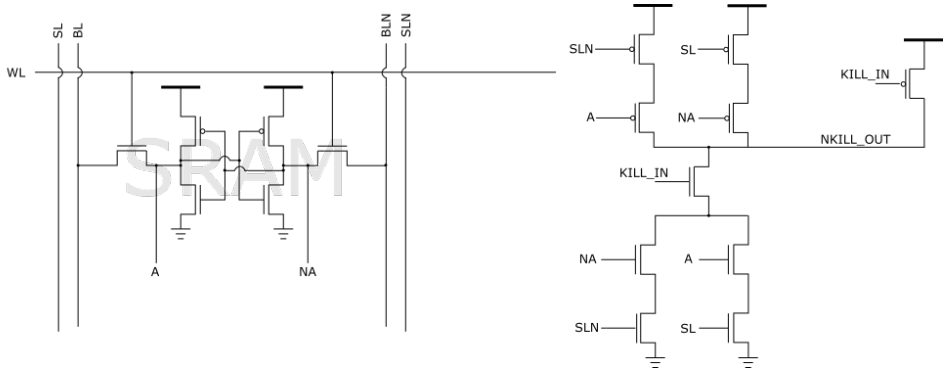


Figure 3.8: Schematic diagram of the NKOXORAM cell

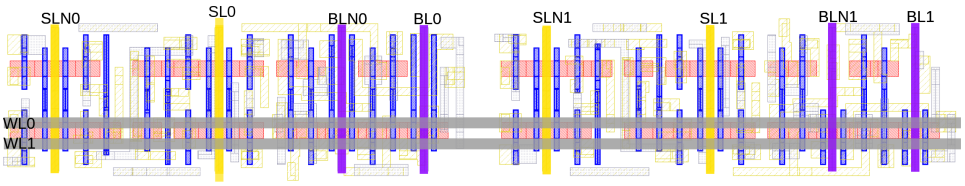


Figure 3.9: Layout of four cells, arranged in two groups, to reduce interconnection capacitances of search lines (SL/SLN)

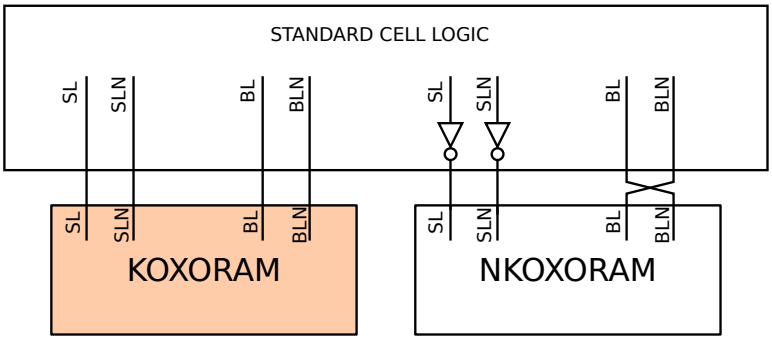


Figure 3.10: Connection scheme between standard cells logic and (N)KOXORAM CAM cells.

3.3.4 Simulations

The KOXORAM cell has been extensively simulated to evaluate its performance before fabrication. Interconnection parasitics have been extracted and used for simulation in worst-case corners. The cell has been simulated with a nominal supply voltage $V_{\mathrm{DD}} = 1\,\mathrm{V}$, and input data rate $f = 200\,\mathrm{MHz}$.

Simulation results demonstrate that in an 18-bit CAM made with the new cell the match signal is asserted to the output with a typical delay of $0.9\,\mathrm{ns}$. The maximum delay, in the slow-slow corner, is $2.5\,\mathrm{ns}$ which is less than the input signal period ($5\,\mathrm{ns}$). Figure 3.12 shows the delay of the matching signal at the output, in all the worst-case corners.

The average energy is 0.69 fJ/bit, which is a figure lower than the consumption of other designs in the technical literature (e.g., [63]).

The lower power consumption is due to the reduction of both the switching activity of cells and the search line capacitances.

3.3.5 Measurements

The AM07 has been fabricated and tested [31]. The chip is fully functional and operates up to 200 MHz. In order to measure the power consumption of cell arrays, input data has been stimulated with a simple dynamic sequence. Input words change between non-matching data (`0x00000`) and matching data (`0x1ABE0`). 500 rows of the memory bank have been written with `0x1ABE0`, and 3596 rows with (`0x3FFFF`).

Baseline current consumption has been measured with constant input (i.e., without switching activity on input data except for the SL<17>s). Current and energy consumption are measured by enabling one memory bank at a time (4×1024 patterns of 8 words each) with a switching activity equal to 0.5, i.e., when one half of the input bits are changing their value with respect to the data in the previous clock period; Δ is defined as the difference between the measurement with switching activity and the baseline measurement. Results of a measurement of 53 chips at 184.32 MHz are summarized in Table 3.4. It is worth remarking that we found a good agreement between measurements and simulation results (within $\pm$ 10%).

Table 3.4: Current/energy consumption, measure done on 53 AM07.

	Current [A]		# bits involved	average energy	sim Calibre	
	Mean	STD dev.		[fJ/bit]	[fJ/bit]	%
Baseline	$3.80 \cdot 10^{-2}$	$8 \cdot 10^{-4}$	524288			
Activity on 1/8 input buses	$4.70 \cdot 10^{-2}$	$1.3 \cdot 10^{-3}$	524288			
Δ (Activity on 1/8 input buses)	$9.04 \cdot 10^{-3}$	$7.3 \cdot 10^{-4}$	65536	0.748	0.69	108%

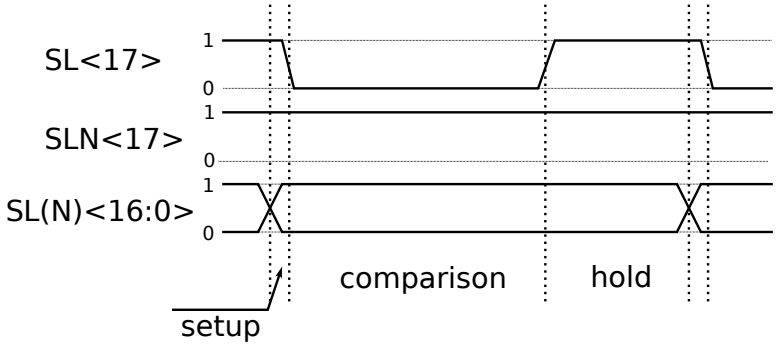


Figure 3.11: Compare timings of a KOXORAM CAM cell.

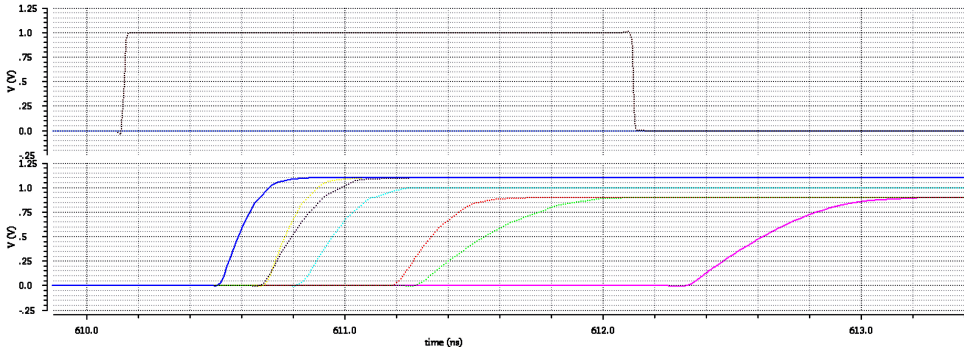


Figure 3.12: Simulation result of an 18-bit CAM cell in different corners. Propagation delays at the end of the chain are: 0.9 ns in typical case (cyan), 0.5 ns in fast-fast corner (blue), and 2.5 ns in slow-slow corner (pink).

3.4 Improvements for AM08

Although the AM07 test shows good results, some design aspects have to be improved for AM08 and AM09. The standard cell logic occupies a lot of silicon area and the routing is very dense, it will become even more complicated in future AM chip.

The quorum circuit is the standard cell logic circuit that occupies more area, hence we redesign it with a full custom approach to reduce area occupation and routing complexity.

To reduce the power consumption peak due to parallel propagation of data inside SLs [52] we design a digital PLL that gives eight different clock with different phases at the eight chip cores.

SRAM cells presents metastability issues, so the SRAM cells has been modified to avoid it.

3.4.1 Hi performance technology

AM07 is fabricated using the low power version of the 28 nm technology. A low power technology has a low leakage current to reduce power consumption during standby. For this reason it is great for battery powered devices, that stays for a lot of time in standby mode. The drawback is that transistors can output less current than in high-performance technologies. Hence it is necessary to add more buffer along signal paths, the fan-out of a logic gate is lower and the maximum length of interconnection lines is shorter.

The high performance version of the technology is oriented to devices that are not powered with batteries and have to perform heavy computing operations.

We choose the High performance technology for AM08 because being faster it permits to reach the aggressive goals. Moreover the improvement due to optimization of routing, fan-out and signal paths will permit us to reach a power consumption similar to low power technology.

3.4.2 KOXORAM+

To reach the specification target of 400 MHz in typical case it is necessary to change the architecture of the memory row. Due to the limited number of metal layers for routing and the propagation speed of the technology used in AM07 the chain that has a maximum of nine KOXORAM cell in series.

The new technology is faster and provides more metal layers suitable for routing, thus in AM08 the row has a series of 12 KOXORAM cell, as shown in Figure 3.13. This architecture is also less power consuming: the power used to propagate the data in SLs is the same, but less power is used to propagate the

kill signal. The probability of cell matching can be approximated to 50%. Being P_{Kill} the power needed to propagate the kill signal, the architecture in Figure 3.6 consumes $2\sum_{n=1}^{9} P_{Kill}/2^n \approx 2P_{Kill}$ and the architecture in Figure 3.13 consumes $\sum_{n=1}^{6} P_{Kill}/2^n + 2\sum_{n=7}^{12} P_{Kill}/2^n \approx P_{Kill}$.

Simulation with back-annotated netlist shows an energy consumption of 0.42 fJ/bit, approximately 30% less than the former KOXORAM memory. KOXORAM+ fulfills the speed requirements of 400 MHz in typical and fast case and 250 MHz in slow. Table 3.5 shows the minimum timings in each simulation corner.

3.4.3 SRAM metastability

AM07 tests shows that the current consumption at startup is higher than after writing the CAM SRAM memories. The current at startup is $27 \pm 6\,\mathrm{mA}$ and after the writing is $4.40(0.01)\,\mathrm{mA}$ per 16 $\times$ 1024 patterns, Figure 3.14 shows the distribution of the 100 current measurements at startup of a single chip. This behavour is typical of SRAM memories that present metastability. Indeed if the output and the input of an inverter are at the same voltage there is a unstable equilibrium point, an SRAM in that working point consumes a lot more power than SRAM with definite states.

The AM09 chip will contain 3$\times$128Ki patterns, and the power consumption at startup due to metastability would be $\approx 0.5\,\mathrm{A}$. To reduce the number of metastable SRAM we design an asymmetric SRAM cell.

Butterfly diagram

Butterfly diagrams are commonly used to study the properties and the metastability of an SRAM [64, 65, 66].

An example of butterfly diagram is shown in Figure 3.15. The butterfly diagram is drawn plotting the two characteristics of the SRAM inverters on the same graph. It is possible to extract a lot of information from this diagram, but we focus our attention on metastability.

An SRAM is metastable when the crossing point between the two characteristics is placed on the bisector of I and III quadrants (where $V_A = V_B$).

Table 3.5: Timings of KOXORAM+ memory

$[ns]$	tt	ss	ff
setup	0.1	0.1	0.1
compare	0.8	1.7	0.5
hold	0.4	0.9	0.2

SRAM design

To characterize the SRAM we run a Montecarlo simulation on the space of technology parameters, with 1000 points, and we plot the butterfly diagrams and the crossing points. The results of the Montecarlo simulation are shown in Figure 3.16.

In particular in Figure 3.16a there is the simulation of a completely symmetric SRAM with minimum dimension transistors. The crossing points are placed across the bisector, indicating a non-negligible metastable behavior.

To reduce the number of metastable SRAM cells we unbalance the SRAM enlarging the width of a p-MOS transistor from 100 nm to 360 nm and opposite n-MOS transistor from 100 nm to 180 nm. This modification is done without increasing the area of the KOXORAM cell, as shown in Figure 3.16d, the extra active area helps also to meet the design rule of minimum active area density.

In Figure 3.16b there is the simulation of the asymmetric SRAM, there are less point near the bisector hence metastability will be rarer than in the symmetric configuration. In Figure 3.16c there is a comparison of the distribution of distance of the crossing points between symmetric and asymmetric SRAM. The distribution shows that the asymmetric SRAM has a negligible probability to show a metastable behaviour.

3.5 Quorum circuit

The AM architecture is structured in words of 18 bits. Each word contains the spatial coordinates of a group of neighboring sensor pixels on the same layer of the detector; 8 words (from 8 different detector layers) make a pattern, and every matching between an input word and the stored data triggers a Set-Reset (SR) latch to high-logic value [15]. To perform the trigger operation we need to know how many patterns are matching. An obvious solution consist in counting how many SR latches have been triggered by the input event data. To obtain a good performance, we need to consider all the possible thresholds, from 0 to 8 matches. For this task, a population count circuit is used.

Conventional population count circuits use full adders and half adders. Dalalah et al. [67] propose an 8-bit circuit that employs CMOS logic gates, with 162 transistors in total. Pedroni, in [68] presents an 8 bit sorter composed by 56 logic gates. Despite the literature solutions optimize the transistor number, for 28nm technology is preferable to design a repetitive architecture to minimize the number of interconnections in order to reduce area and dynamic power consumption.

In AM07 chip, the population count circuit is designed by synthesizing a behavioral VHDL code with the Cadence RC software, and it is implemented using the standard-cell library. With this approach, the area occupied by the population

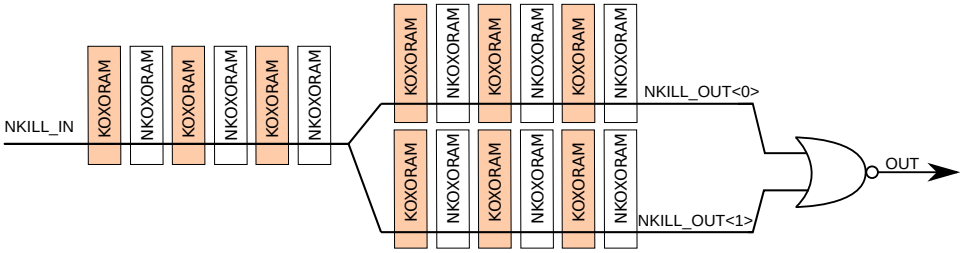


Figure 3.13: Architecture of a single row of KOXORAM+ cell

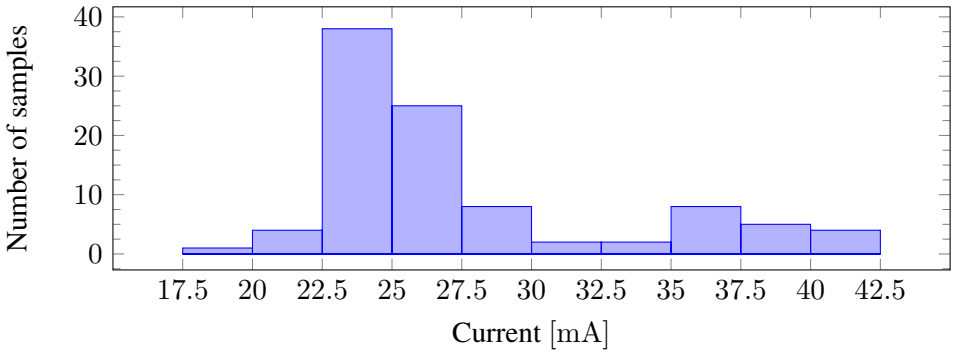


Figure 3.14: Histogram of the measurements of current due to SRAM metastability at startup. The histogram shows the distribution of 100 measurements of a single chip.

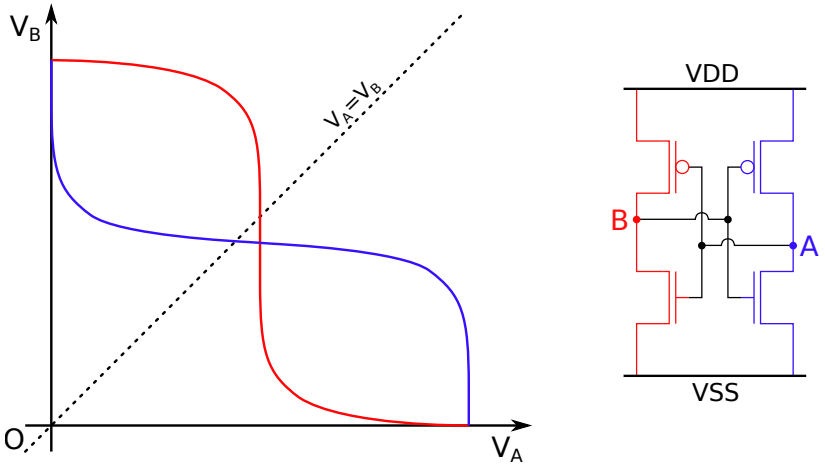


Figure 3.15: Example of butterfly diagram

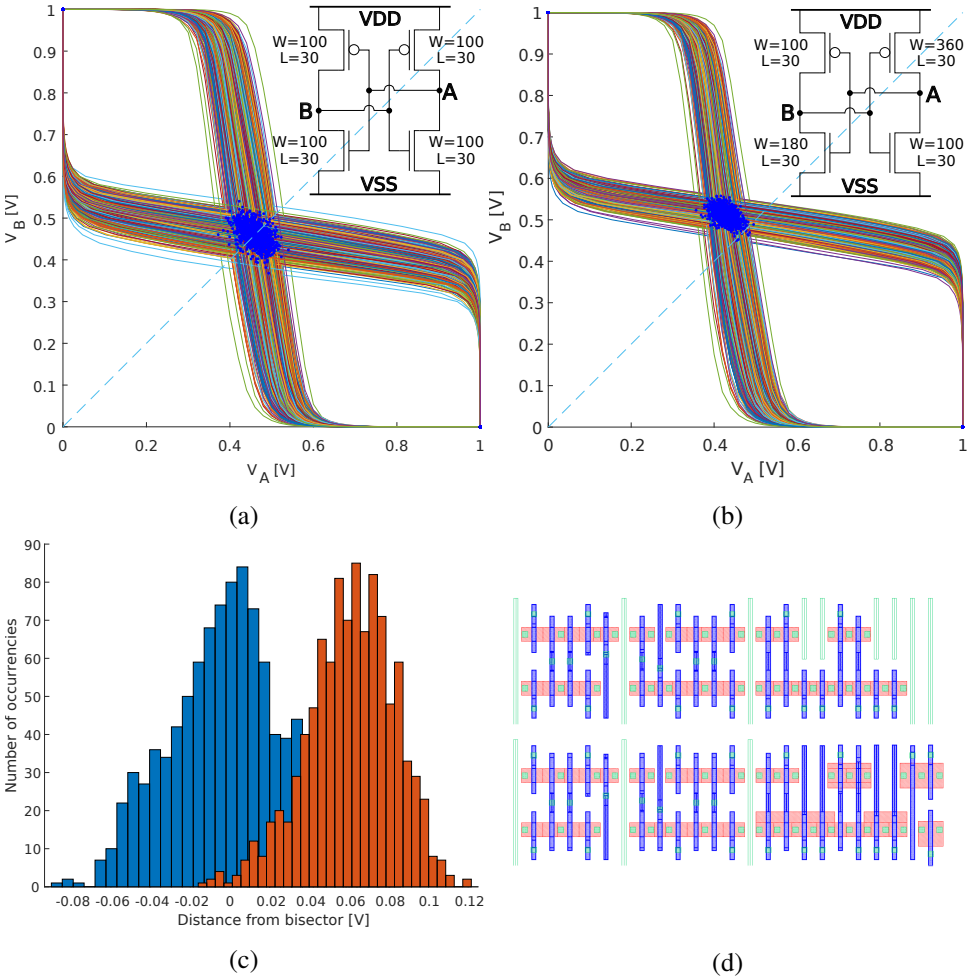


Figure 3.16: (a),(b): Montecarlo butterfly diagrams showing the crossing points in blue. At the top right of each diagram there is the schematic of the simulated SRAM cell reporting the width and the length of each MOS transistors (in nm). (c): distribution of the distance of the crossing points from the bisector. The symmetric SRAM is plotted in blue, the asymmetric in red. (d): layout of the KOXORAM cell. On top there is the layout of the symmetric version and on bottom the layout of the asymmetric one.

count circuit is about three times the area of the CAM blocks. The RC compiler reports a total area of $138\,\mu m^2$.

For this reasons it is designed a new full custom circuit, where the single element is based on the *bubble-sort* algorithm and uses a divide-and conquer approach to perform the population count with a combinational logic implemented in fully-CMOS approach. As in the bubble-sort algorithm in software, the circuit swaps the logic values at the inputs to give an ordered set at the output. The outputs are given to the threshold selector (designed using a multiplexer) that selects the output corresponding to the required threshold, as shown in Figure 3.17.

The designed circuit is asynchronous, and it is made of three layers of combinational logic. The output of the proposed circuit passes trough a full custom control logic circuit, as described in [69]. This control logic is composed by:

- A threshold selector that selects the output corresponding to the chosen threshold (Figure 3.17)
- SR latches that receive in input global signals and WL to put the output of the single quorumA circuit to '1' or '0' independently from the inputs (this feature is useful during the test procedure).
- Logic gates that uses global signals to put the output of the all quorum circuit of a 64 word memory block to '1' or '0' independently from the inputs (this feature is useful during the test procedure)
- an SR latch that stores the read flag signals that comes from the readout tree and indicates if the readout procedure was already made.

The envisaged solution employs more transistors than the circuit proposed in [69], but it leads to a more compact layout design, which occupies a smaller silicon area.

The bubble-sort circuit exhibits also a better power consumption: as it is an asynchronous circuit, it has less switching activity with respect to a synchronous circuit.

The circuit goal is to obtain the number of zeros (or ones) within an 8-bit bus. The positions of zeros are not a-priori predictable. As for the bubble-sort algorithm, the idea consists in sorting the zeros and the ones (to separate them), and in finding the position of the transition from 0 to 1 in the sorted array. To achieve this result, we employ a combinational logic which shifts the ones towards the Most Significant Bit (MSB) position of the output bus and the zeros towards the Least Significant Bit (LSB) position.

In $28\,nm$ technologies, the on-resistance of MOS transistors is too high to allow a design with eight transistors connected in series. For this reason, it is not possible

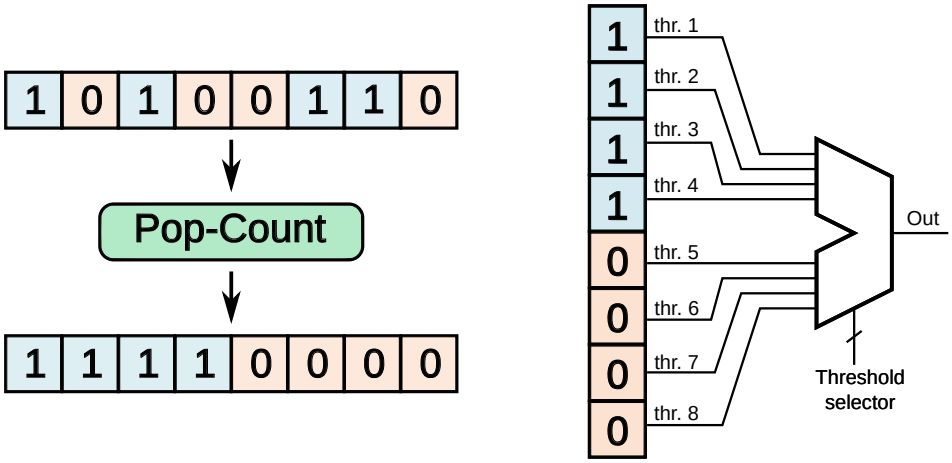


Figure 3.17: The Pop-Count circuit swaps the logic values at the inputs and gives an ordered set at the output. The output are then processed with a Threshold Selector, that selects the output corresponding to the wanted threshold.

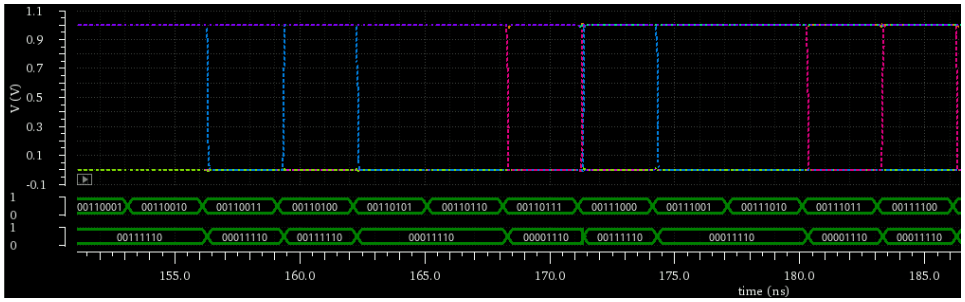


Figure 3.18: Top waveforms: analog signals; middle waveform: input bus; bottom waveform: output bus. The zeros are shifted to the MSB positions, and the ones to the LSB positions

to design a logic with 8 input/output bits, because its timing and functionality will be degraded. Therefore, we limit the number of transistors in series to four.

The overall circuit is obtained by using two sets of three cascaded 4-bit bubble-sorter circuits (Figure 3.19) and by scrambling some wires. A cascade of three stages is needed to guarantee the complete sorting for all input bit configurations.

The boolean expressions for the elementary cells are:

$$\overline{Z} = A + B + C + D \tag{3.1}$$

$$\overline{Y} = (A + D) \cdot (B + C) + A \cdot D + B \cdot C \tag{3.2}$$

$$\overline{X} = C \cdot D \cdot (A + B) + A \cdot B \cdot (C + D) \tag{3.3}$$

$$\overline{U} = A \cdot B \cdot C \cdot D \tag{3.4}$$

The logic generates inverted signals at the output. From the four boolean expressions (3.1–3.4), we derived a fully-CMOS circuit made of stacked transistors, using the Euler's graph method. Figure 3.20 shows the schematic diagrams of the four parts of each block. The overall circuit is made of 48 transistors. Several transistors are stacked in the layout, in order to save silicon area. Figure 3.21 shows the layout of the elementary cell.

Compared with other solutions, this circuit occupies less silicon area. In fact, the circuit described in the previous work [69] is larger by a factor of 1.17, and a circuit with the same functionality implemented with the foundry standard cell library is larger by a factor of 5.

3.5.1 Simulation Results

Simulations have been carried out in different corners. Simulations covered all the possible 2^8 input bit combinations. Output results have been checked with a behavioral model to guarantee the full functionality.

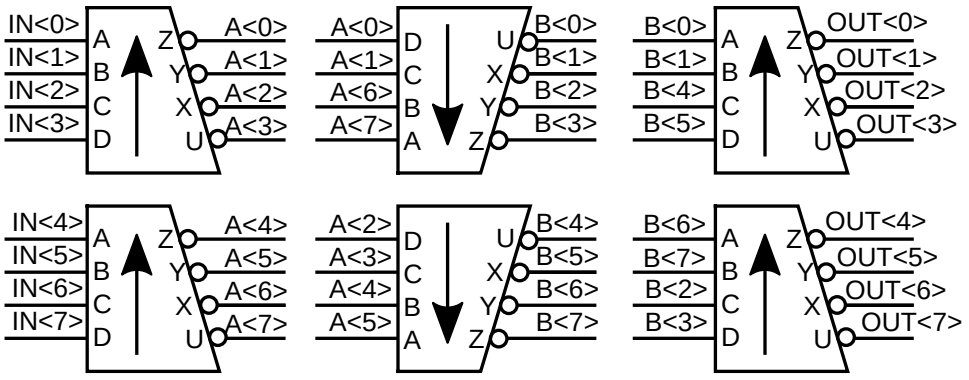


Figure 3.19: Schematic diagram of population counter

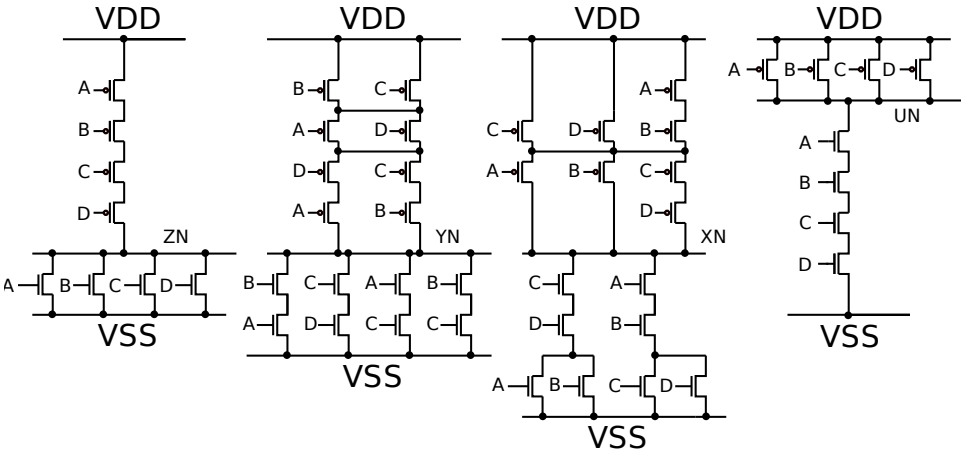


Figure 3.20: CMOS schematics of the elementary cell. The four sub-circuits implement the four boolean expressions in (3.1)–(3.4), respectively.

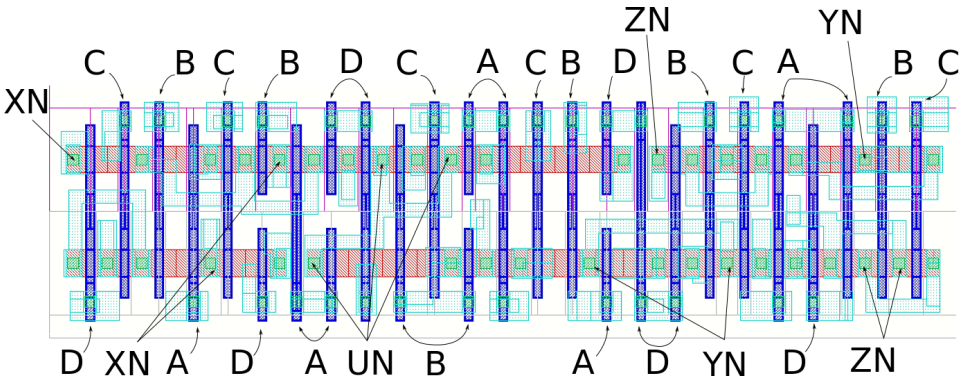


Figure 3.21: Layout of the elementary cell. Transistors have been stacked as much as possible, to obtain a very compact design

Results demonstrates the functionality of the circuit at the required speed: 400 MHz. To avoid possible spurious transitions, output data of the control logic has been registered thanks to a conventional delay FF. Figure 3.18 shows a detail of the simulated waveforms. Simulations have been performed with Cadence Analog Mixed Simulator (AMS) and back-annotated via Mentor Graphics Calibre PEX Extractor. The stimuli are generated using the output of a 8-bit counter written in Verilog code.

The circuit consumes 3.75 aJ/bit. The previous full-custom implementation occupies five times the area of the full-custom circuit and contains more transistors and interconnections. In this technology the dynamic power consumption due to charge and discharge of interconnections is not negligible. For this reason we can tell that the power consumption of the full-custom quorum is certainly lower than the standard-cell one.

3.5.2 Design Verification

Layout Versus Schematic (LVS) and Design Rule Check (DRC) have been performed and results are clean.

3.6 Digitally Controlled Oscillator

AMs are massively parallel circuits. When an AM chip switches from 'idle' mode to 'compare' mode the current rises from zero to several Ampere in about $0.1\,\mathrm{ns}$ and current peaks are synchronous with the clock.

This current consumption generates a ripple on the supply voltage of the chip that can affect the correct functionality of the whole device [59].

For this reason AM09 will implement a circuit to reduce this current peak, splitting the single peak in eight smaller peaks.

This procedure is done using a PLL circuit that generates an internal clock that is eight times faster than input clock. Then with the PLL clock it is possible to generate eight different clock phases with the same frequency than the input clock, as Figure 3.22 shows.

The PLL is based on a Digitally-Controlled Oscillator driven by a fairly simple state-machine. This circuit solution was chosen instead of typical Voltage-Controlled Oscillators (VCOs). In fact VCO schematics are based on operational amplifiers and passive components. The $28\,\mathrm{nm}$ technology is mainly intended for digital circuits and generally do not provide modules for passive components that can be used successfully in the design of analog circuits. Moreover, the value of these analog components suffer from great process variability.

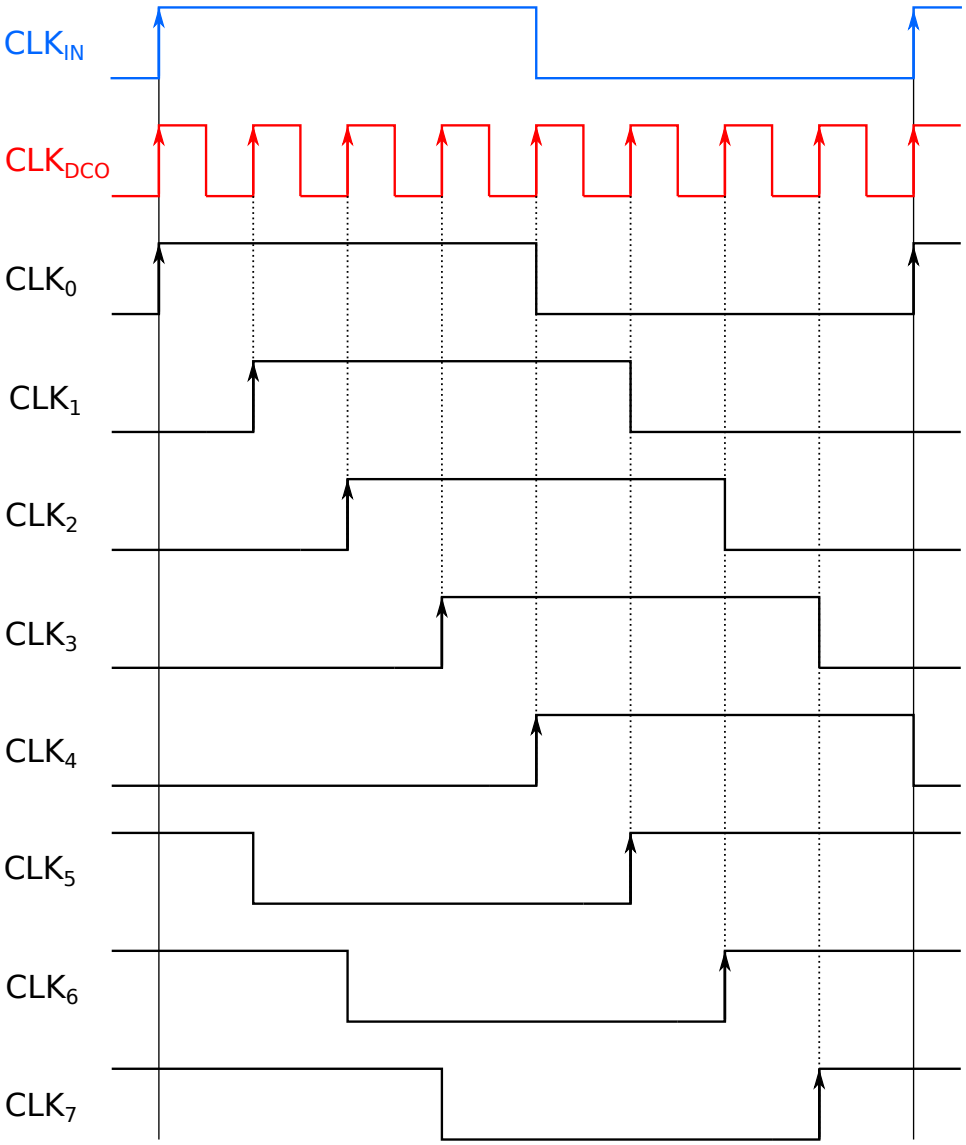


Figure 3.22: The blue signal is the reference clock given in input to the AM chip. The red signal is the clock generated by the PLL. The black signals are the clocks with the same frequency than the input clock, but different phases.

For these reasons the solution adopted is a ring oscillator with full-digital frequency control, in which the analog part of the circuit is minimized and realized only with active devices.

The main requirements for the DCO are: a frequency from 2 GHz to 3.2 GHz, a power consumption of less than $10\,\mathrm{mW}$ and a phase noise not exceeding $1/8$ of the clock period. Linearity is not required, while monotonicity is a strong requirement.

3.6.1 Circuit Structure

The DCO is based on a ring oscillator with a frequency range from $2\,\mathrm{GHz}$ to $3.2\,\mathrm{GHz}$ in typical case. The oscillation frequency is tuned by acting on three separate thermometric controls. The first one, namely the "coarse frequency control" (12 bits), selects the length of the delay chain in the ring. The fastest configuration is made of just an inverter and one delay element, while the slowest is made of an inverter and 25 delay elements. The two other frequency controls, the "semi-coarse control" (6 bits) and the "fine control" (63 bits) are used to adjust the delay introduced by each element.

The length of the chain should be adjusted only at power-up because is intended mainly to select between different operating frequencies (e.g., $2\,\mathrm{GHz}$ or $3\,\mathrm{GHz}$) and to compensate for frequency variations due to the fabrication process. During operation, frequency and phase tracking is performed only with the "fine" and "semi-coarse" controls. The multiplexer can be used also to activate-deactivate the oscillator and to provide a start-up stimulus to avoid possible equilibrium states. The bias circuit takes a $1\,\mu\mathrm{A}$ current reference, turns it into a voltage bias and distributes it to the delay elements. Figure 3.23 shows a block diagram of the ring oscillator.

Each delay element is made of two inverters and a Local Voltage Bias Generator (LVBG).

The first inverter, marked with "D" in Figure 3.23, is the main source of delay,

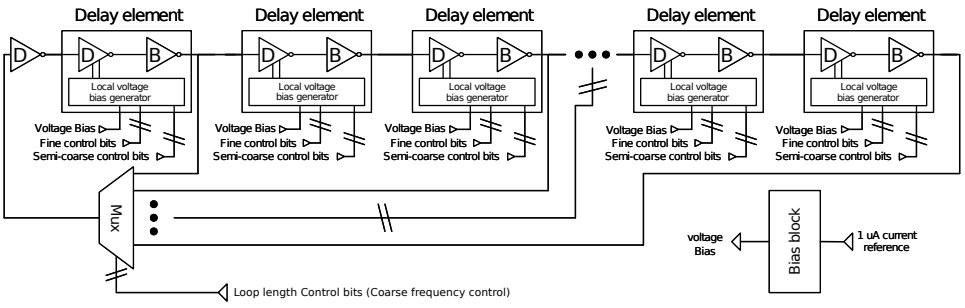


Figure 3.23: Block diagram of the ring oscillator. For clarity, only 5 delay elements of 25 are shown. The bias block in the bottom-right corner produces a voltage bias used by the delay elements. The multiplexer has 13 inputs (only 4 are shown).

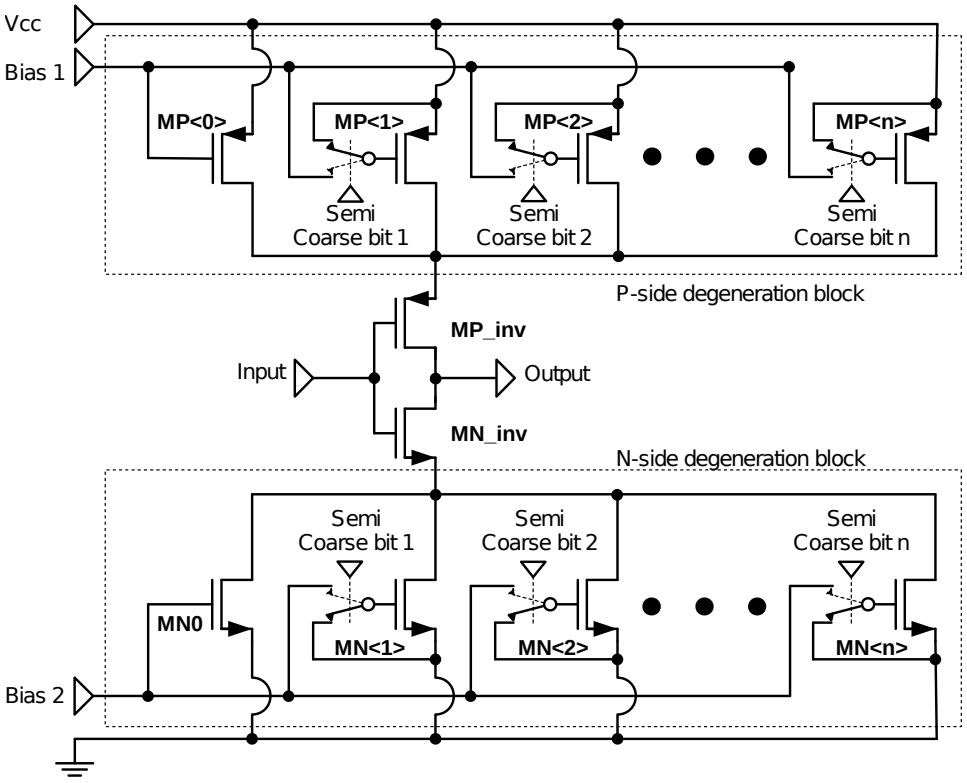


Figure 3.24: Schematic of the first inverter inside the delay element.

while the second inverter (marked with a "B" letter in Figure 3.23) works like an inverting buffer. Figure 3.24 illustrates the schematic diagram of the main delay element. The two voltage bias BIAS1 and BIAS2 are propagated to the degeneration transistors. The speed of the inverter made of transistors $\mathrm{MP}_{\mathrm{inv}}$ and $\mathrm{MN}_{\mathrm{inv}}$ is determined both by the number of active degeneration transistors and by their conductance. The number of active degeneration transistors is the frequency "semi-coarse" control, while the analog bias BIAS1 and BIAS2 depend on the "fine" control. The source of both the N-MOS and P-MOS transistors is degenerated with an array of transistors. These arrays are responsible for a reduction in the current flow from the power rails to the load connected to the output of this stage (and vice-versa) during the switching phase of the NOT port. This translates directly into a delay between the incoming signal and the one produced at the output. The second inverter decouples the first from any possible capacitive load connected to the output of the cell and squares up the signal.

Previous works [70, 71] implement similar circuits but with the gate of the degeneration transistors connected either to VDD or VSS. This means that in similar solutions the degeneration transistors contribute to the switching current flow with their full conductance. In the same work was pointed out how the charge stocked inside the parasitic capacitances of the degeneration transistors gives a relevant contribution to the current flow during the switching phase. With this circuit solution the frequency of the oscillator is heavily dependent on the threshold voltage, which, in turn, has a relevant process variation. Our solution implements a double control of the behavior of the degeneration blocks. In order to limit the frequency dependence on the threshold voltage, the conductance of the degeneration transistors is controlled with a mirror-like circuit and derived from a current reference.

Each degeneration transistor can be either switched "ON" or "OFF"; however, when in the "ON" state, the gate voltage is not fixed to VDD or VSS, but it can span across the available voltage dynamic range. In this way, the conductance of each degeneration transistor can be modulated continuously. The digital control that activates or deactivates each transistor is the "semi-coarse control" while the analog voltage that biases the gate of the degeneration transistors when in "ON" state is determined by the "fine control". The activation of the degeneration transistors is performed following a thermometric code to ensure monotonicity of the control. Since there are two degeneration blocks, one on the N-side and one on the P-side, the "fine control" is translated into two analog voltages, called BIAS1 and BIAS2, that set the conductance of both the N-type degeneration transistors and the P-type ones accordingly. The switches connected to the gate of the degeneration transistors are designed with minimum capacitance, ensuring fast frequency response after switching and minimum voltage bounce on the bias networks.

The voltage biases BIAS1 and BIAS2 are provided by a local voltage bias

generator built inside each delay element (see Figure 3.25). BIAS1 and BIAS2 are produced by mirroring the variable current I_{ARRAY} on a cascade of two trans-diodes. The I_{ARRAY} current is produced by an array of P-type transistors. These transistors can be switched "ON" and "OFF" with the "fine control". When in the "ON" condition, the gate voltage is fixed to a voltage bias derived from the main $1\,\mu\mathrm{A}$ current reference. The array of digital signals that drives this array of transistors must follow a thermometric code in order to guarantee monotonicity.

In fact, monotonicity is critical to ensure the stability of the digital feedback loop of the PLL in which the DCO is inserted. The size of transistors in the array are not uniform because the frequency response of the DCO to the current I_{ARRAY} is not linear. These transistors have been sized in order to maximize the linearity of the code-frequency response across the available dynamic range. The choice of replicating the local voltage bias generator block inside each delay element, instead of having just one that generates the references for all the delay elements, is dictated by the requirements in response speed of the DCO to frequency control changes. Since BIAS1 and BIAS2 are analog signals, their bandwidth is limited on one side by the parasitic capacitance connected to the line and on the other side by the transconductance and the bias current of MP2 and MN2.

Simulations demonstrated that, keeping a single central voltage bias generator connected to all the delay elements, the frequency variation of the DCO induced by a code change could not be faster than $30\,\mathrm{ns}$. Embedding a local voltage bias generator inside each delay element, the length of the BIAS1 and BIAS2 paths was considerably reduced together with the associated parasitic capacitance. Moreover, with this circuitry solution each local voltage bias generator drives just one array of degeneration transistors, whose gate capacitance is dominant in determining the time constant of the frequency changes. This solution reduces the frequency response time from $30\,\mathrm{ns}$ to $10\,\mathrm{ns}$.

3.6.2 Layout Design

The layout of the device was conceived keeping in mind that the propagation delays introduced by metal connections and their associated parasitic capacitances are absolutely critical and should be minimized, being process-dependent. The block diagram of the delay elements placement is reported in Figure 3.26. The delay elements are stacked in two columns, leaving the inverters of the ring in the middle. The signal bus of all the possible ring routings runs in the middle and connects the delay elements to the multiplexer.

The shortest ring length is chosen when the chip belongs to the slow fabrication corner and the speed limitations of the DCO are potentially an issue. Thus the total signal path, especially for the shortest ring configuration, is minimized and the

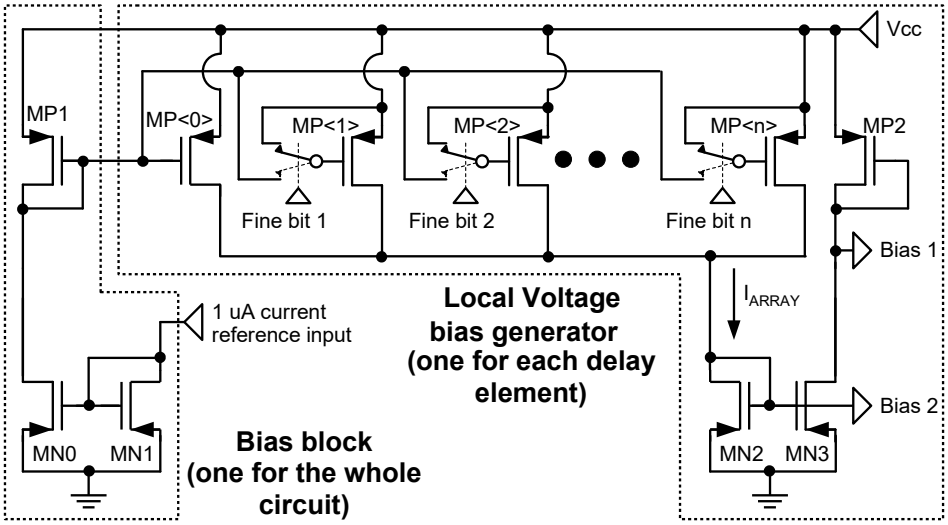


Figure 3.25: Schematic of the Local voltage bias generator inside each delay element. The schematic of the bias block (one for the whole circuit) is reported on the left.

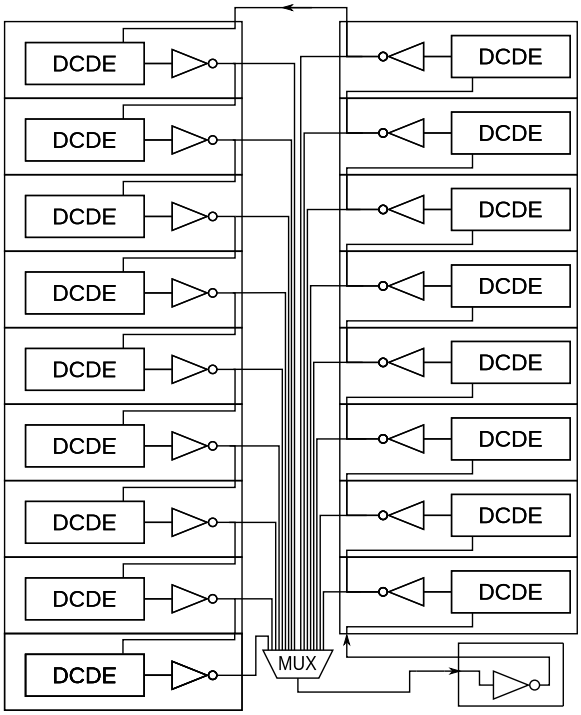


Figure 3.26: Block diagram of the delay elements placement in the layout.

inverters involved must be put very close to the multiplexer. The delay elements are packed in a way to keep the relative distances between the inverters of the ring as low as possible.

Each delay element has an area of $21.9\,\mu\mathrm{m} \times 6.6\,\mu\mathrm{m}$ and the signal path inside is minimized (see Figure 3.27). For this reason, the critical transistors are packed on one side, leaving the static ones in the body of the block. The major part of the area is occupied by the local voltage bias generator. The dimensions of each building block are reported in Table 3.6. The long interconnections between the delay elements and the MUX constitute a long parallel bus that travels across the whole DCO. Such signal lines are realized alternating metal-2 and metal-3 to reduce the parasitic capacitances between them. As common practice, local wirings are realized with low-level metals while the power supply lines are made with higher-level ones. The channel length of the transistors inside the LVBG are deliberately chosen not minimal to reduce the process variability of their electrical parameters.

3.6.3 Simulations

In order to evaluate the performance of the DCO some post-layout simulations have been performed. These include both transient at circuit start-up and long-term simulations varying the frequency control code. The device demonstrated to start oscillating properly independently from the start-up slope of the power voltage lines. Transient simulations never showed the appearance of higher harmonics or meta-stable states.

Since this circuit is strongly dependent on the fabrication speed corners, the DCO behavior has been simulated in every corner in order to ensure its compliance with the AM09 specifications.

Figure 3.28 shows the oscillation frequency as a function of the fine tuning code, for different values of the coarse codes, in typical case. The achieved granularity is fine enough to perform the desired phase-locking with acceptable phase error.

The DC current consumption ranges from $1\,\mathrm{mA}$ (2 GHz) to $5\,\mathrm{mA}$ (3.2 GHz)

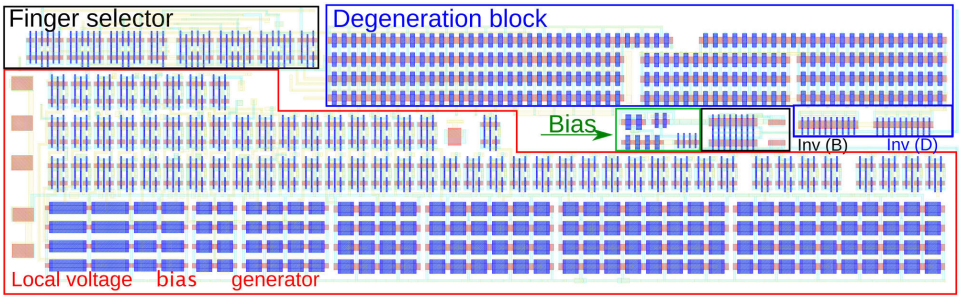


Figure 3.27: Layout of a single delay element

in typical corner. The frequency variation due to the temperature variation is less than the variation due to corners. The influence of the temperature can be adjusted changing the control bits during time.

Table 3.6: Size and relative area occupation of the delay element's building blocks.

Description	Width [µm]	Length [µm]	Area [μm^2]	Percentage [%]
LVBG	21.6	3.7	79.3	55
Deg. block	14.5	2	29	20
Non-active area			24.9	17
MUX	6.8	1	6.8	5
Bias transistors	3.2	1.1	3.5	2
Inverters	1	0.9	0.9	1
Total	21.9	6.6	144.3	100

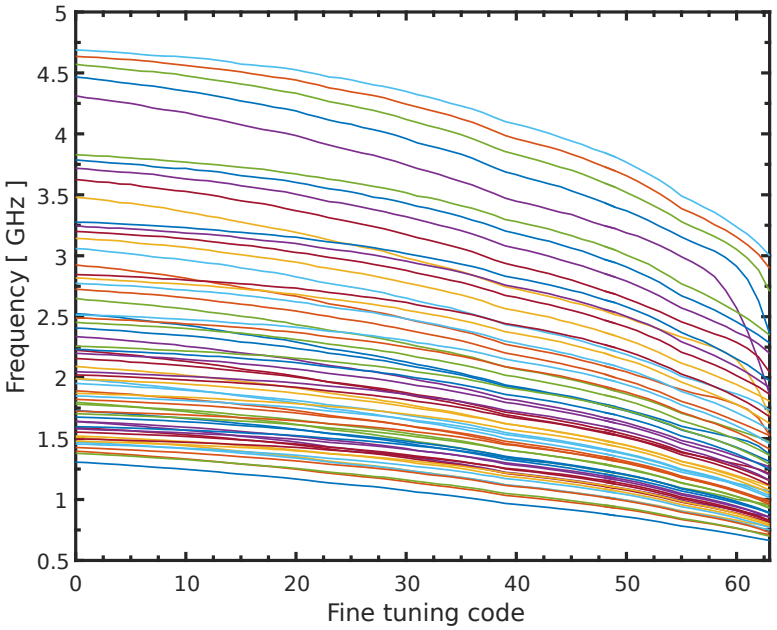


Figure 3.28: DCO frequencies versus fine tuning code in typical case.

Part II

Switching Lattices

Chapter 4

Technology Description

A switching lattice is a two-dimensional network of four-terminal switches. Each switch is linked to the four neighbors of a lattice cell, so that these are either all connected or disconnected, as Figure 4.1 shows. A Boolean function can be represented using a switching lattice associating each four-terminal switch to a Boolean literal: if the literal has value 1 the corresponding switch is connected to its four neighbors, otherwise it is not connected. In this model, the Boolean function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges (see Figure 4.3 for an example). The synthesis problem on a lattice thus consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size.

Consider, for instance, a nanowire array where each crosspoint is controlled by an input voltage. We consider crosspoints that behave like four-terminal switches controlled by an input signal and therefore the proposed nanowire crossbar array can be modeled as a lattice of four-terminal switches. Note that, in general, crossbars can also be modeled by programmable contacts [23]. Conventional implementations typically employ SRAMs for programming crosspoints; other techniques have been suggested for implementing programmable crosspoints such as bistable switches that

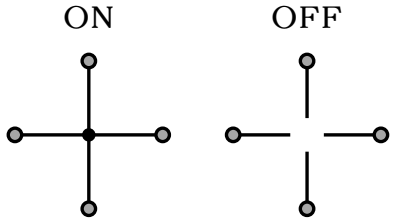


Figure 4.1: On and Off 4-terminal switch

form memory cores, molecular switches and solid-electrolyte nanoswitches [25].

4.1 Boolean function implementation

A Boolean function can be implemented by a lattice in terms of connectivity across it (Figure 4.2):

- each four-terminal switch is controlled by a Boolean literal;
- each switch may be also labelled with the constant 0, or 1;
- if the literal takes the value 1, the corresponding switch is connected to its four neighbours, else it is not connected;
- the function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges;
- input assignments that leave the edges unconnected correspond to output 0.

For instance, the 3×3 network of switches in Figure 4.3 (a) corresponds to the lattice form depicted in Figure 4.3 (b), which implements the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$. If we assign the values 1, 1, 0 to the variables x_1, x_2, x_3, respectively, we obtain paths of gray square connecting the top and the bottom edges of the lattices (Figure 4.3 (c)), indeed on this assignment f evaluates to 1. On the contrary, the assignment $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, on which f evaluates to 0, does not define any path from the top to the bottom edge (Figure 4.3 (d)).

The synthesis problem on a lattice consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size. The size is measured in terms of the number of switches in the lattice.

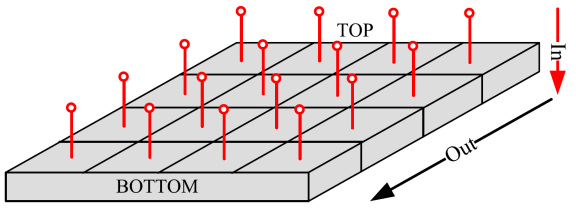


Figure 4.2: 3D draw of signal propagation inside a switching lattice. The input signal is given to each four-terminal switch and the output is given by a path from the top to the bottom of the lattice

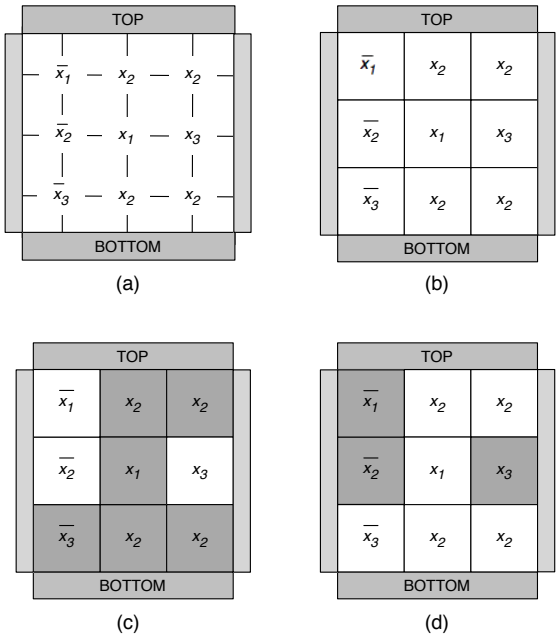


Figure 4.3: A four terminal switching network implementing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ (a); its corresponding lattice form (b); the lattice evaluated on the assignments 1,1,0 (c) and 0, 0, 1 (d), with grey and white squares representing ON and OFF switches, respectively.

4.2 Synthesis methods

Switching lattices are composed by 4-terminal switches, so they require different synthesis methods with respect to CMOS. In This paragraph we show the synthesis methods used to synthesize lattices.

4.2.1 Altun-Riedel

A switching lattice can similarly be equipped with left edge to right edge connectivity, so that a single lattice can implement two different functions. This fact is exploited in [25] where the authors propose a synthesis method for switching lattices simultaneously implementing a function f according to the connectivity between the top and the bottom plates, and its dual function f^D according to the connectivity between the left and the right plates. Recall that the dual of a Boolean function f depending on n binary variables is the function f^D such that $f(x_1, x_2, \ldots, x_n) = \overline{f^D}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$. This method produces lattices with a size that grows linearly with the number of products in an irredundant sum of product (SOP) representation of f, and consists of the following steps:

1. find an irredundant, or a minimal, SOP representation for f and f^D: $SOP(f) = p_1 + p_2 + \cdots + p_s$ and $SOP(f^D) = q_1 + q_2 + \cdots + q_r$;
2. assign each product p_j ($1 \leq j \leq s$) of $SOP(f)$ to a column and each product q_i ($1 \leq i \leq r$) of $SOP(f^D)$ to a row, as shown in Figure 4.4;
3. for all $1 \leq i \leq r$ and all $1 \leq j \leq s$, assign to the switch on the lattice site (i, j) one literal which is shared by q_i and p_j (the fact that f and f^D are duals guarantees that such a shared literal exists for all i and j).

This synthesis algorithm thus produces a lattice for f whose size depends on the number of products in the irredundant SOP representations of f and f^D, and it comes with the dual function implemented for free. For instance, the lattice depicted in Figure 4.3 has been built according to this algorithm, and it implements both the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ and its dual $f^D = x_1\overline{x}_2x_3 + \overline{x}_1x_2 + x_2\overline{x}_3$.

Figure 4.5a shows an example of lattice obtained using this synthesis method.

The time complexity of the algorithm is polynomial in the number of products. However, the method does not always build lattices of minimal size for every target function, since it ties the dimensions of the lattices to the number of products in the SOP forms. In particular this method is not effective for Boolean functions whose duals have a very large number of products. Another reason that could explain the non-minimality of the lattices produced in this way is that the algorithm does not use Boolean constants as input, i.e., each switch in the lattice is always controlled by a Boolean literal.

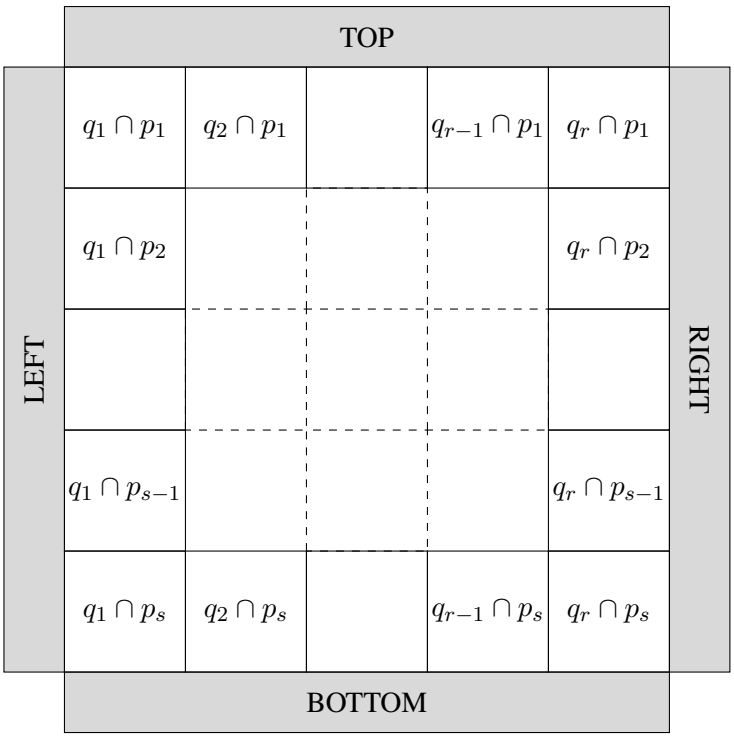
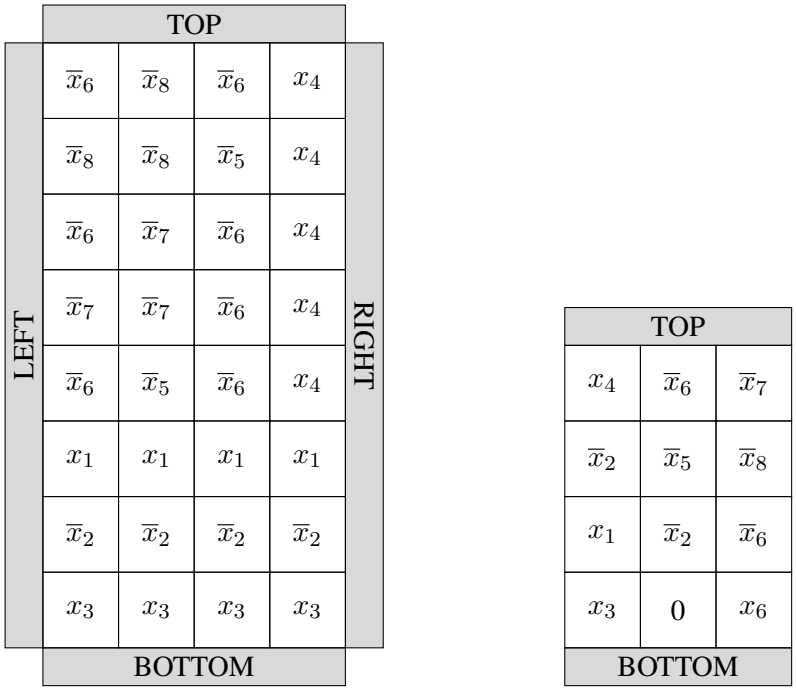


Figure 4.4: Altun-Riedel synthesis method. f is implemented from top to bottom and f^D from left to right.



(a) (b)

Figure 4.5: Lattices obtained synthesizing the Boolean function $f = \overline{x}_8\overline{x}_7\overline{x}_6x_3\overline{x}_2x_1 + \overline{x}_8\overline{x}_7\overline{x}_5x_3\overline{x}_2x_1 + x_4x_3\overline{x}_2x_1$. At right the lattice obtained using Altun-Riedel synthesis method, at left the lattice obtained with Gange-Søndergaard-Stuckey synthesis method.

4.2.2 Gange-Søndergaard-Stuckey

In [26], the authors proposed a different approach to the synthesis of minimal-sized lattices, which is formulated as a satisfiability problem in quantified Boolean logic and solved by quantified Boolean formula solvers. This method uses the Altun-Riedel algorithm to find an upper bound on the dimensions of the lattice. It then searches for successively better implementations until either an optimal solution is found, or else a preset time limit has been exceeded. Experimental results show how this alternative method can decrease lattice sizes considerably. In this approach the use of fixed inputs is allowed, moreover the lattice considers only the top-to-bottom paths and implements the function f, but not its dual.

Figure 4.5b shows an example of lattice obtained using this synthesis method.

Chapter 5

Decomposition Methods

The cost of implementing a four-terminal switching lattice could be mitigated by exploiting Boolean function decomposition techniques. The basic idea of this approach is to first decompose a function into some subfunctions, according to a given functional decomposition scheme, and then to implement the decomposed blocks with physically separated regions in a single lattice. Since the decomposed blocks usually correspond to functions depending on fewer variables and/or with a smaller on-set, their synthesis should be more feasible and should produce lattice implementations of smaller size.

In the framework of switching lattice synthesis, where the available minimization tools are not yet as developed and mature as those available for CMOS technology, we are interested in reducing the size of the function to be minimized with a preprocessing phase. A smaller input function to a minimization algorithm can imply a smaller area circuit and a reduced synthesis time.

5.1 P-circuits and EP-SOP forms

We now review two slightly different bounded-level logic networks called *P-circuits* and *EXOR-Projected Sums of Products forms* (EP-SOP). Both networks are based on generalizations of the standard Shannon decomposition, and can be seen as special logic architectures which realize a Boolean function by projecting it onto overlapping subsets. They were introduced in [72, 73, 74] and in [75, 76, 77], and further studied in [78, 79].

We first give some preliminary definitions.

A *completely specified Boolean function* f is a function $f : \{0,1\}^n \rightarrow \{0,1\}$. A completely specified Boolean function can also be interpreted as the set of points $x \in \{0,1\}^n$ such that $f(x) = 1$.

An *incompletely specified Boolean function* is a function $f : \{0,1\}^n \rightarrow$

$\{0, 1, -\}$, where $-$ is called the don't care value of the function. An incompletely specified function can be described by three sets of points: the *on-set*, the *off-set* and the *don't care set*, which characterize the points in $\{0, 1\}^n$ with images 0, 1, and $-$, respectively.

Given the Boolean space $\{0, 1\}^n$ described by the set $\{x_1, \ldots, x_n\}$ of n binary variables, a *literal* is a variable or its complement; a *cube* is conjunction (or product) of a set of literals, and a *minterm* is a cube when it represents only one point, i.e., when it is a conjunction of n distinct literals. Finally, a *multiple-output Boolean function* f is a function $f : \{0, 1\}^n \rightarrow \{0, 1, -\}^m$; it can be considered also as a vector of Boolean functions $\{f_1, f_2, \ldots, f_m\}$.

P-circuits and EP-SOPs are extended forms of Shannon cofactoring, where the expansion is with respect to an orthogonal basis $\overline{x}_i \oplus p$ (i.e., $x_i = p$), and $x_i \oplus p$ (i.e., $x_i \neq p$), where p is a function defined over all variables except for the critical variable x_i (e.g., the variable with more switching activity or with higher delay that should be projected away from the rest of the circuit).

More precisely, let f be a completely specified Boolean function depending on the set $\{x_1, \ldots, x_n\}$ of n binary variables. Consider the classical Shannon decomposition

$$f = \overline{x}_i f|_{\overline{x}_i} + x_i f|_{x_i} \,,$$

and the more general EXOR-based decomposition [80] [81]

$$f = (\overline{x}_i \oplus p) f|_{x_i = p} + (x_i \oplus p) f|_{x_i \neq p} \,,$$

that corresponds to the classical one when $p = 0$ (as before, p is a function non-depending on x_i). This decomposition partitions the Boolean space $\{0, 1\}^n$ into two subsets: the subset of points where $x_i = p$ and the subset of points where $x_i \neq p$. The characteristic functions of these two subsets are $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively.

Note that these two subsets have always the same cardinality, for any function p non-depending on x_i. This is due to the presence of the EXOR operator in their characteristic functions: indeed, for any minterm $x \in \{0, 1\}^n$, x belongs to the subset where $x_i = p$, i.e., the subset where $(\overline{x}_i \oplus p) = 1$, if and only if the minterm obtained complementing the i-th bit of x belongs to the subset where $x_i \neq p$, i.e., where $(x_i \oplus p) = 1$. The cofactors $f|_{x_i = p}$ and $f|_{x_i \neq p}$ correspond to the projections of f onto the two subsets with $x_i = p$ and $x_i \neq p$, respectively. Observe that this decomposition is suitable for keeping x_i disjoint from the rest of the circuit, but is not oriented to area minimization. In fact, $f|_{x_i = p}$, and $f|_{x_i \neq p}$ do not depend on the variable x_i, but the cubes of f intersecting both subsets $x_i = p$ and $x_i \neq p$ may be split into two smaller subcubes when they are projected onto $f|_{x_i = p}$, and $f|_{x_i \neq p}$, respectively.

P-circuits and EP-SOPs try to overcome this problem in different ways.

5.1.1 P-circuits

The main idea in P-circuits synthesis is to keep unprojected some of the points of the original function. For this purpose, let $I = f|_{x_i=p} \cap f|_{x_i \neq p}$ be the intersection of the two cofactors $f|_{x_i=p}$ and $f|_{x_i \neq p}$. Note that the intersection I contains the cubes whose products do not contain x_i and that cross the two sets. In order to overcome the splitting of these crossing cubes, we could keep I unprojected, and project only the minterms in $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, obtaining the expression

$$f = (\overline{x}_i \oplus p)(f|_{x_i=p} \setminus I) + (x_i \oplus p)(f|_{x_i \neq p} \setminus I) + I\,.$$

Note that p, $f|_{x_i=p} \setminus I$, $f|_{x_i \neq p} \setminus I$ and I do not depend on x_i. However, the points that are in I could be exploited to form bigger cubes in the projected sets. Therefore, if a point is in I and it is useful for a better minimization of the projected parts, it can be kept both in the projection and in the intersection. Moreover, if a point is covered in both the projected sets, it is not necessary to cover it in the intersection. From these observations, we can infer that the projected sub-circuits should cover at least $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, and must be contained in $f|_{x_i=p}$ and $f|_{x_i \neq p}$, respectively. Moreover, the part of the circuit that is not projected should be contained in the intersection I.

In summary, we can define a P-circuit as follows, where $S(f)$ indicates a SOP circuit implementing a Boolean function f.

Definition 5.1.1 ([78]). *A* P-circuit *of a completely specified function f is the circuit $P(f)$ denoted by the expression:*

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^{I})$$

where

1. $(f|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f|_{x_i=p}$
2. $(f|_{x_i \neq p} \setminus I) \subseteq f^{\neq} \subseteq f|_{x_i \neq p}$
3. $\emptyset \subseteq f^{I} \subseteq I$
4. $P(f) = f\,.$

This definition can be easily generalized to incompletely specified Boolean functions $f = \{f^{on}, f^{dc}\}$. For the sake of simplicity, suppose that $f^{on} \cap f^{dc} = \emptyset$; otherwise, following the usual semantics, we consider $f^{on} \setminus f^{dc}$ as the on-set of f. Let I be the intersection of the projections of f onto the two sets $x_i = p$ and $x_i \neq p$:

$$I = (f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}) \cap (f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}).$$

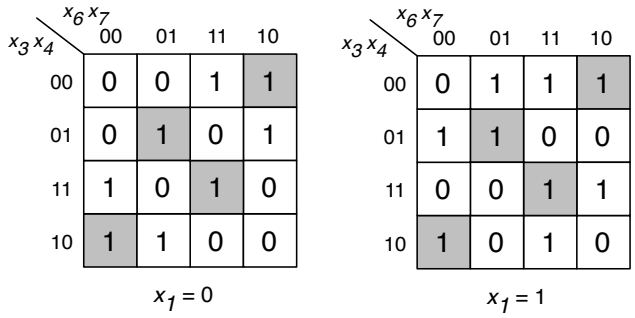


Figure 5.1: Karnaugh map of the benchmark $z = z4(2)$. The cells in grey show the minterms that belong to the intersection between the cofactors $z|_{x_1=0}$ and $z|_{x_1\neq 0}$.

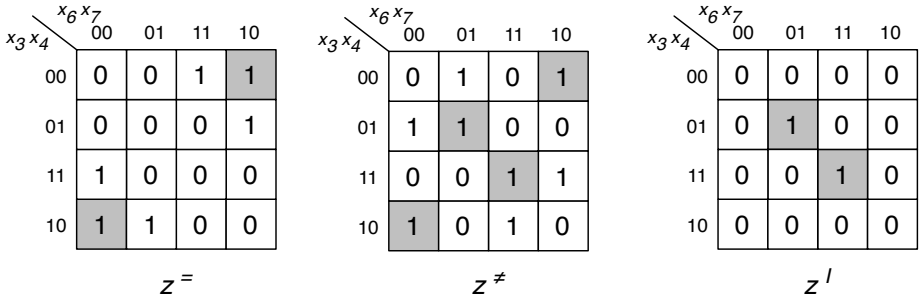


Figure 5.2: Karnaugh maps of the sets $z^{=}$, $z^{\neq}$, and z^I defining an optimal P-circuit for $z = z4(2)$ with respect to x_1 and to the function $p = 0$. The minterms in the intersection between $z|_{x_1=0}$ and $z|_{x_1\neq 0}$, now distributed among $z^{=}$, $z^{\neq}$, and z^I, are highlighted in grey.

Definition 5.1.2 ([78]). *A* P-circuit *of an incompletely specified function* $f = \{f^{on}, f^{dc}\}$ *is the circuit* $P(f)$ *denoted by the expression:*

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$$

where

1. $(f^{on}|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}$
2. $(f^{on}|_{x_i\neq p} \setminus I) \subseteq f^{\neq} \subseteq f^{on}|_{x_i\neq p} \cup f^{dc}|_{x_i\neq p}$
3. $\emptyset \subseteq f^I \subseteq I$
4. $f^{on} \subseteq P(f) \subseteq f^{on} \cup f^{dc}$.

As an example of P-circuit decomposition, let us consider the benchmark $z4$ taken from LGSynth93 [82], and in particular its third output $z4(2)$, simply denoted by z. This function, represented by the Karnaugh map in Figure 5.1, depends

on seven variables $(x_1, x_2, \ldots, x_7)$, two of which (x_2 and x_5) are nonessential. Consider the P-circuit decomposition with respect to the first variable x_1 and to the projection function $p = 0$. The three sets $z^{=}$, $z^{\neq}$, and z^I that define an optimal P-circuit representation of $z = z4(2)$, with respect to x_1 and to the function $p = 0$, are depicted in Figure 5.2. Observe that the two minterms 0010 and 1000 in the intersection between the cofactors $z|_{x_1=0}$ and $z|_{x_1\neq 0}$ have been removed from z^I since they have been kept in both sets $z^{=}$ and $z^{\neq}$ in order to get smaller SOP forms. Moreover, the other two points of the intersection, 0101 and 1111, are kept both in z^I and in the projection $z^{\neq}$, where they are used to form bigger cubes. The P-circuit representation of $z = z4(2)$ is then given by the expression

$$P(z) = \overline{x}_1\, S(z^{=}) + x_1\, S(z^{\neq}) + S(z^I)$$

where $S(z^{=}) = \overline{x}_3\overline{x}_4x_6 + x_3\overline{x}_4\overline{x}_6 + \overline{x}_3x_6\overline{x}_7 + x_3\overline{x}_6\overline{x}_7$, $S(z^{\neq}) = x_3x_4x_6 + \overline{x}_3x_4\overline{x}_6 + x_3x_6x_7 + \overline{x}_3\overline{x}_6x_7 + \overline{x}_3\overline{x}_4x_6\overline{x}_7 + x_3\overline{x}_4\overline{x}_6\overline{x}_7$, and $S(z^I) = x_3x_4x_6x_7 + \overline{x}_3x_4\overline{x}_6x_7$ are the SOP representations of $z^{=}$, $z^{\neq}$, and z^I, respectively.

As we have seen, the idea for synthesis of P-circuits is to construct a network for f by appropriately choosing the sets $f^{=}$, $f^{\neq}$, and f^I as building blocks. Several algorithms for performing such a choice have been studied and proposed in the literature [72, 73, 74, 78]. In particular, in [78], it is shown how the structural flexibility of P-circuits can be completely characterized by using Boolean relations, and how the associated optimal P-circuit decomposition problem, with respect to a given variable x_i and a function p, can be efficiently solved using a Boolean relation minimizer.

5.1.2 EXOR-Projected Sums of Products

Let us now consider the alternative decomposition methods leading to the definition of *EXOR-Projected Sums of Products* (EP-SOPs). These forms are derived as a special case of the generalized Shannon decomposition with *remainder*

$$f = (\overline{x}_i \oplus p)(f|_{x_i=p} \setminus R) + (x_i \oplus p)(f|_{x_i\neq p} \setminus R) + R$$

that restructures a logic function into subsets of points defined by the generalized cofactors with a remainder R containing the cubes that we do not want to split. In particular, EP-SOP forms correspond to the special case $p = x_j$, with $i \neq j$:

$$f = (\overline{x}_i \oplus x_j)(f|_{x_i=x_j} \setminus R) + (x_i \oplus x_j)(f|_{x_i\neq x_j} \setminus R) + R\,.$$

Note that, while the intersection set I in the P-circuit decomposition scheme does not depend on the variable x_i, the remainder R depends in general on all the input variables and is defined as the set of all minterms of f that could form a *crossing*

cube, i.e., a cube of f intersecting both subsets $x_i = x_j$ and $x_i \neq x_j$. In the particular case $p = x_j$, crossing cubes can be precisely identified as those cubes described by products that do not depend on both x_i and x_j; thus the remainder R contains all points x in the on-set of f such that f takes the value 1 on at least one of the two points obtained complementing in x the i-th or the j-th variable. In other words, R contains all cubes described by products of literals

1. that depend on x_i, but not on x_j;
2. that depend on x_j, but not on x_i;
3. that depend neither on x_i, nor on x_j.

If we denote with $x^{(k)}$ the point obtained complementing the k-th bit of x, we get

$$R = \{x \mid f(x) = 1 \wedge (f(x^{(i)}) = 1 \vee f(x^{(j)}) = 1)\}.$$

Moreover, the two cofactors ($f|_{x_i=x_j}$ and $f|_{x_i \neq x_j}$) can be equivalently defined as incompletely specified Boolean functions depending on all input variables, in the following way: 1) $f^{on}|_{x_i=x_j}$ ($f^{on}|_{x_i \neq x_j}$) is the on-set of the original function f such that $x_i = x_j$ (resp. $x_i \neq x_j$); 2) $f^{dc}|_{x_i=x_j}$ ($f^{dc}|_{x_i \neq x_j}$) is the set of points such that $x_i \neq x_j$ (resp. $x_i = x_j$). These don't cares can be inserted in the cofactor $f|_{x_i=x_j}$ since, in the decomposition, this function is multiplied by $(\overline{x}_i \oplus x_j)$, which evaluates to 0 when $x_i \neq x_j$. A symmetric observation holds for $f|_{x_i \neq x_j}$. We will adopt this alternative definition, so that all three functions occurring in the decomposition may depend on all input variables (x_i included).

For example, consider the function f in Figure 5.3(a), $i = 1$ and $j = 2$. In the figure, the subset of the Boolean space where $x_1 = x_2$ ($x_1 \neq x_2$, resp.) is depicted in gray (white, resp.). Figure 5.3(b) shows the remainder for f. Finally, the (non-projected) cofactors $f|_{x_1=x_2}$ and $f|_{x_1 \neq x_2}$ are represented in Figures 5.3(c) and 5.3(d), respectively. Note that $f|_{x_1=x_2}$ corresponds to f for the points where $x_1 = x_2$ and contains don't care conditions where $x_1 \neq x_2$. These don't cares can be inserted in $f|_{x_1=x_2}$ since, in the decomposition, this function is multiplied by $(\overline{x}_1 \oplus x_2)$, which evaluates to 0 when $x_1 \neq x_2$. A symmetric observation holds for $f|_{x_1 \neq x_2}$.

A clear advantage of this representation is that don't care points can be used to form bigger cubes. Consider, for instance, the two distinct cubes $\overline{x}_1\overline{x}_2\overline{x}_3$ and $x_1x_2\overline{x}_3$ in Figure 5.3(a). In the function $f|_{x_1=x_2}$, the corresponding cubes are merged together in a bigger cube, i.e., $\overline{x}_3$, using don't cares, as shown in Figure 5.3(c). Instead, the cube $x_1x_3x_4$ in Figure 5.3(a) is an example of crossing cube: it is split into two minterms: $x_1x_2x_3x_4$ in Figure 5.3(c) and $x_1\overline{x}_2x_3x_4$ in Figure 5.3(d) that are covered by two different cubes (x_1x_4 and x_1x_3, resp.). Observe that, if we keep

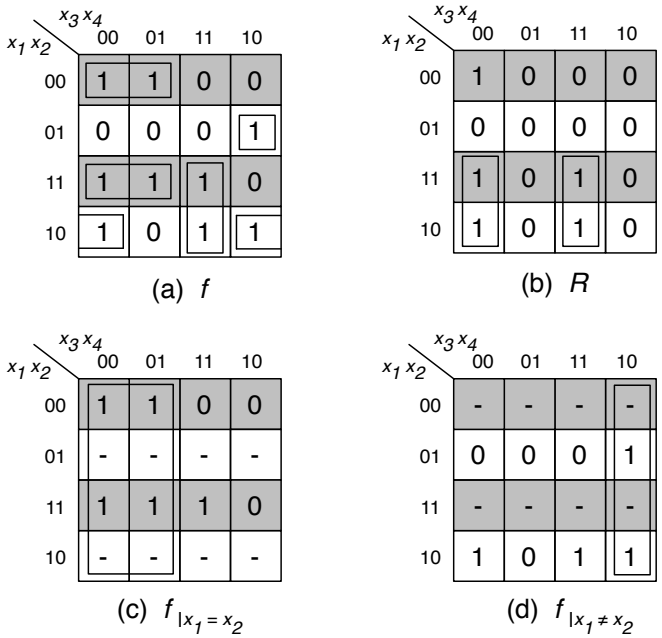


Figure 5.3: (a): A minimal SOP form for the function f. (b), (c) and (d): the remainder R, and the cofactors $f|_{x_1=x_2}$ and $f|_{x_1\neq x_2}$, together with the corresponding covers $f^R = x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4$, $f^{=} = \overline{x}_3$, and $f^{\neq} = x_3\overline{x}_4$, used in a minimal EP-SOP expression for f with respect to the pair of variables x_1 and x_2.

and cover this crossing cube only in the remainder R, then we do not need to cover its minterms in the two cofactors; thus, the cubes x_1x_4 and x_1x_3 will not appear in the final EP-SOP expression.

As for the P-circuit scheme, we can observe that the minterms in the remainder R could be exploited to form bigger cubes in the projected sets, i.e., if a point is in R and is useful for a better minimization of one of the cofactors, it can be kept both in the projected cofactor and in the remainder. Therefore, in order to cover any point of the function at least once and get a minimal decomposition form, with respect to the variables x_i and x_j, we can decide to cover any minterm x in the remainder:

1. only in the remainder,
2. only in the corresponding cofactor (i.e., $f|_{x_i=x_j}$ if $x_i = x_j$ or $f|_{x_i\neq x_j}$ if $x_i \neq x_j$),
3. both in the remainder and in the corresponding cofactor.

The last choice can be convenient when x is useful for forming bigger cubes for both the remainder and the cofactor. For instance, in the running example, the point 1100 can be used in $f|_{x_1=x_2}$ to form the cube $\overline{x}_3$ and in the remainder R to form the cube $x_1\overline{x}_3\overline{x}_4$ with the point 1000.

In summary, we can rephrase the definition of an EP-SOP-circuit given in [76] as follows (as before, $S(f)$ indicates a SOP circuit implementing a Boolean function f).

Definition 5.1.3 ([79])**.** *An* EP-SOP-decomposition*, with respect to the variables x_i and x_j, of a completely specified function f is the expression:*

$$\textit{EP-SOP}(f) = (\overline{x}_i \oplus x_j)\, S(f^{=}) + (x_i \oplus x_j)\, S(f^{\neq}) + S(f^{R})$$

where

1. $(f^{on}|_{x_i=x_j} \setminus R) \subseteq f^{=} \subseteq f^{on}|_{x_i=x_j} \cup f^{dc}|_{x_i=x_j}$
2. $(f^{on}|_{x_i\neq x_j} \setminus R) \subseteq f^{\neq} \subseteq f^{on}|_{x_i\neq x_j} \cup f^{dc}|_{x_i\neq x_j}$
3. $\emptyset \subseteq f^{R} \subseteq R$
4. $\textit{EP-SOP}(f) = f$.

For our running example in Figure 5.3, we get the EP-SOP form EP-SOP$(f) = (\overline{x}_1 \oplus x_2)(\overline{x}_3) + (x_1 \oplus x_2)(x_3\overline{x}_4) + (x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4)$, that contains fewer and bigger cubes than a classical minimal SOP cover $f = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3\overline{x}_4 + x_1x_2\overline{x}_3 + x_1x_3x_4 + x_1\overline{x}_2\overline{x}_4$. Note that any point of the function is covered at least once, by $f^{=}$, $f^{\neq}$, or f^{R}. For example, 0000 is covered by $f^{=}$, 1100 is covered by

both $f^{=}$ and f^{R}, and 1000 is covered by f^{R} (note that 1000 is also covered by $f^{=}$ but not by $(\overline{x}_1 \oplus x_2)f^{=}$ in the final form).

This definition can be generalized to incompletely specified Boolean functions $f = \{f^{on}, f^{dc}\}$. When f is an incompletely specified Boolean function, $f|_{x_i=x_j}$ and $f|_{x_i \neq x_j}$ can be defined as follows: 1) $f^{on}|_{x_i=x_j}$ ($f^{on}|_{x_i \neq x_j}$) contains the points of f^{on} such that $x_i = x_j$ (resp. $x_i \neq x_j$); 2) $f^{dc}|_{x_i=x_j}$ ($f^{dc}|_{x_i \neq x_j}$) contains the points of f^{dc} such that $x_i = x_j$ (resp. $x_i \neq x_j$) together with the points such that $x_i \neq x_j$ (resp. $x_i = x_j$).The definition of the remainder R can be immediately generalized to incompletely specified Boolean functions, noting that the points potentially included in a crossing cube can now be defined as the points x in the on-set or in the dc-set of f, such that the two points obtained complementing in x the i-th and the j-th variable are not both included in the off-set.

Definition 5.1.4 ([79])**.** *An* EP-SOP-decomposition*, with respect to the variables x_i and x_j, of an incompletely specified function $f = \{f^{on}, f^{dc}\}$ is the expression:*

$$\textit{EP-SOP}(f) = (\overline{x}_i \oplus x_j)\, S(f^{=}) + (x_i \oplus x_j)\, S(f^{\neq}) + S(f^{R})$$

where

$$R = \{x \in f^{on} \cup f^{dc} \,|(x^{(i)} \in f^{on} \cup f^{dc}) \vee (x^{(j)} \in f^{on} \cup f^{dc})\}$$

1. $(f^{on}|_{x_i=x_j} \setminus R) \subseteq f^{=} \subseteq f^{on}|_{x_i=x_j} \cup f^{dc}|_{x_i=x_j}$
2. $(f^{on}|_{x_i \neq x_j} \setminus R) \subseteq f^{\neq} \subseteq f^{on}|_{x_i \neq x_j} \cup f^{dc}|_{x_i \neq x_j}$
3. $\emptyset \subseteq f^{R} \subseteq R$
4. $f^{on} \subseteq \textit{EP-SOP}(f) \subseteq f^{on} \cup f^{dc}$.

As for P-circuits, the problem of EP-SOP synthesis can be nicely formalized and efficiently solved using Boolean relations, as discussed and proved in [79].

We finally observe that both decomposition techniques could be recursively applied to the cofactors $f^{=}$, $f^{\neq}$, and f^{I} or f^{R}, but this leads to an increase in the number of logic levels. Thus, this recursive approach could be very interesting for the synthesis of unbounded multilevel forms.

5.2 D-reducible Boolean functions

D-reducible functions are functions whose points are completely contained in an affine space A strictly smaller than the whole Boolean cube $\{0,1\}^n$:

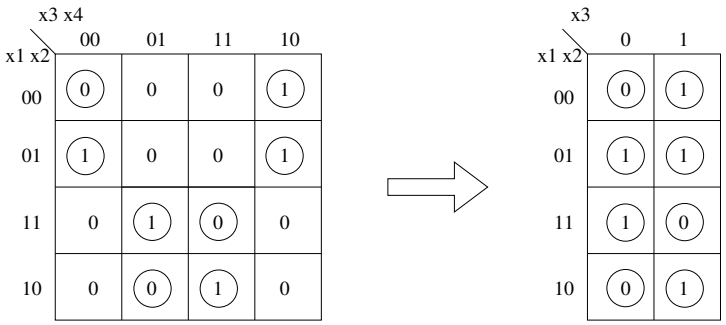


Figure 5.4: Karnaugh maps of a D-reducible function f and its corresponding projection f_A.

Definition 5.2.1. *The Boolean function* $f : \{0,1\}^n \rightarrow \{0,1\}$ *is* D-reducible *if* $f \subseteq A$*, where* $A \subset \{0,1\}^n$ *is an affine space of dimension strictly smaller than* n*.*

Recall that an affine space is a vector space, or the translation of a vector space. Formally, given a vector subspace V of $(\{0,1\}^n, \oplus)$, and a point α in $\{0,1\}^n$, then the set $A = \alpha \oplus V = \{\alpha \oplus v \mid v \in V\}$ is an *affine space* over V with *translation point* α [83].

Let f be a D-reducible function. The minimal affine space A containing f is unique and it is called the *associated affine space* of f. The function f can be represented in the following way: $f = \chi_A \cdot f_A$, where $f_A \subseteq \{0,1\}^{\dim A}$ is the projection of f onto A and χ_A is the characteristic function of A. Moreover, as shown in [84], an affine space can be represented by a simple expression, called *pseudoproduct*, consisting in an AND of EXORs or literals. In particular, an affine space of dimension $\dim A$ can be represented by a pseudoproduct containing $(n - \dim A)$ EXOR factors.

For instance, consider the function $f = \{0010, 0100, 0110, 1011, 1101\}$ in the Karnaugh map on the left side of Figure 5.4. The function f is D-reducible, i.e., we can project it onto a space of dimension three (the space marked with circles in the Karnaugh map). We can therefore study the new function f_A that depends only on three variables, represented in the Karnaugh map on the right side of Figure 5.4. Notice that f and f_A have the same number of points, but these are now compacted in a smaller space. If we synthesize f and f_A in the classical SOP framework we obtain $f = \overline{x}_1 x_3 \overline{x}_4 + \overline{x}_1 x_2 \overline{x}_4 + x_1 \overline{x}_2 x_3 x_4 + x_1 x_2 \overline{x}_3 x_4$, and $f_A = \overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3$. (Note that f depends on all the variables $x_1, \ldots x_4$.) The new and more compact form for f is then $f = (x_1 \oplus \overline{x}_4)(\overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3)$. The EXOR $(x_1 \oplus \overline{x}_4)$ represents the new Boolean space where we study f_A.

The test that establishes whether a function f is D-reducible and the computation of the smallest affine space containing f can be performed in polynomial time, by finding the reduced row echelon form of a matrix derived from any SOP

representation of f (see [85] for more details). Moreover, the projection f_A of f onto A can be simply derived from f by deleting $\dim A$ variables. It is important to note that f_A can be computed starting from any SOP representation of f without generating all its minterms.

5.3 Autosymmetric functions

In this section we briefly review autosymmetric functions that are introduced in [86] and further studied in [87, 88, 89, 90, 91, 92]. For the description of these particular regular functions we need to summarize several concepts of Boolean algebra [93].

Given two binary vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, let $\boldsymbol{\alpha} \oplus \boldsymbol{\beta}$ be the elementwise EXOR between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, for example $11010 \oplus 11000 = 00010$. We recall that $(\{0,1\}^n, \oplus)$ is a vector space, and that a *vector subspace* V is a subset of $\{0,1\}^n$ containing the zero vector $\mathbf{0}$, such that for each $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in V we have that $\boldsymbol{v}_1 \oplus \boldsymbol{v}_2 \in V$. The vector subspace V contains 2^k vectors, where k is the *dimension* of V, and it is generated by a basis B containing k vectors. Indeed B is a minimal set of vectors of V such that each point of V is an EXOR combination of some vectors in B.

Let us consider a completely specified Boolean function $f : \{0,1\}^n \to \{0,1\}$, recalling that f can be described as the set of binary vectors in $\{0,1\}^n$ for which f takes the value 1 (i.e., the ON-set of f). Using this notation we can give the following definition. The function f is *closed under* a vector $\boldsymbol{\alpha} \in \{0,1\}^n$, if for each vector $\boldsymbol{w} \in \{0,1\}^n$, $\boldsymbol{w} \oplus \boldsymbol{\alpha} \in f$ if and only if $\boldsymbol{w} \in f$.

For example, the function $f = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1011, 1101, 1110\}$ is closed under $\boldsymbol{\alpha} = 0011$, as it can be easily verified.

It is easy to observe that any function f is closed under the zero vector $\mathbf{0}$. Moreover, if a function f is closed under two different vectors $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \{0,1\}^n$, it is also closed under $\boldsymbol{\alpha}_1 \oplus \boldsymbol{\alpha}_2$. Therefore, the set $L_f = \{\boldsymbol{\beta}$: f is closed under $\boldsymbol{\beta}\}$ is a vector subspace of $(\{0,1\}^n, \oplus)$. The set L_f is called the *vector space* of f. For instance, the function f of our previous example is closed under the vectors in the vector space $L_f = \{0000, 0011, 0101, 0110\}$.

For an arbitrary function f, the vector space L_f provides the essential information for the definition of the autosymmetry property:

Definition 5.3.1 ([90]). *A completely specified Boolean function f is* k-autosymmetric, *or equivalently f has* autosymmetry degree k, $0 \leq k \leq n$, *if its vector space L_f has dimension k.*

In general, f is *autosymmetric* if its autosymmetry degree is $k \geq 1$. For instance, the function f of our running example is 2-autosymmetric since its vector space L_f has dimension 2.

We now define a special basis, called canonical, to represent L_f. Consider a $2^k \times n$ matrix M whose rows correspond to the points of a vector space V of dimension k, and whose columns correspond to the variables $x_1, x_2, \ldots, x_n$. Let the row indices of M be numbered from 0 to $2^k - 1$. We say that V is in *binary order* if the rows of M are sorted as increasing binary numbers. We have:

Definition 5.3.2 ([90])**.** *Let V be a vector space of dimension k in binary order. The* canonical basis *B_V of V is the set of points corresponding to the rows of M with indices $2^0, 2^1, \ldots, 2^{k-1}$. The variables corresponding to the first 1 from the left of each row of the canonical basis are the* canonical variables *of V, while the other variables are* non-canonical*.*

It can be easily proved that the canonical basis is indeed a vector basis [94]. The canonical variables of L_f are also called canonical variables of f.

Example 5.3.1. *Consider the vector space L_f of the function f of our running example. We can arrange its vectors in a matrix in binary order:*

	x_1	x_2	x_3	x_4
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0

The canonical basis is composed of the vectors in position **1** *and* **2***, that are the vectors* 0011 *and* 0101*. The canonical variables of f are x_2 (corresponding to the first* 1 *in* 0101*) and x_3 (corresponding to the first* 1 *in* 0011*). The remaining variables x_1 and x_4 are non-canonical.*

For a vector $\boldsymbol{\alpha} \in \{0,1\}^n$ and a subset $S \subseteq \{0,1\}^n$, consider the set $\boldsymbol{\alpha} \oplus S = \{\boldsymbol{\alpha} \oplus \boldsymbol{s} \mid \boldsymbol{s} \in S\}$. In a sense, the vector $\boldsymbol{\alpha}$ is used to "translate" the subset S. If the set S is a vector space, then its "translations" are called *affine spaces*:

Definition 5.3.3. *Let V be a vector subspace of $(\{0,1\}^n, \oplus)$. The set $A = \boldsymbol{\alpha} \oplus V$, $\boldsymbol{\alpha} \in \{0,1\}^n$, is an* affine space *over V with* translation point $\boldsymbol{\alpha}$*.*

Note that $\boldsymbol{\alpha} \in A$, because S contains the zero vector $\mathbf{0}$, hence $\boldsymbol{\alpha} = \boldsymbol{\alpha} \oplus \mathbf{0} \in A$. Moreover, any other vector of A could be chosen as translation point $\boldsymbol{\alpha}$, thus generating the same affine space.

There is a simple formula that characterizes the vector space associated to a given affine space A, namely [93]:

$$V = \boldsymbol{\alpha} \oplus A, \text{ with } \boldsymbol{\alpha} \text{ any point in } A.$$

That is, given an affine space A there exists a unique vector space V such that $A = \boldsymbol{\alpha} \oplus V$, where $\boldsymbol{\alpha}$ is any point of A.

As proved in [87], the points of a k-autosymmetric function f can be partitioned into $\ell = |f|/2^k$ disjoint sets, where $|f|$ denotes the number of points of f; all these sets are affine spaces over L_f. I.e., $S = \boldsymbol{\alpha} \oplus L_f$, where S is any such a space and $\boldsymbol{\alpha} \in f$. Thus:

$$f = \bigcup_{i=1}^{\ell} (\boldsymbol{\alpha}^i \oplus L_f)$$

and for each i, j, $i \neq j$, $(\boldsymbol{\alpha}^i \oplus L_f) \cap (\boldsymbol{\alpha}^j \oplus L_f) = \emptyset$. The vectors $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^\ell$ are chosen as all the points of f where all the canonical variables have value 0.

Example 5.3.2. *Consider the function*

$$f = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1011, 1101, 1110\}$$

of our running example. By Example 5.3.1 the canonical variables of f are x_2 and x_3. Thus, if we take the points of f with all canonical variables set to 0, i.e., $\boldsymbol{\alpha}^1 = 0000$, $\boldsymbol{\alpha}^2 = 0001$, and $\boldsymbol{\alpha}^{\,1} = 1000$, we have

$$f = (0000 \oplus L_f) \cup (0001 \oplus L_f) \cup (1000 \oplus L_f),$$

where $L_f = \{0000, 0011, 0101, 0110\}$.

Autosymmetric functions can be reduced to "equivalent, but smaller" functions; in fact, if a function f is k-autosymmetric, then there exists a function f_k over $n-k$ variables, y_1, y_2, ..., y_{n-k}, such that

$$f(x_1, \ldots, x_n) = f_k(y_1, \ldots, y_{n-k}),$$

where each y_i is an EXOR combination of a subset of x_i's. These combinations are denoted $EXOR(X_i)$, where $X_i \subseteq X$, and the equations $y_i = EXOR(X_i)$, $i = 1, \ldots, n-k$, are called *reduction equations*. The function f_k is called a *restriction* of f; indeed f_k is "equivalent" to, but smaller than f, and has $|f|/2^k$ points only.

The restriction f_k can be computed from f and its vector space L_f by first identifying the canonical variables, and then deriving the cofactor of f where all the canonical variables are set to 0 (see [87] and [90] for more details). The reduction equations correspond to the homogeneous system of linear equations whose solutions define the vector space L_f, and they can be derived applying standard linear algebra techniques as shown in [87, 90].

Example 5.3.3. *Consider the 2-autosymmetric function f in our running example, with $L_f = \{0000, 0011, 0101, 0110\}$ and canonical variables x_2 and x_3. We can build f_2 by taking the cofactor $f_{x_2=0,x_3=0} = \{00, 01, 10\}$, that contains only 3 points and corresponds to the function $f_2(y_1, y_2) = \overline{y_1 y_2}$. The homogeneous system whose solutions are $\{0000, 0011, 0101, 0110\}$ is:*

$$\begin{cases} x_1 &= 0 \\ x_2 \oplus x_3 \oplus x_4 &= 0 \end{cases}$$

Thus the reduction equations are given by

$$y_1 = x_1 \tag{5.1}$$

$$y_2 = x_2 \oplus x_3 \oplus x_4 \,. \tag{5.2}$$

Finally, the function f can be represented as:

$$f(x_1, x_2, x_3, x_4) = f_2(y_1, y_2) = \overline{y_1 y_2} = \overline{x_1(x_2 \oplus x_3 \oplus x_4)} \,.$$

We can note that f is indeed a composition of f_2 and the reduction equations (5.1) and (5.2).

5.4 Internal Composition

We recall from [26] that given the switching lattices implementing two functions f and g, we can easily consstruct the lattices representing their disjunction $f + g$ and their conjunction $f \cdot g$ composing the two lattices for f and g and using a padding column of 0s and a padding row of 1s, respectively, as shown in Figure 5.5. In particular, for the disjunction, the column of 0s separates all top-to-bottom paths in the lattices for f and g, so that the accepting paths for the two functions never intersect; this, in turn, implies that there exists a top-to-bottom connected path in the lattice for $f + g$ if and only if there is at least one connected path for f or for g. If the lattices for f and g have a different number of rows, we add some rows of 1s to the lattice with fewer rows, so that each accepting path can reach the bottom edge. Similarly, for the conjunction the padding row of 1s allows to join any top-to-bottom accepting path for the function f with any top-to-bottom accepting path for g, so that the overall lattice evaluates to 1 if and only if both f and g evaluate to 1. As before, if the lattices for f and g have a different number of columns, we add some columns of 0s to the lattice with fewer columns, so that an accepting path for one of the two functions can never reach the opposite edge of the lattice if the other function evaluates to 0.

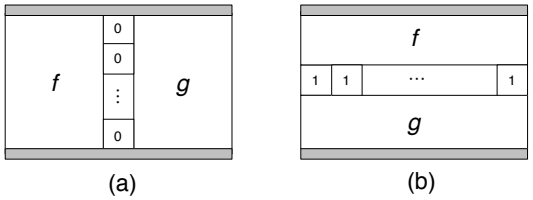


Figure 5.5: Lattice implementation of $f \vee g$ (a) and of $f \wedge g$ (b).

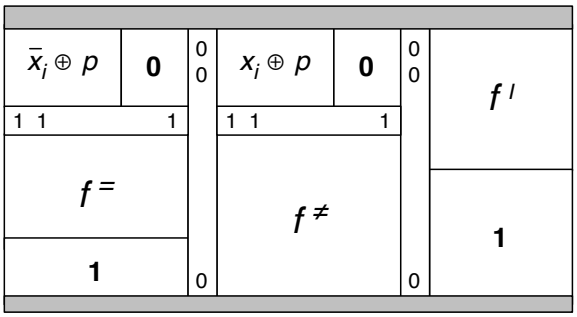


Figure 5.6: Lattice implementation of a P-circuit.

More in general, these simple composition rules can be used to implement a switching lattice for a function f starting from a decomposition of f into subfunctions. The basic idea of this approach is to first decompose f according to a given functional decomposition scheme, then generate the lattices for each component function, and finally implement the original function by a single composed lattice obtained by gluing together appropriately the lattices of the component functions. Since the decomposed blocks usually correspond to functions depending on fewer variables and/or with a smaller on-set, their synthesis should produce lattice implementations of smaller size, yielding an overall lattice of smaller dimension in an affordable computation time. In this section, the lattice for the original function has been obtained implementing the decomposed blocks with physically separated regions in a single overall lattice. We will refer to this approach as *internal composition*.

5.4.1 P-Circuits

We can use these simple composition rules to derive a lattice describing a P-circuit expression $P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$ for a given function f, using lattices for the three sets $f^{=}$, $f^{\neq}$, and f^I and for the projection functions $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$ as building blocks, as depicted in Figure 5.6.

We now formally prove that the lattice implementation of a P-circuit $P(f)$

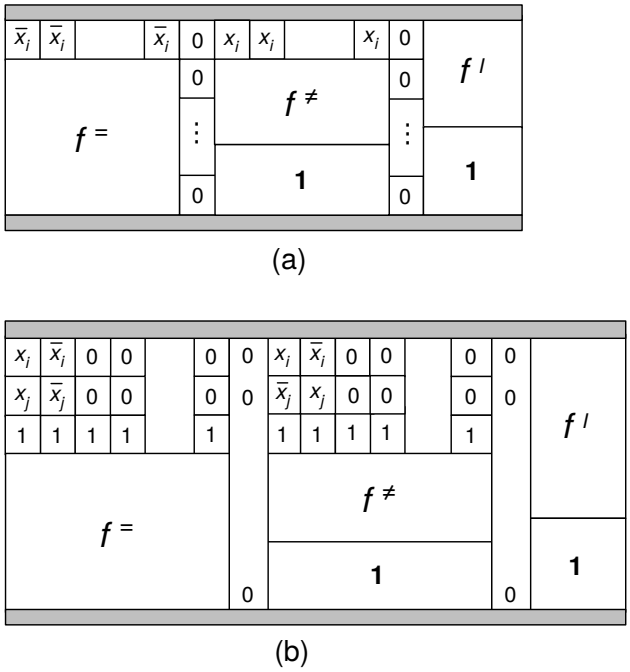


Figure 5.7: Lattice implementation of a P-circuit with projection function $p = 0$ (a) and with projection function $p = x_j$ (b).

described in Figure 5.6 correctly implements the function f.

Theorem 5.4.1. *Let f be a Boolean function depending on n binary variables, and let $P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$ be a P-circuit representing f. The lattice obtained composing the lattices for the three sets $f^{=}$, $f^{\neq}$, and f^I and for the projection functions $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, as shown in Figure 5.6, implements the function f.*

Proof. In order to prove the theorem, we need to show that the function f evaluates to 1 on a given input assignment if and only if there exists a connected path between the top and the bottom edge of the lattice in Figure 5.6.

First suppose that f evaluates to 1 on a given input $(x_1, x_2, \ldots, x_n)$. Then, at least one of the three terms in the expression for $P(f)$ must evaluate to 1. Suppose that the first term $(\overline{x}_i \oplus S(p))\, S(f^{=})$ takes the value 1 on $(x_1, x_2, \ldots, x_n)$. Then, both $(\overline{x}_i \oplus S(p))$ and $S(f^{=})$ are equal to 1, giving rise to a top-to-bottom connected path in the left side of the lattice. An analogous situation arises if the second or the third term are equal to 1.

Now suppose that a given input assignment $(x_1, x_2, \ldots, x_n)$ induces some connected top-to-bottom paths on the lattice. Due to the presence of the two columns of 0s, separating the left, center, and right portions of the lattice corresponding to

the implementations of the functions $(\overline{x}_i \oplus p)\, f^{=}$, $(x_i \oplus p)\, f^{\neq}$, and f^I, respectively, each connected path is entirely contained in one of the three portions. This implies that at least one of the three terms in input to the final OR gate of the P-circuit representing f is equal to 1, and the thesis immediately follows. □

The lattice description of the P-circuit can be simplified depending on the chosen projection function. For instance, if we choose the P-circuit $P(f) = \overline{x}_i\, S(f^{=}) + x_i\, S(f^{\neq}) + S(f^I)$ based on the generalization of the classical Shannon decomposition with projection function $p = 0$, which experimentally represents a very efficient and effective solution, we get the lattice shown in Figure 5.7 (a), where the two padding rows of 1s have been substituted with one row of cells assigned to the literal $\overline{x}_i$ and one row of cells assigned to x_i. Figure 5.7 (b) shows the lattice implementation of the P-circuit $P(f) = (\overline{x}_i \oplus x_j)\, S(f^{=}) + (x_i \oplus x_j)\, S(f^{\neq}) + S(f^I)$ corresponding to the choice $p = x_j$. Both lattices correctly implement the function f, as proved in the following corollary.

Corollary 5.4.1. *The two lattices in Figure 5.7 implement the function f through its P-circuit representations with projection functions $p = 0$ and $p = x_j$, respectively.*

Proof. Let us consider the first lattice, corresponding to the P-circuit representation of f with projection function $p = 0$. As before, observe that each top-to-bottom connected path must be entirely contained in one of the three portions of the lattice, because of the two padding columns of 0s. Moreover, the row of cells assigned to $\overline{x}_i$ on top of the lattice for $f^{=}$ allows to derive a top-to-bottom connected path for f from a connected path for $f^{=}$ if and only if $x_i = 0$; analogously, the row of cell assigned to x_i on top of the lattice for $f^{\neq}$ allows to derive a top-to-bottom connected path for f from a connected path for $f^{\neq}$ if and only if $x_i = 1$. Finally, any connected path for f^I can be extended to a connected path for f. Thus the thesis immediately follows from the definition of P-circuit and of the terms $f^{=}$, $f^{/=}$, and f^I (see Definitions 5.1.1 and 5.1.2).

Now consider the lattice in Figure 5.7 (b), corresponding to the P-circuit for f with projection function $p = x_j$. Any connected path for f^I can be extended to a connected path for f. Moreover, any connected path for $f^{=}$ can be extended to a connected path for f if and only if both variables x_i and x_j assume the same value, i.e., $x_i = x_j$, while any connected path for $f^{\neq}$ can be extended to a connected path for f if and only if the two variables assume different values, i.e., $x_i = \overline{x}_j$. Since the presence of the two columns of 0s prevents the existence of top-to-bottom connected paths intersecting different regions of the lattice, the thesis immediately follows from Definitions 5.1.1 and 5.1.2. □

As an example of synthesis of lattices based on the P-circuit decomposition, let us consider the third output, here denoted by z, of the benchmark $z4$ taken

$\overline{x}_1$	$\overline{x}_1$	$\overline{x}_1$	$\overline{x}_1$	0	x_1	x_1	x_1	x_1	x_1	x_1	0	x_4	x_4
x_6	x_3	x_6	x_3	0	x_6	$\overline{x}_3$	x_6	$\overline{x}_3$	$\overline{x}_3$	$\overline{x}_4$	0	x_7	x_7
$\overline{x}_3$	$\overline{x}_6$	$\overline{x}_3$	$\overline{x}_6$	0	x_3	$\overline{x}_6$	x_3	$\overline{x}_6$	$\overline{x}_4$	$\overline{x}_4$	0	x_6	$\overline{x}_3$
$\overline{x}_4$	$\overline{x}_4$	$\overline{x}_7$	$\overline{x}_7$	0	x_6	$\overline{x}_3$	x_6	$\overline{x}_3$	$\overline{x}_3$	$\overline{x}_7$	0	x_3	$\overline{x}_6$
1	1	1	1	0	x_3	$\overline{x}_6$	x_3	$\overline{x}_6$	$\overline{x}_7$	$\overline{x}_7$	0	1	1
1	1	1	1	0	x_3	x_4	x_3	x_7	x_6	x_3	0	1	1
1	1	1	1	0	x_4	x_4	x_7	x_7	$\overline{x}_3$	$\overline{x}_6$	0	1	1

Figure 5.8: Lattice for the benchmark $z = z4(2)$ based on the P-circuit decomposition. The sublattices for $z^{=}$, $z^{\neq}$, and z^I are highlighted in grey.

from LGSynth93 [82], whose P-circuit decomposition is shown in Figure 5.2 and discussed in Section 5.1.1. We can derive a lattice implementation of z using lattices for the three subfunctions $z^{=}$, $z^{\neq}$, and z^I as building blocks, as depicted in Figure 5.7 (a). Recall from Section 5.1.1 that the SOP representations of the three subfunctions are

$$S(z^{=}) = \overline{x}_3\overline{x}_4x_6 + x_3\overline{x}_4\overline{x}_6 + \overline{x}_3x_6\overline{x}_7 + x_3\overline{x}_6\overline{x}_7\,,$$
$$S(z^{\neq}) = x_3x_4x_6 + \overline{x}_3x_4\overline{x}_6 + x_3x_6x_7 + \overline{x}_3\overline{x}_6x_7 + \overline{x}_3\overline{x}_4x_6\overline{x}_7 + x_3\overline{x}_4\overline{x}_6\overline{x}_7\,,$$
$$S(z^I) = x_3x_4x_6x_7 + \overline{x}_3x_4\overline{x}_6x_7\,,$$

and contain 4, 6, and 2 products, respectively. The SOP expressions for the corresponding dual functions are

$$S(z^{=^D}) = x_3x_6 + \overline{x}_3\overline{x}_6 + \overline{x}_4\overline{x}_7\,,$$
$$S(z^{\neq^D}) = \overline{x}_3\overline{x}_4x_6 + x_3\overline{x}_4\overline{x}_6 + \overline{x}_3x_6\overline{x}_7 + x_3\overline{x}_6\overline{x}_7 + x_3x_4x_6x_7 + \overline{x}_3x_4\overline{x}_6x_7\,,$$
$$S(z^{I^D}) = x_4 + x_7 + \overline{x}_3x_6 + x_3\overline{x}_6\,,$$

with 3, 6, and 4 products, respectively. Using the method described in [25], we can compute the three sublattices for $z^{=}$, $z^{\neq}$, and z^I, whose dimensions are 3×4, 6×6, and 4×2. Finally, composing these three sublattices as shown in Figure 5.8, we get an overall lattice of dimension 7×14 for the benchmark $z = z4(2)$. Note that the synthesis method in [25] applied directly to z, without exploiting its P-circuit decomposition, would produce a lattice of dimension 12×12.

As already observed, the main idea behind this approach is that lattice synthesis of the subfunctions $f^{=}$, $f^{\neq}$, and f^I, which depend on $n-1$ variables instead of n and have a smaller on-set than f, should be an easier task, and should produce lattices of reduced size, so that the overall lattice for f - derived using minimal

lattices for $f^{=}$, $f^{\neq}$, and f^{I} as building blocks - could be smaller than the one derived for f without exploiting its decomposition in P-circuits. This expectation has been confirmed by our experimental results.

5.4.2 EXOR-Projected-Sums Of Products

In a very similar way, we can synthesize on lattice a function f exploiting its EP-SOP decomposition form. The decomposed lattice has the same structure of the lattice shown in Figure 5.7 (b), with the lattice for the intersection f^{I} replaced with a lattice for the remainder f^{R}. Also recall that now the three building blocks $f^{=}$, $f^{\neq}$, and f^{R} depend on n variables, but contain fewer points than the original function f. The lattice obtained in this way correctly implements the function f, as stated in the following theorem, whose proof can be immediately derived from the proofs of Theorem 5.4.1 and Corollary 5.4.1.

Theorem 5.4.2. *Let f be a Boolean function depending on n binary variables, and let EP-SOP$(f) = (\overline{x}_i \oplus x_j)\, S(f^{=}) + (x_i \oplus x_j)\, S(f^{\neq}) + S(f^{R})$ be an EP-SOP form for f. The lattice obtained composing the lattices for $f^{=}$, $f^{\neq}$, f^{R}, and for the projection function $(\overline{x}_i \oplus x_j)$ implements the function f.*

5.4.3 D-Reducible functions

In this section we will discuss how to obtain a lattice for a D-reducible function implementing the characteristic function of the affine space A and the projection f_A with physically separated regions in a single lattice. Recall from Section 5.2 that a D-reducible function f can be written as $f = \chi_A \cdot f_A$, where A is its unique associated affine space, χ_A is the characteristic function of A, and f_A is the projection of f onto A.

Given the two switching lattices implementing χ_A and f_A, we can easily construct the lattices representing their conjunction using a padding row of 1s, as shown in Figure 5.9. Indeed, the row of 1s allows to join any top-to-bottom accepting path for the characteristic function of A with any top-to-bottom accepting path for the projection f_A, so that the overall lattice evaluates to 1 if and only if both χ_A and f_A evaluate to 1. Of course, if the lattices for χ_A and f_A have a different number of columns, we add some columns of 0s to the lattice with fewer columns, so that an accepting path for one of the two functions can never reach the opposite edge of the lattice if the other function evaluates to 0.

Since the two functions f_A and χ_A depend on fewer variables than the original function f, their synthesis should be more feasible and should produce lattice implementations of smaller area. In the framework of switching lattice synthesis, where the available minimization tools are not yet as developed and mature as those

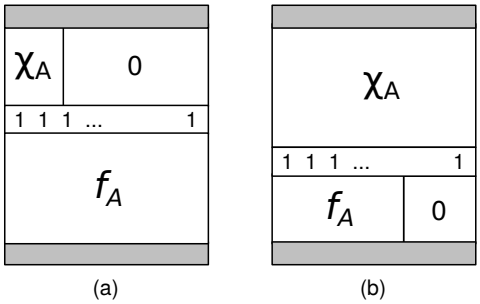


Figure 5.9: Lattice implementations of a D-reducible function $f = \chi_A \cdot f_A$. (a) Composition scheme when the lattice for χ_A has fewer columns. (b) Composition scheme when the lattice for f_A has fewer columns.

available for CMOS technology, reducing the synthesis of a target Boolean function to the synthesis of smaller functions could represent a very beneficial approach.

To further reduce the overall lattice area, we could exploit the peculiar structure of the function χ_A, that represents the minterms of an affine subspace of $\{0,1\}^n$, building its lattice representation block by block. More precisely, we have two possible approaches for the synthesis of a minimal-sized lattice for χ_A: *(i)* directly apply one of the synthesis methods presented in [25] and in [26], or *(ii)* build and compose the lattices representing each EXOR factor, or group of EXOR factors, occurring in χ_A. For this second approach, we now describe a method for implementing a compact lattice representation of affine spaces that can be represented by a 2-*pseudoproduct*, i.e., a product of EXOR of at most two literals. Observe that the class of D-reducible functions whose affine subspace can be described with a 2-pseudoproduct is particularly interesting as EXOR factors are considered technologically feasible if they contain a bounded number of literals, typically 2 [95].

The problem of the minimization of the number of literals in the characteristic function χ_A of the affine space A has been addressed in [96]. The representation of an affine space is not unique, and different pseudoproducts can be characteristic functions of the same affine space. Unfortunately, finding a minimal pseudoproduct, in terms of number of literals, representing an affine space is an NP-hard problem [96], therefore, in the same paper, a greedy heuristic algorithm has been designed. Thus, to avoid the presence of EXOR factors with an unbounded number of literals in the function χ_A, we can first heuristically find an optimal representation of the affine space, and then remove from it all EXOR factors with more than two literals. In this way we obtain the algebraic representation of a new affine subspace A' that contains the original affine space A, and we can still decompose f as $f = \chi_{A'} \cdot f_{A'}$.

In the following analysis we will then restrict our attention to D-reducible

functions decomposed w.r.t. an affine subspace, not necessarily the smallest, represented by the product of single literals and EXOR factors of two literals. The presence of at most two literals in each EXOR factors gives us a simple method for partitioning the variables in order to determine a compact lattice for χ_A. First of all, observe that the pseudoproduct describing A corresponds to a linear system, whose solutions are exactly the minterms in A. For instance, the pseudoproduct $(x_1 \oplus \overline{x}_3) \cdot (x_2 \oplus x_4) \cdot \overline{x}_5 \cdot (x_1 \oplus x_8)$, which describes an affine subspace of $\{0,1\}^8$, is represented by the system

$$\begin{cases} x_1 \oplus \overline{x}_3 &= 1 \\ x_2 \oplus x_4 &= 1 \\ \overline{x}_5 &= 1 \\ x_1 \oplus x_8 &= 1 \end{cases}$$

that can be rewritten as

$$\begin{cases} x_1 &= x_3 \\ x_2 &= \overline{x}_4 \\ x_5 &= 0 \\ x_1 &= \overline{x}_8 \end{cases}$$

From this system we immediately derive the following equalities

$$\begin{array}{lll} x_1 &= x_3 &= \overline{x}_8 \\ x_2 &= \overline{x}_4 & \\ x_5 &= 0\,, & \end{array}$$

that suggest a natural partition of a subset of the input variables:

$$\{\{0, x_5\}, \{x_1, x_3, \overline{x}_8\}, \{x_2, \overline{x}_4\}\}\,,$$

where each subset of the partition contains a set of literals (or the constant 0) that get the same value on A. The input variables missing from the partition (x_6 and x_7 in the example) are the variables that can assume all the possible values on A. In particular, in our example x_5 must be always equal to 0, while x_1, x_3, and $\overline{x}_8$ must have the same value (0 or 1), as well as x_2 and $\overline{x}_4$.

As this example clearly suggests, it is always possible to describe an affine space A, described by an EXOR of at most two literals, through a partition P_A of the input variables, where two variables, possibly complemented, are in the same subset of the partition if and only if they are equal on A. This partition can now be exploited to build the lattice for χ_A.

Theorem 5.4.3. *Let A be an affine subspace of $\{0,1\}^n$ described by the product of single literals and EXOR of two literals, let P_A be the partition of the subset of input variables that defines A, and let $n' \leq n$ be the number of distinct variables*

occurring in P_A. Suppose that P_A contains ℓ subsets of literals, in addition to the subset C with the constant 0. Finally, let c be the number of literals in C. Then A can be implemented with a lattice of area $r \times 2$, where the number r of rows is given by

$$r = \begin{cases} n' & \text{if } c \geq \ell - 1 \\ n' + \ell - 1 - c & \text{if } c < \ell - 1 \end{cases}$$

Proof. Let $S \in P_A$, be one of the ℓ subsets of P_A without the constant 0. The literals in S must be equal to each other on A, thus this subset can be described by the disjunction of two products: the product of all literals in S and the product of the complement of each literal. Thus we can easily build the lattice for S, using two columns representing the two products, that have the same length. Since the two switches on each row are controlled by a variable and its complement, the top-to-bottom accepting paths cannot intersect the two columns, therefore we do not need the padding column of 0 between the two terms of the disjunction. Now, let us consider the set C that contains the c literals that are constant and equal to 0 on A. This set can be implemented with a single column lattice, with a switch assigned to each literal of C. Since this one column lattice must be composed with the previous two column ones, we can extend it with a second column, identical to the first one. Observe that each of the n' literals in P_A occurs in exactly one subset of the partition, and therefore in exactly one row of the lattice.

To compose the sublattices and build the overall lattice representing A, we can exploit the particular structure of the sublattice for C to save padding rows of 1s. Indeed, thanks to the presence of the two identical columns, the two switches on each row of the sublattice for C are controlled by the same Boolean literal. Thus, each single row can be directly inserted between two sublattices representing EXOR factors, as the repeated literal allows to extend any accepting path that reaches the bottom of the first sublattice to the top of the other sublattice, whenever the literal gets the value 0 (i.e., its complement is true). In other words, the sublattice for C is split and each row is inserted between two sublattices for the other subsets of P_A, in order to save padding rows of 1s. Now observe that to join the ℓ sublattices representing the subsets of P_A other than C, we would need $\ell - 1$ padding rows of 1s, that can be all saved if C contains enough literals, i.e., if $c \geq \ell - 1$. In this case, the overall number of rows is given by the number n' of literals occurring in the partition P_A. Otherwise, if $c < \ell - 1$, we must insert $\ell - 1 - c$ padding rows of 1s. □

Applying the construction described in this theorem to our running example $\chi_A = (x_1 \oplus \overline{x}_3) \cdot (x_2 \oplus x_4) \cdot \overline{x}_5 \cdot (x_1 \oplus x_8)$, we get the lattice of size 12 depicted in Figure 5.10 (a). Observe that we do not need the two padding rows of 1s after and before the row whose switches are controlled by the same Boolean literal $\overline{x}_5$,

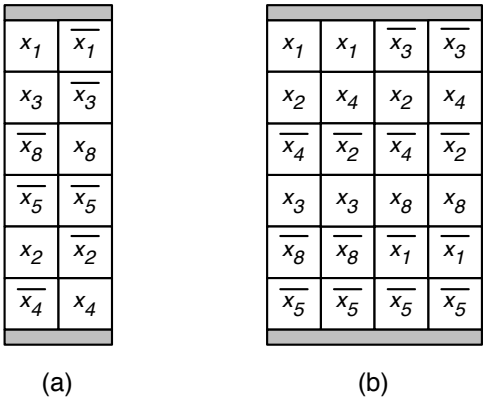


Figure 5.10: Lattice implementations of the function $\chi_A = (x_1 \oplus \overline{x}_3) \cdot (x_2 \oplus x_4) \cdot \overline{x}_5 \cdot (x_1 \oplus x_8)$: (a) lattice derived applying Theorem 5.4.3; (b) lattice synthesized with the algorithm described in [25].

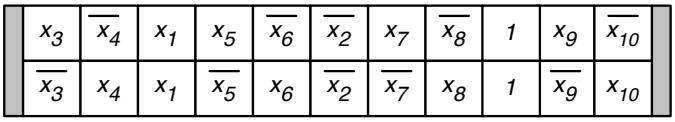


Figure 5.11: Left-to-right lattice implementation of the function $\chi_A = x_1\overline{x}_2(x_3 \oplus x_4) \cdot (x_5 \oplus x_6) \cdot (x_7 \oplus x_8) \cdot (x_9 \oplus x_{10})$.

and that the number of rows is equal to the number of distinct variables occurring in the characteristic function χ_A. Figure 5.10 (b) shows the lattice of size 24 for χ_A, obtained using the synthesis algorithm described in [25]. The method based on SAT, described in [26], synthesizes a lattice of size 12, equivalent to the one obtained applying Theorem 5.4.3.

Now, consider the affine space $\chi_A = x_1\overline{x}_2(x_3 \oplus x_4) \cdot (x_5 \oplus x_6) \cdot (x_7 \oplus x_8) \cdot (x_9 \oplus x_{10})$, corresponding to the partition $P_A = \{\{0, \overline{x}_1, x_2\}, \{x_3, \overline{x}_4\}, \{x_5, \overline{x}_6\}, \{x_7, \overline{x}_8\}, \{x_9, \overline{x}_{10}\}\}$. In this example, $c < \ell - 1$, as $c = 2$ and $\ell = 4$. Thus, the lattice for χ_A, built applying Thereom 5.4.3, contains a padding row of 1s, in addition to the $n' = 10$ rows associated to the literals occurring in P_A, as shown in Figure 5.11 for a lattice with left-to-right connectivity.

With the construction described in Theorem 5.4.3, it is possible to derive lattices more compact than those synthesized with the method presented in [25], with a gain in area that increases with the number ℓ of subsets in the partition associated to the affine space A. Consider for instance an affine space described by exactly ℓ EXOR of two literals, with no literal in common. Applying our method, we can synthesize

a lattice of area $(2\,\ell + \ell - 1) \times 2 = 6\,\ell - 2$, while the algorithm proposed in [25] would synthesize a lattice of area $2^\ell \times 2\,\ell = 2^{\ell+1}\ell$, since the minimal SOP forms for this affine space and for its dual contain 2^ℓ and $2\,\ell$ products, respectively.

In general, the affine subspaces containing EXOR of more than two literals in their characteristic function, have a more complex structure, which cannot be simply described with a partition of the input variables. In this case, we can build a lattice implementation composing the lattices derived for each EXOR factor, or group of EXOR factors, in the characteristic function χ_A of the given affine space A. More precisely, we can implement a lattice representing the product of the single literals and of the EXOR factors of two literals occurring in χ_A applying Theorem 5.4.3, and compose it with the lattice implementations of the other EXOR factors. Moreover, we can use the recursive method developed in [25] for the specific case of the parity function, to implement an EXOR of m literals with a lattice of area $m \times 2^{m-1}$, instead of the general method that would synthesize a lattice of dimension $2^{m-1} \times 2^{m-1}$. Finally, we can use the rows controlled by single literals to join the sublattices of the different EXOR factors, in place of the padding rows of 1s.

5.4.4 Experimental Results

In this section we report the experimental results obtained by applying the decomposition with lattices described in Sections 5.4.1, 5.4.2 and 5.4.3. The experiments have been run on a machine with two AMD Opteron 4274HE for a total of 16 CPUs at 2.5 GHz and 128 GByte of main memory, running Linux CentOS 6.6. The benchmarks (in PLA form) are taken from LGSynth93 [82]. We considered each output as a separate Boolean function. Due to the limited space available, we report in the following only a significant subset of the functions as representative indicators of our experiments.

In order to evaluate the utility of our approach, we compare our results with the ones obtained by the methods presented in [25] and in [26]. We used ESPRESSO to implement the method described in [25], and a collection of Python scripts for computing minimum-area lattices by transformation to a series of SAT problems, to simulate the results reported in [26].

P-circuits and EP-SOP

The algorithms for P-circuit and EP-SOP decomposition have been implemented in C, using the CUDD library for OBDDs [97, 98, 99, 100] to represent Boolean functions, and BREL [101] to solve Boolean relations, as detailed in [78, 79]. For the P-circuit decomposition model, we evaluate and report the results for both projection function $p = 0$, using $x_i = x_0$, i.e. we decompose with respect to

the first input variable of each benchmark, and for $p = x_j$, using $p = x_1$, that is we decompose with respect to the subspace $x_0 \oplus x_1$ described by the first two variables. This choice of variables is arbitrary, as the main objective of this set of experiments is to evaluate how decomposition techniques allow to mitigate the cost of implementing switching lattices. However, we recall here that P-circuit decomposition should be performed with respect to critical signal that should be pushed closer to the outputs, i.e., we should choose as x_i the signal with the highest switching activity (to decrease power consumption), or with higher delay; on the other hand, the choice of the function p should be driven by other considerations, for instance area reduction.

EP-SOP decomposition, originally introduced mainly for area minimization, is performed in this experiments with respect to the pair of variables appearing together most frequently among the products of a minimal SOP for the target function, as suggested in [79].

In Table 5.1 we report dimensions and areas of lattices, in Table 5.2 we report simulation times. Each row of the tables lists the results for any separate output function of the benchmark circuit.

More precisely, in Table 5.1, the first column reports the name and the number of the considered output of each instance. The following columns report dimension ($X \times Y$) and area ($Area = X \cdot Y$) of lattices for each method. In particular, the first group refers to the synthesis of the lattices, as described in [25] (columns 2 and 3) and in [26] (columns 4 and 5) without any decomposition; the second group refers to the synthesis of the lattices, as described in [25] (columns 6 and 7) and in [26] (columns 8 and 9), based on the P-circuit scheme decomposition with $p = 0$; the third group refers to the synthesis of the lattices, as described in [25] (columns 10 and 11) and in [26] (columns 12 and 13), based on the P-circuit scheme decomposition with $p = x_j$; finally, the last group refers to the synthesis of the lattices, as described in [25] (columns 14 and 15) and in [26] (columns 16 and 17) based on the EP-SOP decomposition. For each function, we bolded the best area. We marked with $-$ all cases where the synthesis of a non-decomposed lattice is stopped.

Moreover, in some cases the method proposed in [26] fails in computing a result in reasonable run time. For this reason, we set a time limit (equal to ten minutes) for each SAT execution; if we do not find a solution within the time limit, the synthesis is stopped. In the synthesis of sublattices, whenever [26] is stopped, we use the respective sublattice synthesized with [25], because without a sublattice it would be impossible to complete the synthesis of the overall decomposed lattice. We marked these cases with $\star$. Note that, for many benchmarks, the method in [26] did not find a solution within the fixed time limit for at least one sublattice, and had to be replaced with [25]. We also note that some benchmarks (e.g., *exam(5)* or *mp2d(9)*)

Table 5.1: Proposed lattice sizes for standard benchmark circuits: a comparison of the proposed method with the results presented in [25] and in [26]. All cases where the SAT-based synthesis of a non-decomposed lattice (columns 4 and 5) is stopped after 10 minutes are marked with $-$. Results for decomposed lattices are marked with * when SAT-based synthesis [26] is stopped after 10 minutes and replaced with [25].

	Standard Synthesis				Decomp. with $p = x_0$				Decomp. with $p = x_0 \oplus x_1$				Decomp. EP-SOP			
	[25]		[26]		[25]		[26]		[25]		[26]		[25]		[26]	
	X×Y	A	X×Y	A	X×Y	A	X×Y	A	X×Y	A	X×Y	A	X×Y	A	X×Y	A
adr4(1)	36×36	1296	–	–	37×19	703	37×19	703*	37×21	777	37×21	777*	30×21	630	**25×19**	**475***
alu2(2)	11×10	110	**7×3**	**21**	13×7	91	10×6	60	14×8	112	11×8	88	12×11	132	8×6	48*
alu2(5)	14×13	182	–	–	**16×10**	**160**	**16×10**	**160***	17×10	170	17×10	170*	15×14	210	15×14	210*
alu2(6)	4×4	16	**4×3**	**12**	4×4	16	**4×3**	**12***	4×4	16	4×3	12*	4×7	28	4×6	24*
alu3(0)	**5×4**	**20**	–	–	7×4	28	7×4	28	8×6	48	8×6	48	7×5	35	7×5	35
alu3(1)	8×7	56	**4×3**	**12**	10×5	50	8×5	40	11×7	77	9×7	63	9×8	72	6×6	36*
b12(0)	4×6	24	**3×4**	**12**	4×6	24	**3×4**	**12***	6×7	42	6×7	42*	5×7	35	4×6	24*
b12(1)	7×5	35	**4×4**	**16**	7×5	35	**4×4**	**16***	8×7	56	6×7	42*	8×6	48	5×6	30*
b12(2)	7×6	42	**4×4**	**16**	8×6	48	8×5	40*	11×7	77	11×6	66	10×8	80	10×7	70
bcc(5)	9×27	243	–	–	9×26	234	9×26	234*	10×23	230	10×23	230*	**10×20**	**200**	**10×20**	**200***
bcc(7)	11×31	341	–	–	12×29	348	12×29	348*	12×28	336	12×28	336*	**12×26**	**312**	**12×26**	**312***
bcc(8)	12×31	372	–	–	13×29	377	13×29	377*	13×27	351	13×27	351*	**13×24**	**312**	**13×24**	**312***
bcc(12)	11×31	341	–	–	11×30	330	11×30	330*	12×28	336	12×28	336*	**12×25**	**300**	**12×25**	**300***
bcc(27)	19×38	741	–	–	20×33	660	20×33	660*	20×31	620	20×31	620*	**21×29**	**609**	**21×29**	**609***
bcc(43)	10×20	200	–	–	**11×16**	**176**	**11×16**	**176***	10×22	220	10×22	220*	12×20	240	12×20	240*
bcd.div3(1)	3×4	12	**3×3**	**9**	5×4	20	5×4	20	5×5	25	5×5	25*	4×5	20	4×5	20*
bcd.div3(2)	3×4	12	**3×3**	**9**	5×4	20	5×4	20	7×5	35	7×5	35	4×5	20	4×5	20*
bcd.div3(3)	3×5	15	**3×4**	**12**	4×4	16	4×4	16*	5×5	25	5×5	25*	5×4	20	5×4	20*
bench1(2)	24×45	1080	–	–	33×29	957	33×29	957*	31×27	837	31×27	837*	**31×23**	**682**	**31×22**	**682***
bench1(3)	16×31	496	–	–	20×18	360	21×18	378*	23×22	506	23×22	506*	**21×17**	**357**	**21×17**	**357**
bench1(5)	2×50	1350	–	–	**32×28**	896	32×28	896*	**28×22**	**616**	**28×22**	**616***	31×26	806	31×26	806*
bench1(6)	21×35	735	–	–	**26×24**	**624**	**26×24**	**624***	28×30	840	28×30	840*	26×26	676	26×26	676*
bench1(7)	21×43	903	–	–	**27×20**	**540**	**27×20**	**540***	28×28	784	28×28	784*	29×30	870	29×30	870*
bench1(8)	24×44	1056	–	–	31×26	806	31×26	806*	29×28	812	29×28	812*	26×24	624	25×24	600*
bench(6)	4×8	32	**3×4**	**12**	6×3	18	6×3	18	5×5	25	5×5	25*	8×6	48	8×6	48
br2(4)	**8×18**	**144**	–	–	**8×18**	**144**	**8×18**	**144***	8×20	160	8×20	160*	10×15	150	10×15	150*
br2(5)	**4×14**	**56**	–	–	**4×14**	**56**	**4×14**	**56***	4×16	64	4×16	64*	6×13	78	6×13	78*
br2(6)	**5×16**	**80**	–	–	**5×16**	**80**	**5×16**	**80***	5×18	90	5×18	90*	7×13	91	7×13	91*
clpl(2)	**2×2**	**4**	**2×2**	**4**	3×2	6	3×2	6*	7×5	35	7×5	35	5×4	20	5×4	20*
clpl(3)	6×6	36	**6×3**	**18**	9×6	54	9×6	54*	10×8	80	10×8	80*	10×9	90	10×6	60
clpl(4)	5×5	25	**5×3**	**15**	8×5	40	8×5	40*	9×7	63	9×7	63*	9×8	72	9×6	54
co14(0)	14×92	1288	–	–	15×80	1200	15×80	1200*	15×71	1065	15×71	1065*	**15×67**	**1005**	**15×67**	**1005**
dc1(0)	4×4	16	**3×3**	**9**	5×4	20	4×4	16*	5×6	30	5×6	30*	4×6	24	4×6	24
dc1(1)	**2×3**	**6**	**2×3**	**6**	3×3	9	3×3	9*	5×6	30	5×6	30*	5×6	30	5×6	30
dc1(4)	4×5	20	**3×4**	**12**	5×4	20	5×4	20*	7×6	42	7×6	42	7×6	42	7×6	42
dc1(6)	3×3	9	**3×2**	**6**	4×2	8	4×2	8*	7×5	35	7×5	35	3×6	18	3×5	15
dc2(4)	9×10	90	**4×5**	**20**	10×9	90	8×5	40*	13×12	156	9×7	63*	10×11	110	8×7	56*
dc2(5)	6×6	36	**2×6**	**12**	7×7	49	5×6	30*	10×8	80	6×7	42*	6×9	54	2×9	18*
dk17(0)	2×8	16	**2×6**	**12**	4×4	16	4×4	16*	5×6	30	5×6	30*	4×4	16	4×4	16
dk17(1)	2×8	16	**2×6**	**12**	4×4	16	4×4	16*	5×6	30	5×6	30*	4×4	16	4×4	16
dk17(3)	3×11	33	**2×7**	**14**	4×7	28	6×3	18*	7×7	49	7×6	42*	9×6	54	9×6	54
dk17(4)	3×9	27	**2×7**	**14**	6×4	24	6×4	24	8×6	48	8×6	48	6×7	42	6×6	36
dk27(6)	**1×2**	**2**	**1×2**	**2**	**1×2**	**2**	**1×2**	**2***	2×5	10	2×5	10*	1×3	3	1×3	3*
ex4(4)	**6×17**	**102**	–	–	**6×17**	**102**	**6×17**	**102***	**6×17**	**102**	**6×17**	**102***	6×20	120	6×20	120*
ex4(5)	**45×35**	**1575**	–	–	**45×35**	**1575**	**45×35**	**1575***	**45×35**	**1575**	**45×35**	**1575***	45×38	1710	45×38	1710*
ex5(31)	8×4	32	**6×3**	**18**	10×4	40	10×3	30	11×5	55	11×5	55	9×7	63	7×6	42*
ex5(32)	10×4	40	**6×4**	**24**	13×3	39	13×3	39	13×5	65	13×5	65	13×5	65	13×5	65*
ex5(33)	**7×3**	21	–	–	7×3	21	7×3	21*	11×5	55	11×5	55	8×6	48	8×6	48*
ex5(36)	**8×2**	**16**	**8×2**	**16**	10×2	20	10×2	20	12×4	48	12×4	48	13×4	52	13×4	52
ex5(38)	9×4	36	**6×4**	**24**	13×3	39	13×3	39	13×5	65	13×5	65	13×5	65	13×5	65
ex5(39)	8×2	16	–	–	11×3	33	11×3	33	10×4	40	10×4	40*	11×5	55	11×5	55
ex5(40)	12×6	72	–	–	15×5	75	13×4	52	17×7	119	15×6	90*	13×9	117	13×9	117*
ex5(43)	14×8	112	–	–	17×6	102	13×4	52*	17×8	136	13×6	78*	16×14	224	16×14	224*
exam(5)	6×11	66	–	–	7×6	42	**6×5**	**30***	8×7	56	7×7	49*	7×9	63	8×6	48*
exam(9)	30×59	1770	–	–	38×30	1140	**33×30**	**990***	42×29	1218	42×29	1218*	39×47	1833	39×47	1833*
max128(5)	17×14	238	–	–	19×9	171	14×5	70	20×12	240	13×7	91*	18×9	162	**11×6**	**66***
max128(8)	10×5	50	–	–	11×4	44	**10×4**	**40***	9×6	54	9×6	54*	9×5	45	9×5	45*
max128(17)	25×26	650	–	–	26×15	390	26×15	390*	25×16	400	25×16	400	18×11	198	**11×7**	**77**
mp2d(6)	6×10	60	–	–	6×10	60	**3×7**	**21***	7×11	77	5×10	50*	7×11	77	5×10	50*
mp2d(9)	8×6	48	–	–	9×6	54	**9×4**	**36***	11×7	77	11×7	77*	10×7	70	10×6	60
mp2d(10)	3×6	18	**3×4**	**12**	4×5	20	4×5	20*	3×7	21	2×7	14*	5×7	35	5×7	35*
z4(0)	15×15	225	**4×5**	**20**	16×11	176	6×6	36*	18×6	108	10×6	60	16×16	256	7×7	49*
z4(1)	28×28	784	–	–	30×16	480	**10×7**	**70**	22×16	352	10×8	80	22×19	418	24×23	552*
Z5xp1(2)	11×12	132	–	–	13×7	91	**11×5**	**55***	13×9	117	13×7	91*	13×8	104	11×7	77*
Z5xp1(3)	18×18	324	–	–	19×11	209	**10×6**	**60***	19×12	228	13×7	91*	14×13	182	13×6	78*

Table 5.2: Proposed lattice synthesis times for standard benchmark circuits: a comparison of the proposed method with the results presented in [25] and in [26]. All cases where the SAT-based synthesis of a non-decomposed lattice (column 3) is stopped after 10 minutes are marked with −.

	Standard Synthesis		$p = x_0$		$p = x_0 \oplus x_1$		$EP - SOP$	
	[25]	[26]	[25]	[26]	[25]	[26]	[25]	[26]
	t(s)	t(s)	t(s)	t(s)	t(s)	t(s)	t(s)	t(s)
adr4(1)	**0**	–	1.14	1.14	**0**	**0**	2.87	2.59
alu2(2)	**0**	5420.91	0.95	2.02	**0**	53.012	1.36	1470.75
alu2(5)	**0**	–	2.87	2.87	**0**	**0**	4.63	4.61
alu2(6)	**0**	5420.91	2.98	4.702	**0**	1.734	4.89	6.642
alu3(0)	**0**	–	0.03	0.03	**0**	**0**	0.07	0.07
alu3(1)	**0**	1.749	0.18	0.923	**0**	0.759	0.58	5.7
b12(0)	**0**	2.485	**0**	0.872	**0**	**0**	0.01	0.197
b12(1)	**0**	94.477	0.01	3.292	**0**	1.841	0.02	1271
b12(2)	**0**	309.477	0.11	4.672	**0**	4.505	0.63	5881
bcc(5)	1.09	–	0.39	0.4	**0**	**0**	3.51	3.51
bcc(7)	1.09	–	0.58	0.54	**0.03**	**0.03**	3.63	3.63
bcc(8)	1.09	–	0.66	0.65	**0.03**	**0.03**	1.09	3.7
bcc(12)	1.09	–	0.98	0.97	**0**	**0**	4.13	4.12
bcc(27)	1.09	–	2.35	2.32	**0.03**	**0.03**	12.55	12.36
bcc(43)	1.09	–	11.51	11.62	**0**	**0**	58.65	58.43
bcd.div3(1)	**0**	0.331	0.02	0.03	**0**	**0**	0.05	0.05
bcd.div3(2)	**0**	0.315	0.05	0.06	**0**	**0**	0.09	0.09
bcd.div3(3)	**0**	0.599	0.08	0.09	**0**	**0**	0.11	0.1
bench1(2)	**0.04**	–	13.45	13	0.332	0.03	11.73	11.71
bench1(3)	**0.04**	–	15.73	19.8	**0**	3.72	14.16	83.2
bench1(5)	0.04	–	23.76	23.88	**0**	**0**	20.56	20.58
bench1(6)	0.04	–	27.1	27.25	**0**	**0**	23.74	23.75
bench1(7)	0.04	–	30.21	32.547	**0**	**0**	27.08	27.07
bench1(8)	0.04	–	33.92	34.08	**0**	**0**	30.2	32.097
bench(4)	**0**	1.546	0.41	0.41	**0**	0.115	0.76	1.094
bench(5)	**0**	**0**	0.44	0.44	**0**	**0**	0.83	0.83
bench(6)	**0**	2.668	0.51	0.51	**0**	**0**	1.1	1.11
br2(4)	**0**	–	0.07	0.08	**0**	**0**	0.19	0.19
br2(5)	**0**	–	0.09	0.1	**0**	**0**	0.26	0.25
br2(6)	**0**	–	0.11	0.13	**0**	**0**	0.31	0.3
clpl(2)	**0**	**0**	0.24	0.23	**0**	**0**	0.44	0.42
clpl(3)	**0**	2953	0.96	10.078	**0**	9.127	1.19	308.15
clpl(4)	**0**	52.9	1.46	2.324	**0**	0.86	1.71	10.258
co14(0)	**0**		0.98	0.97	**0**	**0**	0.6	0.6
dc1(0)	**0**	0.413	0.01	0.207	**0**	**0**	0.04	0.04
dc1(1)	**0**	**0**	0.02	0.02	**0**	**0**	0.06	0.06
dc1(4)	**0**	0.585	0.04	0.04	**0**	**0**	0.14	0.14
dc1(6)	**0**	0.198	0.06	0.06	**0**	**0**	0.35	0.17
dc2(4)	**0**	1452.36	0.44	15.284	**0**	117.233	1.42	16.69
dc2(5)	**0**	35.305	0.52	2.424	**0**	5.72	1.54	45.741
dk17(0)	**0**	58.086	0.07	0.07	**0**	**0**	0.05	0.05
dk17(1)	**0**	56.658	0.14	0.14	**0**	**0**	0.1	0.1
dk17(3)	**0**	890.945	0.48	74.169	**0**	0.92	0.58	0.58
dk17(4)	**0**	788.761	0.66	0.67	**0**	**0**	0.68	1166
dk27(6)	**0**	**0**	0.11	0.13	**0**	**0**	0.08	0.08
ex4(4)	0.07	–	2.98	3	**0**	**0**	5.24	5.24
ex4(5)	0.07	–	8.34	8.36	**0**	**0**	10.69	10.68
ex5(31)	0.02	14.408	0.41	1.967	**0**	1.544	0.41	43.511
ex5(32)	0.02	2192.91	0.94	0.91	**0**	**0**	0.92	0.91
ex5(33)	0.02	–	1.23	1.21	**0**	**0**	1.24	1.23
ex5(36)	0.02	0.02	2.38	2.39	**0**	**0**	2.73	2.72
ex5(38)	0.02	1027.5	3.38	3.39	**0**	**0**	3.84	3.83
ex5(39)	0.02	–	3.83	3.84	**0**	**0**	4.32	4.31
ex5(40)	0.02	–	4.6	7.348	**0**	252.83	5.11	5.1
ex5(43)	0.02	–	6.18	96.164	**0**	819.055	6.96	6.94
exam(5)	0.03	–	4.89	40.155	**0**	67.401	4.94	16.151
exam(9)	0.03	–	11.19	7113.73	**0**	**0**	10.84	11.642
max128(5)	**0**	–	2.24	5266.51	**0**	10602.8	2.27	400.61
max128(8)	**0**	–	2.84	7.372	**0**	**0**	3.19	3.22
max128(17)	**0**	–	8	9.503	**0**	**0**	8.63	876.63
mp2d(6)	**0**	–	2.18	3164.15	**0**	433.23	2.62	424.213
mp2d(9)	**0**	–	2.37	2.847	**0**	**0**	2.99	3.05
mp2d(10)	**0**	1.704	2.37	2.4	**0**	0.327	3.01	3.279
z4(0)	**0**	1888.26	0.89	7.31	**0**	5.358	0.97	274.141
z4(1)	**0**	–	2.51	7329.12	**0**	6816.19	2.22	10.065
Z5xp1(2)	**0**	–	0.55	46.701	**0**	200.352	0.72	8.902
Z5xp1(3)	**0**	–	0.58	2279.54	**0**	9393.85	1.79	195.769

show an overall area reduction with respect to the decomposed lattice synthesized only with [25], even if only one or two sublattices are synthesized with [26].

The first column of Table 5.2 reports the name and the number of the considered output of each instance. The following columns report the execution times for each considered method. The time values include the time for the decomposition plus the time for the synthesis of lattices used to compose the final lattice. For each function, we bolded the best execution time. Note that the execution times for the method in [25] are often equal to zero. To synthesize the lattices we used ESPRESSO that has a time granularity of $0.01\,\mathrm{s}$; smaller synthesis-times are indicated as zero.

In Table 5.3 we summarize the improvements of synthesis with decomposition vs. synthesis without decomposition. Every line of the table shows a different type of decomposition. The first column shows the considered decomposition scheme. The second and third columns report the percentages of lattices with smaller area with respect to the results reported in [25] and in [26], respectively. Columns four and five report the average area gain of the lattices that have a smaller area after the decomposition. Columns six and seven show the percentages of lattices taking less time for synthesis (not necessary with a more compact area); finally the last two columns report the average gain of execution time. The negative values of time gain in column eight are due to the poor granularity of ESPRESSO time. The synthesis in [25] is performed using ESPRESSO, and in many cases it takes less than 0.01 s. There are negative values because the time for decomposition is usually less than few seconds, and therefore it has a negligible impact on total synthesis time with respect to area gain.

By comparing the two projection function $p = 0$ and $p = x_j$ for the P-circuit decomposition model, we observe that we obtain better area results considering the $p = 0$ choice, at the expense of a limited increase in the run time needed to decompose the input functions.

By comparing our results with the ones obtained in [26], we obtain a higher percentage of benchmarks with a smaller area by applying the EP-SOP circuit decomposition scheme (35% of smaller area, in contrast to 15% and 13% obtained with the P-circuit scheme), with an average gain of about 33%. The difference between P-circuit-based and EP-SOP-based synthesis results is only partially explained by the fact that two decomposition methods differ in the way they handle crossing cubes. Most likely, this difference is a consequence of the fact that the two decompositions are performed with respect to different input variables. In particular, EP-SOP decomposition is performed with respect to the pair of variables appearing together most frequently among the products of a minimal SOP for the target benchmark, while for P-circuits we used the first two variables x_0 and x_1.

Table 5.3: Results of the decomposition. The values indicate the improvement with respect to the standard synthesis methods. When [26] does not complete, it is compared with [25].

	Smaller area		Average area gain		Less time		Average time gain	
	[25]	[26]	[25]	[26]	[25]	[26]	[25]	[26]
$p = x_0$	34%	15%	25%	30%	6%	25%	-2900%	53%
$p = x_0 \oplus x_1$	27%	13%	22%	27%	53%	50%	98%	57%
$EP - SOP$	27%	35%	28%	33%	2%	3%	-4900%	90%

D-Reducible functions

EXOR factors are considered technologically feasible if they contain a bounded number of literals, typically 2 [95]. For this reason, in our experiments we have considered only D-reducible functions decomposed with respect to affine subspaces, not necessarily the smallest, represented by the product of single literals and EXOR factors of two literals. In the following, we will refer to these functions as *2D-reducible functions*.

The lattice implementation of the characteristic functions of the affine subspaces have been derived applying Theorem 5.4.3, as they are described by products of EXOR factors of at most two literals.

In Table 5.4 for each benchmark we compare the area of the lattice for the plain benchmark with the area of the lattice built applying the decomposition based on the D-reducibility property (see Figure 5.9). In more detail, the first column reports the name of the benchmarks. The following four columns report the dimensions (X,Y), the area ($Area$) and the synthesis time ($Time$, in seconds) of plain lattices. The other columns report the dimensions of the lattices obtained applying the decomposition scheme. In particular we report the dimensions of the overall lattice and the synthesis time (X_{tot}, Y_{tot}, $Area_{tot}$, $Time$), the dimension of the lattice implementation of the affine spaces (X_χ, Y_χ, $Area_\chi$), and the dimension of the projection of the benchmark on the affine space (X_Z, Y_Z, $Area_Z$). In column eight we have bolded the values where we obtain a more compact area. In the last row we report the sum of the corresponding columns.

Results demonstrate that the lattice synthesis of 2D-reducible Boolean functions allows to obtain a more compact area in 15% of the considered cases, with an average gain of about 24%. In particular, considering only the subset of functions whose affine subspace description contains at least one EXOR of two literals (i.e., not only single literals), we obtain a more compact area in about 11% of the cases, with an average gain of about 40%. Moreover, our results show that for 2D-reducible functions we can reduce the synthesis time of the lattices of about 50%, with respect to the time needed for the synthesis of plain lattices. This is due to the fact that the proposed approach is based on a polynomial-time computation of the lattice

Table 5.4: Lattice sizes for 2D-reducible benchmark circuits: a comparison of plain and decomposed lattices.

	[25]				**2D-Red**									
Benchmark	X	Y	$Area$	$Time$	X_{tot}	Y_{tot}	$Area_{tot}$	$Time$	X_χ	Y_χ	$Area_\chi$	X_Z	Y_Z	$Area_Z$
addm4(0)	9	12	108	0.32	3	8	**24**	27.4	1	3	3	3	5	15
addm4(1)	22	23	506	87.14	22	23	506	24.42	1	1	1	22	22	484
addm4(2)	33	36	1188	92.85	33	36	1188	1089	1	1	1	33	35	1155
amd(3)	4	5	**20**	446.04	3	7	21	36.79	1	1	1	3	6	18
amd(4)	10	14	140	723.56	10	14	140	55.98	1	1	1	10	13	130
amd(5)	6	2	**12**	37.72	2	8	16	1	2	6	12	2	2	4
amd(6)	6	3	**18**	1686.65	3	8	24	0.34	1	5	5	3	3	9
amd(7)	5	5	20	88.57	3	6	**18**	4.13	1	1	1	3	5	15
exam(4)	9	25	225	78.26	9	20	**180**	1286.98	1	2	2	9	18	162
exp(6)	5	4	**20**	1118.05	3	7	21	16.1	1	2	2	3	5	15
exp(10)	6	12	72	3361.18	6	5	**30**	294	1	2	2	5	4	20
exp(11)	6	12	72	2058.63	5	8	**40**	248	2	4	8	5	3	15
gary(2)	12	14	**168**	0.03	12	15	180	152.15	2	2	4	12	12	144
gary(3)	5	12	60	253.35	5	12	60	76.3	1	2	2	5	10	50
in2(6)	39	36	1404	0.1	39	35	**1365**	0.1	1	2	2	39	33	1287
in2(7)	17	26	442	0.03	17	26	442	0.06	1	1	1	17	25	425
in2(8)	27	31	837	0.06	27	31	837	0.08	1	1	1	27	30	810
in2(9)	40	36	1440	0.11	40	36	1440	0.11	1	1	1	40	35	1400
in5(6)	5	4	**20**	1129	3	7	21	1.03	1	3	3	4	4	16
m2(5)	8	9	72	1222.7	4	6	**24**	975	1	1	1	4	5	20
t1(0)	6	9	54	1409.76	3	8	**24**	1416.78	1	1	1	3	7	21
t1(1)	7	9	63	236.07	7	9	63	831.62	1	1	1	7	8	56
t1(3)	3	4	**12**	0.97	3	5	15	0.33	1	1	1	3	4	12
t1(4)	3	3	**9**	0.31	3	4	12	0.04	1	1	1	3	3	9
t1(5)	5	3	15	12.77	3	5	15	10.24	1	1	1	3	4	12
			6997	14044.23			6706	6458						

implementation of the affine spaces, and this is useful to reduce the dimension of the considered problem, allowing to reduce the overall time needed to compute the solution.

5.5 External Composition

There are situations where internal composition cannot be directly applied. For instance, consider a function f depending on n binary variables defined as

$$f(x_1, \ldots, x_n) = g(y_1, \ldots, y_k)\,,$$

where *(i)* g is a Boolean function depending on $k < n$ variables; *(ii)* for any i, $y_i = h_i(S_i)$, $S_i \subseteq \{x_1, \ldots, x_n\}$, and h_i is a Boolean function depending on $|S_i|$ variables. Ideally, we would like to derive a lattice implementation for f substituting in a lattice implementation for g each occurrence of a variable y_j with a lattice implementation of the corresponding function h_j. This task, however, requires some care to be performed correctly.

Consider a very simple case: $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2)(x_3 \oplus x_4)$, that can be seen as a functional composition $f = g(y_1, y_2)$ where g is simply a conjunction of two variables, and y_1 and y_2 are EXORs of two variables. Then, we can build a lattice for f (Figure 5.12(c)) starting from the very simple 2×1 lattice representation of g (Figure 5.12(a)), and substituting y_1 and y_2 with the lattice representations of $(x_1 \oplus x_2)$ and $(x_3 \oplus x_4)$, which are shown in Figure 5.12(b). Note that we need to insert a row of 1s between the two sublattices, so that we can extend any accepting path in the sublattice on top, with any accepting path in the bottom sublattice. The overall lattice for f has size 5×2. Notice that using the lattice synthesis method presented in [25] directly on f, we would get a lattice of size 4×4.

Now, consider the function $f(x_1, x_2, x_3, x_4, x_5) = (x_1 \oplus x_2)x_3 + (x_2 \oplus x_5)x_4$. Given a lattice for the function $g = y_1y_3 + y_2y_4$, we could try to build a lattice for f by simply substituting the occurrences of y_1 and y_2 with sublattices for $(x_1 \oplus x_2)$ and $(x_2 \oplus x_5)$, and the occurrences of y_3 and y_4 with x_3 and x_4, respectively. Note that we need to duplicate some variables in order to get a rectangular lattice, besides inserting a padding column of 0, as shown in Figure 5.13. Indeed, without the padding column, the lattice would contain a top-to-bottom path on the assignment 11100, whereas $f(1, 1, 1, 0, 0) = 0$. As a final example, let us consider the parity function of 4 variables

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4\,,$$

that can be interpreted as $f = g(y_1, y_2) = y_1 \oplus y_2$, where $y_1 = x_1 \oplus x_2$ and $y_2 = x_3 \oplus x_4$. If we derive a lattice for f using this decomposition, we need to

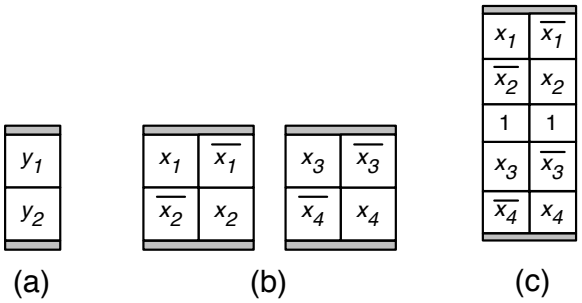


Figure 5.12: (a) Lattice implementation of $g = y_1y_2$; (b) lattices for $y_1 = x_1 \oplus x_2$ and $y_2 = x_3 \oplus x_4$; (c) final lattice for $f = (x_1 \oplus x_2)(x_3 \oplus x_4)$

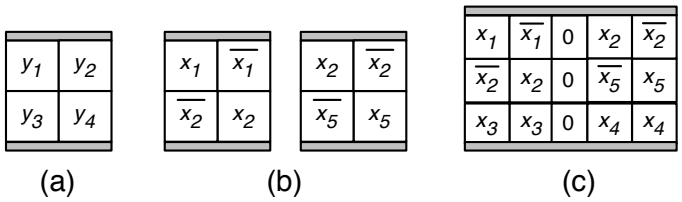


Figure 5.13: (a) Lattice implementation of $g = y_1y_3 + y_2y_4$; (b) lattices for $y_1 = x_1 \oplus x_2$ and $y_2 = x_2 \oplus x_5$; (c) final lattice for $f(x_1, x_2, x_3, x_4, x_5) = (x_1 \oplus x_2)x_3 + (x_2 \oplus x_5)x_4$.

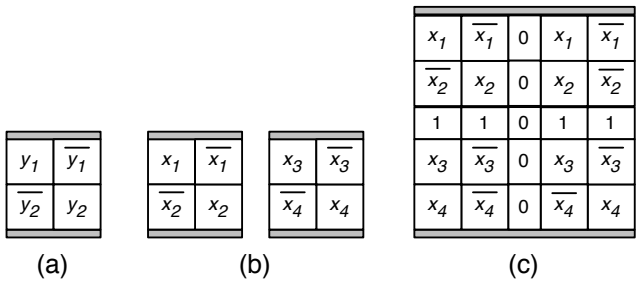


Figure 5.14: (a) Lattice implementation of $g = y_1 \oplus y_2$; (b) lattices for $y_1 = x_1 \oplus x_2$ and $y_2 = x_3 \oplus x_4$; (c) final lattice for $f(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

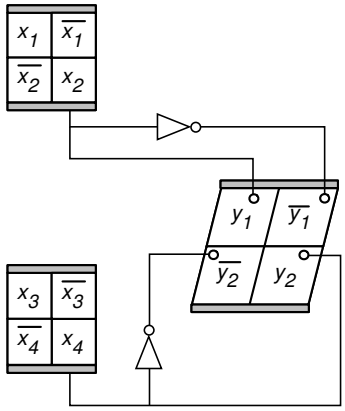


Figure 5.15: Multiple lattice implementation of $f(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

appropriately insert padding rows and columns as depicted in Figure 5.14: the padding rows of 1s are needed to join the accepting paths in the sublattices on top, implementing y_1 and $\overline{y}_1$ with the accepting paths in the bottom sublattices for y_2 and $\overline{y}_2$; while the column of 0s is needed to avoid intersections between accepting paths on the left and on the right side of the overall lattices. Without the column of 0s, the lattice in Figure 5.14 would contain a top-to-bottom path e.g., on the input assignment 0110. With the padding rows and columns, the size of the overall lattice becomes 5×5, that is not competitive with the size of an optimal lattice for the parity of 4 variables, which is 3×5 [102].

A possible strategy to overcome some of these problems could be a different lattice composition technique, that we could call *external composition*. The idea is simply to use multiple nanoarrays, i.e., multiple lattices and to connect the output of a lattice with one or more literals occurring in another lattice as depicted in Figure 5.15. Observe that the overall lattice composition in this picture implements the parity function of 4 variables as a 2×2 lattice representing $g = y_1 \oplus y_2$, connected to two external lattice implementations for $y_1 = x_1 \oplus x_2$ and $y_2 = x_3 \oplus x_4$. In this way, we get a multiple lattice implementation of overall size 12, smaller than an optimal standard lattice for the parity of four variables, whose size is 15 [102]. As this simple example clearly shows, multiple lattices allow to reduce the number of switches and thus the overall dimension of the lattice. However, the gain in the dimension comes at the expense of an increase in the interconnection cost.

5.5.1 Autosymmetric functions

The lattice implementation of autosymmetric functions can be derived exploiting the external lattice composition discussed in the previous Session 5.5. Recall from

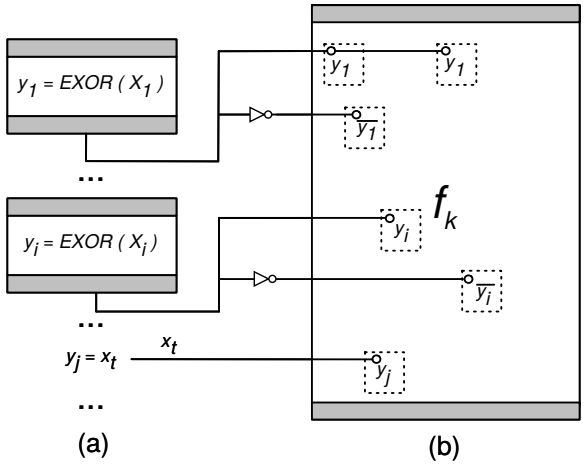


Figure 5.16: Multiple lattice implementation of an autosymmetric function: (a) lattice implementation of the reduction equations; (b) lattice implementation of the restriction f_k.

Section 5.3, that a k-autosymmetric function f can be decomposed as

$$f(x_1, \ldots, x_n) = f_k(y_1, \ldots, y_{n-k}) \,,$$

where *(i)* the *restriction* f_k depends on $n-k$ binary variables, and has $|f|/2^k$ points only; and *(ii)* each y_i is defined by a *reduction equation*, i.e., an EXOR of a subset of the original variables x_is: $y_i = EXOR(X_i)$, $X_i \subseteq \{x_1, \ldots, x_n\}$. Therefore, we can build a multiple lattice implementing f composing a lattice $L(f_k)$ for the restriction f_k with $n-k$ sublattices representing the reduction equations: for each i, $1 \leq i \leq n-k$, the output of the sublattice $L(y_i)$ implementing y_i is connected, possible through an inverter, to all occurrences of the literal y_i in $L(f_k)$ (see Figure 5.16). Of course, for all variables y_j whose associated reduction equation is a single variable, e.g., x_t, there is no need to connect the switch to an external lattice, but just to x_t.

Since f_k depends on fewer variables, and has a smaller on-set with respect to f, its lattice synthesis should be an easier task, and should produce a lattice of reduced size.

5.5.2 P-Circuits

In a very similar way, we can use external lattice composition in the lattice implementation of *P-circuits*.

As before, the lattice obtained by internal composition of the sublattices for $f^{=}$, $f^{\neq}$, and f^{I}, contains padding rows and columns of 1s and 0s, that could be in part avoided applying the external decomposition scheme depicted in Figure 5.17

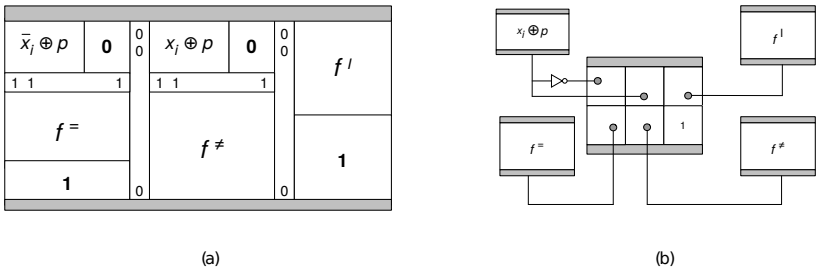


Figure 5.17: Lattice implementation of P-circuits: (a) standard lattice implementation based on internal composition; (b) multiple lattice implementation.

(b). The main lattice is composed of three columns of two switches. The first two columns contains switches connected to the output of external lattices implementing the projection function $(x_i \oplus p)$ and the cofactors $f^=$ and $f^{\neq}$; the last column contains one switch connected to the output of the external lattice implementing f^I, and one switch with constant value 1 used to connect the accepting path in the lattice for f^I to the bottom edge of the main lattice. Observe that the lattice correctly computes the disjunctions between f^I and the projections of $f^=$ and $f^{\neq}$, without the need for padding columns of 0s. This is due to the fact that the lattice contains only two rows, and that each accepting path must contain at least two switches in the same column, as switches are not connected diagonally. Also note that the switch with constant value 1 can be replaced with a second switch connected to f^I, at the expense of the interconnection cost, so that the lattice does not contain switches labelled with constant values.

The overall lattice is therefore composed by a main lattice of size 6, four external sublattices, and one inverter, with an evident gain in the overall area.

5.5.3 D-Reducible functions

In paragraph 5.4.4 we use D-reducible functions to independently find lattice implementations for the projection f_A and for the characteristic function χ_A of A, and then to compose them in a single lattice in order to construct the lattice for f, as shown in Figure 5.18 (a).

From Figure 5.18 (a) we can note that the overall lattice contains the padding row of 1s, used to join the accepting paths of χ_A and f_A, and some extra columns of 0s added to the lattice with fewer columns so that the final lattice has a rectangular shape and the accepting paths for one of the two functions never reach the opposite edge of the lattice if the other function evaluates to 0. This extra row and columns could be avoided using the external composition scheme as depicted in Figure 5.18 (b). The main lattice now consists in only one column of two switches, one switch

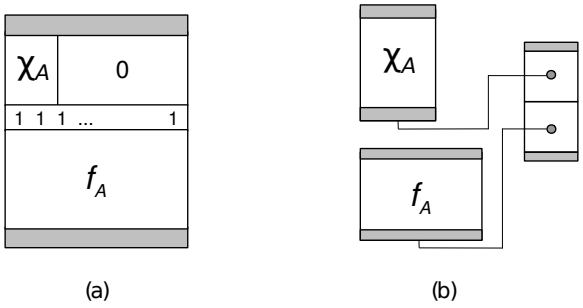


Figure 5.18: Lattice implementation of a D-reducible function: (a) standard lattice implementation based on internal composition; (b) multiple lattice implementation.

is connected to the output of an external lattice implementing χ_A and the other switch is connected to the output of an external lattice for f_A. Since the main lattice consists in just two switches, we do not need any padding switch between them, and the lattice evaluates to 1 if and only if both external sublattices evaluate to 1. The size of the overall lattice is therefore given by 2 plus the sum of the sizes of the two sublattices, and it is smaller than the size of the lattice in Figure 5.18 (a) obtained by standard, internal composition. Only in a few cases, e.g., when the two sublattices for χ_A and f_A have the same number of columns and can be joined without the row of padding 1s, the internal composition produces a lattice of smaller area, but just for an additive factor 2.

5.6 Experimental results

In this section we report the experimental results obtained applying the multiple lattice implementation of autosymmetric functions described in Section 5.5.1. Since a k-autosymmetric function $f_k(y_1, \ldots, y_{n-k})$ depends on fewer variables w.r.t. the corresponding original function $f(x_1, \ldots, x_n)$, our aim is to obtain lattices of reduced size.

The algorithms have been implemented in C, using the CUDD library for OBDDs [100, 99, 97, 98] to represent Boolean functions, and BREL [101] to solve Boolean relations, as detailed in [79, 78]. The experiments have been run on a machine with two AMD Opteron 4274HE for a total of 16 CPUs at $2.5\,\mathrm{GHz}$ and $128\,\mathrm{GByte}$ of main memory, running Linux CentOS 6.6. The benchmarks are taken from LGSynth93 [82]. We considered each output as a separate Boolean function, for a total of 607 functions, including 53 autosymmetric functions on which we applied the lattice implementation described in the previous sections.

To evaluate the utility of our approach, in Table 5.5 we compare the lattice synthesis results obtained applying the decomposition scheme based on autosymmetry,

Table 5.5: Proposed lattice sizes for autosymmetric benchmark circuits: a comparison of the proposed method with the results presented in [25] and in [26]. When the synthesis of a lattice is stopped, there is no lattice (−). Results are marked with * when SAT is stopped.

		[25]					[26]								
	$\#_{y_i}$	Std. Synth.	Decomposed Synthesis				Std. Synth.		Decomposed Synthesis						
		Area	f_k Area	XOR Area	tot. Area	inv.	Area	time	f_k Area	f_k Time	XOR Area	XOR Time	tot. Area	tot. time	inv.
add6(0)	1	**4**	1	4	5	0	**4**	**0.021**	1	0.028	4	0.024	5	0.052	0
add6(1)	1	36	9	4	**14**	1	15	2.561	9	0.019	4	0.028	**14**	**0.047**	1
add6(2)	1	256	64	4	**69**	1	-	-	15	124	4	0.026	**20**	0.026	1
add6(3)	1	1296	324	4	**329**	1	-	-	-	-	4	0.029	**329***	0.029	1
add6(4)	1	5776	1444	4	**1449**	1	-	-	-	-	4	0.035	**1449***	0.035	1
add6(5)	1	24336	6084	4	**6089**	1	-	-	-	-	4	0.026	**6089***	0.26	1
adr4(1)	1	345	324	4	**329**	1	-	-	-	-	4	0.025	**329***	0.025	1
adr4(2)	1	1296	64	4	**69**	1	-	-	-	-	4	0.027	**20***	0.027	1
adr4(3)	1	256	9	4	**14**	1	15	2.57	9	0.019	4	0.026	**14**	**0.045**	1
adr4(4)	1	36	1	4	**5**	0	-	-	1	0.026	4	0.031	**5**	0.026	0
al2(11)	1	125	60	4	**64**	0	-	-	-	-	4	0.021	**64***	0.057	0
alcom(5)	1	12	6	4	**10**	0	12	**0.028**	6	0.025	4	0.3	**10**	0.325	0
b11(5)	1	**6**	2	4	**6**	0	**6**	**0.023**	2	0.025	4	0.025	**6**	0.05	0
b12(6)	1	54	35	4	**39**	0	**20**	**918**	20	1350	4	0.027	24	1350	0
dekoder(0)	1	8	3	4	**7**	0	8	**0.027**	3	0.024	4	0.022	**7**	0.046	0
dekoder(1)	1	**6**	2	4	**6**	0	**6**	**0.024**	2	0.023	4	0.023	**6**	0.046	0
dk27(8)	1	**1**	1	4	5	0	**1**	**0.021**	1	0.026	4	0.024	5	0.05	0
exps(18)	3	**16**	7	12	19	0	**12**	22.2	7	0.027	12	0.067	19	**0.094**	0
exps(19)	6	**16**	7	24	31	0	**16**	**0.03**	7	0.024	24	0.137	31	0.161	0
f51m(6)	1	**4**	1	4	5	0	**4**	**0.03**	1	0.025	4	0.026	5	0.051	0
luc(3)	1	16	9	4	**13**	0	**12**	0.861	9	0.024	4	0.031	13	**0.055**	0
m1(8)	1	15	8	4	**12**	0	**12**	1.65	8	0.333	4	0.024	**12**	**0.357**	0
max1024(0)	8	56	14	32	**46**	0	-	-	14	1.16	32	0.202	**46**	116	0
max1024(1)	8	324	81	32	**119**	6	324	**0.034**	24	2543	32	0.198	**63**	2543	7
max1024(2)	9	1216	304	36	**348**	8	-	-	304	0.036	36	0.23	**348**	0.266	8
max1024(3)	9	3072	800	36	**844**	8	3072	**0.133**	-	-	36	0.22	**844***	0.22	8
max1024(4)	9	6806	1978	36	**2023**	9	6806	**0.23**	1978	0.07	36	0.218	**2023**	0.288	9
max1024(5)	9	14274	3968	36	**4013**	9	-	-	-	-	36	0.221	**4013***	0.221	9
newcond(1)	2	**8**	3	8	11	0	**8**	0.244	3	0.023	8	0.051	11	**0.074**	0
newcwp(0)	2	20	6	8	**15**	1	**12**	0.521	6	0.023	8	0.053	15	**0.076**	1
newcwp(1)	1	16	1	12	**13**	0	**9**	**0.303**	1	0.025	9	0.281	10	0.306	0
newcwp(3)	2	**4**	1	4	5	0	**4**	**0.022**	1	0.021	4	0.027	5	0.048	0
p82(10)	2	10	4	4	**8**	0	**8**	0.106	4	0.021	4	0.025	**8**	**0.046**	0
pope.rom(18)	3	**12**	5	12	17	0	**10**	1.45	5	0.025	12	0.07	17	**0.095**	0
pope.rom(32)	1	8	3	4	**7**	0	**6**	0.086	3	0.027	4	0.028	7	**0.055**	0
pope.rom(34)	1	8	3	4	**7**	0	8	1806	3	0.023	4	0.019	**7**	**0.042**	0
pope.rom(35)	1	**6**	2	8	10	0	**6**	0.077	2	0.025	8	0.05	10	**0.075**	0
pope.rom(41)	1	10	4	4	**8**	0	10	**0.022**	4	0.031	4	0.023	**8**	0.054	0
radd(0)	1	**4**	1	4	5	0	**4**	**0.025**	1	0.021	4	0.026	5	0.047	0
radd(1)	1	36	9	4	**14**	1	15	2.45	9	0.026	4	0.027	**14**	**0.053**	1
radd(2)	1	256	64	4	**69**	1	-	-	15	122.1	4	0.025	**20**	122	1
radd(3)	1	1296	324	4	**329**	1	-	-	-	-	4	0.024	**329***	0.024	1
rd53(1)	4	100	30	16	**50**	4	-	-	12	0.676	16	0.118	**32**	0.794	4
rd53(2)	1	256	1	80	**81**	0	-	-	1	0.022	-	-	**81***	0.022	0
rd73(2)	1	1225	1	448	**449**	0	-	-	1	0.023	-	-	**449***	0.023	0
risc(4)	1	**6**	2	4	**6**	0	**6**	**0.026**	2	0.027	4	0.02	**6**	0.047	0
sqn(0)	2	272	49	8	**58**	1	-	-	15	11	8	0.049	**24**	11.5	1
wim(2)	1	**6**	2	4	**6**	0	**6**	**0.026**	2	0.023	4	0.019	**6**	0.042	0
z4(1)	5	784	64	28	**97**	5	784	**11.34**	15	115	25	0.412	**45**	115	5
z4(2)	3	144	9	20	**32**	3	-	-	9	0.021	17	0.341	**29**	0.362	3
z4(3)	1	16	1	12	**13**	0	**9**	**0.29**	1	0.027	9	0.286	10	0.313	0
z5xp1(8)	1	**4**	1	4	5	0	**4**	**0.027**	1	0.019	4	0.026	5	0.045	0
z9sym(0)	8	6192	2016	32	**2056**	8	-	-	-	-	32	0.217	**2056***	0.276	8

Table 5.6: Proposed lattice sizes for standard benchmark circuits: a comparison of the proposed external D-reducible decomposition method with the results of internal decomposition, [25] and [26]. When the synthesis of a lattice is stopped, there is no lattice (−). Results are marked with * when SAT is stopped.

	[25]								[26]							
	Std. synthesis		Internal Decomp.		External Decomp.				Std. synthesis		Internal Decomp.		External Decomp.			
	X×Y	Area	X×Y	Area	χ	f_A	cost	Area	X×Y	Area	X×Y	Area	χ	f_A	cost	Area
addm4(0)	9×12	180	9×15	135	1×3	9×12	2	**113**	-	-	3×8	24	1×3	3×5	2	**20**
addm4(1)	22×23	506	22×23	506	1×1	22×22	1	**486**	-	-	22×23	506*	1×1	22×22	1	**486***
addm4(2)	33×36	1180	33×36	1188	1×1	33×35	1	**1157**	-	-	33×36	1188*	1×1	33×35	1	**1157***
adr4(4)	2×2	**4**	2×2	**4**	2×3	1×0	0	6	2×2	**4**	2×5	10	2×5	0×0	0	10
alu2(6)	4×4	**16**	4×5	20	1×1	4×4	1	18	3×4	**12**	4×4	16	1×1	4×3	1	14
amd(3)	6×8	**48**	6×9	54	1×1	6×8	1	50	4×5	**20**	3×7	21	1×1	3×6	1	**20**
amd(4)	10×14	140	10×14	140	1×1	10×13	1	**132**	-	-	10×14	140*	1×1	10×13	1	**132***
amd(5)	2×8	**16**	2×9	18	2×6	2×2	2	18	6×2	**12**	2×8	16	2×6	2×2	2	18
amd(6)	3×9	**27**	3×14	42	1×5	3×9	2	34	6×3	18	3×8	24	1×5	3×3	2	**16**
amd(7)	6×7	**42**	6×8	48	1×1	6×7	1	44	4×5	20	3×6	18	1×1	3×5	1	**17**
apla(0)	4×13	52	4×13	52	1×3	4×10	2	**45**	6×3	30	5×6	30	1×3	5×3	2	**20**
apla(1)	4×12	48	4×12	48	1×3	4×9	2	**41**	6×3	18	4×6	24	1×3	4×3	2	**17**
apla(2)	3×6	**18**	5×18	90	1×2	5×16	2	84	5×2	**10**	4×4	16	1×2	4×2	2	12
apla(7)	5×10	**50**	6×12	72	1×2	6×10	2	64	-	-	4×6	24	1×2	4×4	2	**20**
apla(9)	5×15	**75**	6×22	132	1×2	6×20	2	124	-	-	5×6	30	1×2	5×4	2	**24**
b10(2)	10×14	**140**	10×19	190	1×5	10×14	2	147	-	-	10×14	140*	1×5	10×9	2	**97***
br1(3)	2×12	**24**	2×13	26	2×13	2×1	2	30	-	-	2×12	**24**	2×11	2×1	2	26
br1(4)	6×15	**90**	6×20	120	1×5	6×15	2	97	-	-	5×9	45	1×5	5×4	2	**27**
br2(4)	8×18	**144**	8×20	160	1×2	8×18	2	148	-	-	8×18	144*	1×2	8×16	2	**132***
br2(5)	4×14	56	8×16	128	2×9	4×6	2	**44**	-	-	4×21	84*	2×17	4×3	2	**48***
br2(6)	5×16	80	5×17	85	2×7	5×9	2	**61**	-	-	4×12	48*	2×7	4×4	2	**32***
dk48(2)	2×13	**26**	2×22	44	1×9	2×13	2	37	-	-	2×13	26*	1×9	2×4	2	**19***
dk48(3)	2×13	**26**	4×17	68	2×15	1×0	0	30	-	-	2×15	**30***	2×15	0×0	0	**30***
exam(4)	9×25	225	11×22	242	1×2	11×20	2	**224**	-	-	9×20	180*	1×2	9×18	2	**166***
exp(6)	6×7	**42**	8×12	96	1×2	8×10	2	84	5×4	20	3×7	21	1×2	3×5	2	**19**
exp(10)	6×12	**72**	6×15	90	1×2	6×13	2	82	-	-	5×6	30	1×2	5×4	2	**24**
exp(11)	6×12	72	5×12	60	2×4	5×7	2	**45**	-	-	5×8	40	2×4	5×3	2	**25**
gary(2)	12×14	168	12×15	180	2×3	12×12	2	**152**	-	-	12×18	216*	2×5	12×12	2	**156***
gary(3)	5×12	**60**	5×14	70	1×2	5×12	2	64	-	-	5×12	60	1×2	5×10	2	**54**
in2(6)	39×36	**1404**	39×38	1482	1×2	39×36	2	1408	-	-	39×35	1365*	1×2	39×33	2	**1291***
in2(7)	17×26	**442**	17×27	459	1×1	17×26	1	444	-	-	17×26	442*	1×1	17×25	1	**427***
in2(8)	27×31	**837**	27×32	864	1×1	27×31	1	839	-	-	27×31	837*	1×1	27×30	1	**812***
in2(9)	40×36	**1440**	40×37	1480	1×1	40×36	1	1442	-	-	40×36	1440*	1×1	40×35	1	**1402***
in7(6)	11×18	**198**	11×21	231	1×3	11×18	2	203	-	-	11×18	198*	1×3	11×15	2	**170***
m2(6)	10×13	**130**	10×14	140	1×1	10×13	1	132	-	-	10×13	130*	1×1	10×12	1	**122***
m2(7)	14×14	**196**	14×15	210	1×1	14×14	1	198	-	-	14×14	196*	1×1	14×13	1	**184***
m2(12)	6×11	**66**	6×13	78	1×2	6×11	2	70	5×4	**20**	4×6	24	1×2	4×4	2	**20**
m2(13)	9×12	**108**	9×14	126	1×2	9×12	2	112	-	-	4×8	32	1×2	4×6	2	**28**
m2(15)	16×18	**288**	17×19	323	1×1	17×18	1	308	-	-	16×18	288*	1×1	16×17	1	**274***
m4(9)	4×7	**28**	4×10	40	1×3	4×7	2	33	5×3	**15**	4×6	24	1×3	4×3	2	17
m4(10)	7×7	**49**	7×9	63	1×2	7×7	2	53	7×7	49	3×8	24	1×2	3×6	2	**22**
max128(3)	6×6	36	5×7	35	2×3	5×4	2	**28**	3×6	**18**	3×11	33	2×5	3×5	2	27
newapla1(6)	5×6	**30**	5×11	55	1×5	5×6	2	37	-	-	5×6	30	1×5	5×1	2	**12**
newapla(0)	4×6	**24**	4×7	28	1×1	4×6	1	26	3×6	**18**	3×7	21	1×1	3×6	1	20
newcpla1(6)	6×11	**66**	6×13	78	1×2	6×11	2	70	-	-	4×6	24	1×2	4×4	2	**20**
newcpla1(7)	7×6	**42**	7×7	49	1×1	7×6	1	44	4×5	20	3×6	18	1×1	3×5	1	**17**
newcpla1(8)	7×12	**84**	7×15	105	1×3	7×12	2	89	-	-	5×7	35	1×3	5×4	2	**25**
newtpla1(1)	2×9	**18**	2×15	30	1×6	2×9	2	26	2×9	18	2×9	18	1×6	2×3	2	**14**
newtpla2(2)	2×9	**18**	2×17	34	1×8	2×9	2	28	2×9	18	2×9	18	1×8	2×1	2	**12**
newtpla(6)	3×12	**36**	3×16	48	1×4	3×12	2	42	-	-	6×7	42	1×4	6×3	2	**24**
newtpla(9)	2×7	**14**	2×13	26	1×6	2×7	2	22	6×2	12	2×7	14	1×6	2×1	2	**10**
newxcpla1(1)	7×6	**42**	7×7	49	1×1	7×6	1	44	3×6	18	3×6	18	1×1	3×5	1	**17**
p1(2)	3×6	**18**	7×10	70	2×6	7×4	2	42	4×3	**12**	2×9	18	2×5	2×3	2	18
p1(11)	6×9	54	5×12	60	2×4	5×7	2	**45**	3×5	**15**	5×8	40	2×4	5×3	2	25

Table 5.7: Proposed lattice sizes for standard benchmark circuits: a comparison of the proposed external P-circuit decomposition method, [25] and [26]. When the synthesis of a lattice is stopped, there is no lattice (−). Results are marked with ⋆ when SAT is stopped.

	[25]									[26]								
	Std. synthesis		Internal Decomp.		External Decomp.					Std. synthesis		Internal Decomp.		External Decomp.				
	X×Y	Area	X×Y	Area	$A_{f^=}$	$A_{f^\neq}$	$A_{f^!}$	cost	Area	X×Y	Area	X×Y	Area	$A_{f^=}$	$A_{f^\neq}$	$A_{f^!}$	cost	Area
adr4(1)	36×36	1296	37×19	703	324	324	0	5	**653**	–		37×19	703⋆	324⋆	324	0	5	**653**⋆
alu2(2)	11×10	110	13×7	91	6	5	56	7	**74**	7 × 3	**21**	10×6	60	6	5	15	7	33
alu2(5)	14×13	182	16×10	160	8	6	110	7	**131**	–		16×10	160⋆	8	6	110	7	**131**⋆
alu3(0)	5×4	20	7×4	28	2	3	4	7	**16**	3 × 3	**9**	7×4	28	2	3	4	7	16
alu3(1)	8×7	56	10×5	50	4	4	20	7	**35**	5 × 3	**15**	8×5	40	4	4	9	7	24
alu3(2)	10×11	110	12×8	96	4	8	56	7	**75**	6 × 4	**24**	11×5	55⋆	4	8	18	7	37⋆
b12(0)	4×6	24	4×6	24	20	0	0	3	**23**	3 × 4	**12**	3×4	**12**⋆	9	0	0	3	**12**⋆
b12(1)	7×5	35	7×5	35	28	0	0	3	**31**	4 × 4	16	4×4	16⋆	12	0	0	3	**15**⋆
bcc(5)	9×27	243	9×26	234	225	0	0	3	**228**	–		9×26	234⋆	225	0	0	3	**228**⋆
bcc(7)	11×31	341	12×29	348	280	10	0	5	**295**	–		12×29	348⋆	280	10	0	5	**295**⋆
bcc(8)	12×31	372	13×29	377	308	10	0	5	**323**	–		13×29	377⋆	308	10	0	5	**323**⋆
bcc(12)	11×31	341	11×30	330	319	0	0	3	**322**	–		11×30	330⋆	319	0	0	3	**322**⋆
bcc(27)	19×39	741	20×33	660	416	90	0	5	**511**	–		20×33	660⋆	416	90	0	5	**511**⋆
bcc(43)	10×20	200	11×16	176	78	60	0	5	**143**	–		11×16	176⋆	78	60	0	5	**143**⋆
bcd.div3(1)	3×4	**12**	5×4	20	3	1	2	7	13	3 × 3	**9**	5×4	20	3	1	2	7	13
bcd.div3(2)	3×4	**12**	5×4	20	3	1	3	7	14	3 × 3	**9**	5×4	20	3	1	3	7	14
bcd.div3(3)	3×5	15	4×4	16	6	0	3	5	**14**	3 × 4	**12**	4×4	16⋆	6	0	3	5	14⋆
bench1(2)	24×45	1080	33×29	957	350	504	0	5	**859**	–		33×29	957⋆	350	504	0	5	**859**⋆
bench1(3)	16×31	496	20×18	360	104	136	14	7	**261**	–		21×18	378⋆	104	136	12	7	**259**⋆
bench1(5)	27×50	1350	32×28	896	128	78	448	7	**661**	–		32×28	896⋆	128	78	448	7	**661**⋆
bench1(6)	21×35	735	26×24	624	160	345	0	5	**510**	–		26×24	624⋆	160	345	0	5	**510**⋆
bench1(7)	21×43	903	27×20	540	247	190	14	7	**458**	–		27×20	540⋆	247	190	10	7	**454**⋆
bench1(8)	24×44	1056	31×26	806	375	345	0	5	**725**	–		31×26	806⋆	375	345	0	5	**725**⋆
bench(6)	4×8	32	6×3	18	2	2	6	7	**17**	3 × 4	**12**	6×3	18	2	2	6	7	17
br2(4)	8×18	144	8×18	144	0	136	0	2	**138**	–		8×18	144⋆	0	136	0	2	**138**⋆
br2(5)	4×14	56	4×14	56	0	52	0	2	**54**	–		4×14	56⋆	0	52	0	2	**54**⋆
br2(6)	5×16	80	5×16	80	0	75	0	2	**77**	–		5×16	80⋆	0	75	0	2	**77**⋆
clpl(2)	2×2	**4**	3×2	6	0	1	4	4	6	2 × 2	**4**	3×2	6⋆	0	1	1	4	6⋆
clpl(3)	6×6	**36**	9×6	54	1	10	20	6	37	6 × 3	**18**	9×6	54⋆	1	10	12	7	30⋆
clpl(4)	5×5	**25**	8×5	40	1	8	12	6	27	5 × 3	**15**	8×5	40⋆	1	8	9	7	25⋆
co14(0)	14×92	1288	15×80	1200	1027	13	0	5	**1045**	–		15×80	1200⋆	1027	13	0	5	**1045**
dc1(0)	4×4	**16**	5×4	20	9	2	0	5	**16**	3 × 3	**9**	4×4	16⋆	6	2	0	5	13⋆
dc1(1)	2×3	**6**	3×3	9	2	0	3	5	10	2 × 3	**6**	3×3	9⋆	2	0	3	5	10⋆
dc1(4)	4×5	20	5×4	20	9	0	3	5	**17**	3 × 4	**12**	5×4	20⋆	9	0	3	5	17⋆
dc1(6)	3×3	9	4×2	**8**	2	0	2	5	9	3 × 2	**6**	4×2	8⋆	2	0	2	5	9⋆
dc2(4)	9×10	90	10×9	90	48	18	0	5	**71**	4 × 5	**20**	8×5	40⋆	16	12	0	5	33⋆
dc2(5)	6×6	36	7×7	49	12	18	0	5	**35**	2 × 6	**12**	5×6	30⋆	8	10	0	5	23⋆
dk17(0)	2×8	16	4×4	16	6	2	0	5	**13**	2 × 6	**12**	4×4	16⋆	6	2	0	5	13⋆
dk17(1)	2×8	16	4×4	16	6	2	0	5	**13**	2 × 6	**12**	4×4	16⋆	6	2	0	5	13⋆
dk17(3)	3×11	33	4×7	**28**	0	0	28	0	**28**	2 × 7	**14**	6×3	18⋆	0	0	18	0	18⋆
dk17(4)	3×9	27	6×4	24	3	2	6	7	**18**	2 × 7	**14**	6×4	24	3	2	6	0	18
dk17(6)	1×3	**3**	1×3	**3**	0	0	3	0	**3**	1 × 3	**3**	1×3	**3**⋆	0	0	3	0	**3**⋆
exp(4)	6×17	**102**	6×17	**102**	0	0	102	0	104	–		6×17	**102**⋆	0	0	102	0	**102**⋆
exp(5)	45×35	**1575**	45×35	**1575**	0	0	1575	0	1577	–		45×35	**1575**⋆	0	0	1575	0	**1575**⋆
exp(32)	10×4	40	13×3	39	10	8	4	7	**29**	6 × 4	**24**	13×3	39	10	8	4	7	29
exp(33)	7×3	21	7×3	21	0	1	15	4	**20**	–		7×3	21⋆	0	1	15	4	**20**⋆
exp(34)	10×4	**40**	12×5	60	2	4	30	7	43	6 × 4	**24**	11×3	33	2	4	15	7	28
exp(36)	8×2	**16**	10×2	20	1	1	12	6	20	8 × 2	**16**	10×2	20	1	1	12	7	21
exp(38)	9×4	36	13×3	39	12	8	1	6	**28**	7 × 4	**24**	13×3	39	12	8	1	7	28
exp(39)	8×2	**16**	11×3	33	1	8	8	6	23	–		11×3	33	1	8	8	7	**24**
exp(40)	12×6	72	15×5	75	16	18	12	7	**53**	–		13×4	52	8	12	9	7	**36**
exp(43)	14×8	112	17×6	102	36	35	0	5	**76**	–		13×4	52⋆	18	18	0	5	**41**⋆
exam(5)	6×11	66	7×6	42	4	0	30	5	**39**	–		6×5	30⋆	4	0	16	5	**25**⋆
exam(9)	30×59	1770	38×30	1140	754	143	0	5	**902**	–		33×30	990⋆	754	24	0	5	**783**⋆
max128(5)	17×14	238	19×9	171	16	80	21	7	**124**	–		14×5	70	9	24	12	7	**52**
max128(8)	10×5	50	11×4	44	18	8	0	5	**31**	–		10×4	40⋆	15	8	0	5	**28**⋆
max128(17)	25×26	650	26×15	390	144	182	0	5	**331**	–		26×15	390⋆	144	182	0	5	**331**⋆
mp2d(6)	6×10	60	6×10	60	0	54	0	2	**56**	–		3×7	21⋆	0	18	0	2	**20**⋆
mp2d(9)	8×6	48	9×6	54	0	15	5	4	**24**	–		9×4	36⋆	0	9	5	4	**18**⋆
mp2d(10)	3×6	18	4×5	20	8	4	0	5	**17**	3 × 4	**12**	4×5	20⋆	8	4	0	5	17⋆
z4(0)	15×15	225	16×11	176	0	24	77	4	**105**	4 × 5	**20**	6×6	36⋆	0	10	12	4	26⋆
z4(1)	28×28	784	30×16	480	32	32	192	7	**263**	–		10×7	70	12	12	24	7	**55**
Z5×p1(2)	11×12	132	13×7	91	12	36	8	7	**63**	–		11×5	55⋆	12	16	8	7	**43**⋆
Z5×p1(3)	18×18	324	19×11	209	80	80	0	5	**165**	–		10×6	60⋆	20	20	0	5	**45**⋆

with the results obtained with the standard synthesis methods presented in [25] and in [26], without exploiting the autosymmetry property. In Table 5.6 and Table 5.7 we compare the lattice synthesis results obtained applying the external decomposition scheme to D-reducible functions and P-circuits, with the corresponding internal decomposition scheme and the results obtained with the standard synthesis methods presented in [25] and in [26], without exploiting the decomposition property.

To simulate the results reported in [26], we used a collection of Python scripts for computing minimum-area switching lattices, by transformation to a series of SAT problems.

Each row of Table 5.5 lists the results for each separate autosymmetric output function of the benchmark circuit. More precisely, the first column reports the name and the number of the considered output of each instance; the second column reports the number of EXOR lattices used to implement the reduction equations (y_j) when the decomposition method is applied. The following five columns refer to the synthesis of lattices as described in [25], with (columns 4-7) and without (column 3) the multiple lattice decomposition based on autosymmetry. In particular, column 3 shows the area of lattices derived applying the standard synthesis method (i.e., without exploiting the autosymmetry property), column 4 shows the area of the lattice for the restriction f_k, column 5 shows the total area of the lattices for the EXOR terms y_i, column 6 shows the total area occupied by lattices ($TotalArea = A_{f_k} + \sum_i A_{y_i} + num.inv$), and column 7 indicates the number $num.inv$ of inverters necessary to make the signal routing. The synthesis in [25] is performed using ESPRESSO, and in all cases it takes less than 0.01 s, that is the minimum time resolution of the synthesizer; for this reason the synthesis time is omitted.

Columns 8 to 16 refer to lattices synthesized using the methodology presented in [26], with and without decomposition on multiple lattices. In particular columns 8 and 9 report the area and the synthesis time of lattices obtained with standard synthesis; columns 10 and 11 report the area and the synthesis time of the lattice for the restriction f_k, column 12 and 13 report the total area of the lattices for the EXOR terms y_i and their synthesis time; columns 14 and 15 show the total area occupied by lattices ($TotalArea = A_{f_k} + \sum_i A_{y_i} + num.inv$) and the total synthesis time; finally, column 16 indicates the number $num.inv$ of inverter necessary for signal routing.

For each function, we bolded the best areas (col. 3 vs col. 5 vs col. 6 and col. 8 vs col. 14) and the best total time (col. 9 vs col.15).

Each row of Table 5.6 lists the results for each separate D-reducible output function of the benchmark circuits. More precisely, the first column reports the name and the number of the considered output of each instance; The following two columns refer to the synthesis of lattices as described in [25], without lattice decomposition (columns 2-3), with internal decomposition (column 4-5), and external

decomposition (column 6-9). In particular, columns 2-3 show the dimension and the area of lattices derived applying the standard synthesis method (i.e., without exploiting the D-reducibility property), columns 4-5 show the dimension and the area of the lattice obtained applying the internal decomposition method, column 6 shows the dimension of χ, column 7 shows the dimension of the lattice of f_A, column 8 shows the cost in term of lattice area due to external decomposition, column 9 shows the total area occupied by lattices ($TotalArea = A_\chi + A_{f_A} + cost$).

Columns 10 to 17 of Table 5.6 refer to lattices synthesized using the methodology presented in [26], the content refers to the same lattice as columns 2-9.

For each function, we bolded the best areas (col. 3 vs col. 5 vs col. 9 and col. 11 vs col. 13 vs col. 17).

Each row of Table 5.7 lists the results for each separate output function, represented as a P-circuit, of the benchmark circuits. More precisely, the first column reports the name and the number of the considered output of each instance. The following two columns refer to the synthesis of lattices as described in [25], without lattice decomposition (columns 2-3), with internal decomposition (column 4-5), and external decomposition (column 6-10). In particular, columns 2-3 show the dimension and the area of lattices derived applying the standard synthesis method (i.e., without exploiting the P-circuit decomposition), columns 4-5 show the dimension and the area of the lattice obtained applying the internal decomposition method, column 6 shows the dimension of the lattice of the projection $f^{=}$, column 7 shows the dimension of the lattice of the projection $f^{\neq}$, column 8 shows the dimension of the lattice of the intersection f^I, column 9 shows the cost in term of lattice area due to external decomposition, and column 10 reports the total area occupied by lattices ($TotalArea = A_{f^=} + A_{f^\neq} + A_{f^I} + cost$).

Columns 11 to 19 of Table 5.7 refer to lattices synthesized using the methodology presented in [26], the content refers to the same lattice as columns 2-10.

For each function, we bolded the best areas (col. 3 vs col. 5 vs col. 10 and col. 12 vs col. 14 vs col. 19).

In some cases the method proposed in [26] fails in computing a result in reasonable run time. For this reason, we set a time limit (equal to ten minutes) for each SAT execution; if we do not find a solution within the time limit, the synthesis is stopped. We marked with $-$ all cases where the synthesis of lattices has been stopped. In the synthesis of sublattices, whenever [26] is stopped, we use the sublattices synthesized with [25], because without a sublattice it would be impossible to complete the synthesis of the overall decomposed lattice. We marked these cases with $\star$. Note that, for many benchmarks, the method in [26] did not find a solution within the fixed time limit for at least one sublattice, and had to be replaced with [25].

The results are promising. Considering the methodology presented in [25], for the class of autosymmetric functions (see Table 5.5) we obtain a smaller total area

Table 5.8: Comparison between external decomposition method with not decomposed lattice and internal decomposed lattice

Decomposition method		Not-decomposed lattices		Lattice with internal decomposition	
		less area	area gain	less area	area gain
D-reducible	[25]	16%	9%	78%	9%
	[26]	54%	9%	75%	6%
P-circuit	[25]	54%	36%	93%	15%
	[26]	60%	39%	93%	17%
Autosymmetric	[25]	58%	53%	–	–
	[26]	48%	60%	–	–

w.r.t. the standard synthesized lattices in 58% of the benchmarks, with an average gain of 53%. Considering the methodology presented in [26], we obtain a smaller total area in 48% of the benchmarks, with an average gain of 60%. Note that in many cases the synthesis time necessary to decompose the function as described in this paper (column 15 in Table 5.5) is smaller than the time necessary to perform the standard synthesis (column 9 in Table 5.5).

As for external vs internal decomposition, we report in Table 5.8 the overall results concerning the external decomposition applied to all different decomposition methods with respect to standard synthesized lattices and internal decomposition. In this table, each row is referred to a different decomposition method. Column 2 shows the synthesis method used for the experiment, columns 3 and 4 show the percentage of lattices with less area and how much area is gained with respect to not-decomposed lattice, columns 5 and 6 show the percentage of lattices with less area and how much area is gained with respect to internal decomposed lattices. The comparison between internal and external decomposition shows a high percentage of lattices where external decomposition produces lattices with less area with respect to internal decomposition. This is due to the typical construction of external decomposition that can omit the padding cells and columns or rows of ones or zeros to connect the decomposed sub-lattices together. Note that a direct comparison between columns 3-4 and 5-6 is not correct because not all the lattices with internal decomposition are smaller that the not-decomposed lattices, and a lattice with external decomposition can be smaller that a lattice with internal decomposition but bigger that the non-decomposed lattice.

These results clearly show how the use of multiple lattices often allows to reduce the number of switches and thus the overall dimension of the lattice, even if the gain in the dimension comes at the expense of an increase in the interconnection cost.

Chapter 6

Defect Tolerance

The fault density of crossbars can be up to 10% of the entire crossbar, we consider all faults independent, as reported in [103].

The fault injection in lattices is performed substituting a single cell with an always Stuck-At 1 (SA1) or Stuck-At 0 (SA0) cell. The fault injection procedure is repeated for each cell of the lattice.

We use a method that utilizes a prior sensitivity analysis of crossbar to specify critical switches, and strengthens them with proposed mitigation factors. The same naming conventions are applicable, regarding defects which are categorized as SA1 and SA0.

6.1 Defect Injection Methodology

We perform a defect injection with uniform distribution to lattice reaching defect densities up to 10%. Every cell (a four-terminal switch) is presumed to have only SA0 or SA1. Once the "defective" lattice is obtained, the algorithm generates all the possible 2^n inputs (where n is the number of variables). For each input, the simulation algorithm compares the given output with the correct one. Let E^0_{ij} (resp., E^1_{ij}), with $1 \leq i \leq r$, $1 \leq j \leq s$ (where r are the number of rows and s the number of columns of the lattice), be the number of defective outputs with a SA0 (resp., SA1) in the cell (i, j) of the given lattice. Note that $0 \leq \{E^0_{ij},\ E^1_{ij}\} \leq 2^n$. Moreover, when E^0_{ij} (resp., E^1_{ij}) is equal to 0 we have that, for any possible input, the lattice output is never changed by the SAF in the cell (i, j). In this case, we call the cell (i,j) robust w.r.t. SA0 (resp., SA1). Let R^0 (resp., R^1) be the total number of robust cells w.r.t. SA0 (resp., SA1) in the lattice. Finally, let $E^0 = \sum_{i=1}^{i=r} \sum_{j=1}^{j=s} E^0_{ij}$ (resp., $E^1 = \sum_{i=1}^{i=r} \sum_{j=1}^{j=s} E^1_{ij}$) be the total number of defective output with SA0 (resp. SA1) in the simulation. For an example of function $f = x_4\overline{x_5}x_7 + \overline{x_4}x_6\overline{x_7} + \overline{x_4}x_5\overline{x_6}x_7 + x_4\overline{x_6x_7} + x_4x_6x_7$ realized in

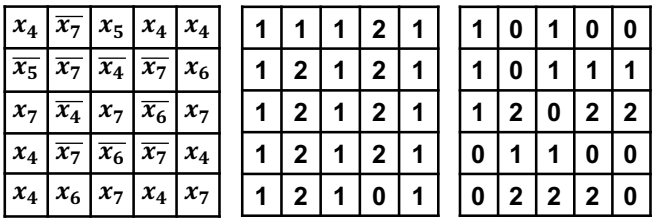


Figure 6.1: a) Lattice design for the example function f and its sensitivity map for b) SAO and c) SA1.

Figure 6.1(a) (with the method in [25]), in the Figure 6.1(b) (resp., 6.1(c) shows the map containing E^0_{ij} (resp., E^1_{ij}) in each cell.

6.2 Metrics used for Sensitivity Analysis

In order to evaluate the sensitivity of a lattice to SA0 and SA1 defects, we propose two metrics. The first one measures the average number of defective outputs considering sensitive cells to SA0 or SA1 only. The second one measures the average number of defective outputs in the entire lattice. Note that the total number of cells is the area of the lattice (i.e., $r \times s$), the number of non-robust cells for SA0 (resp., SA1) is $r \times s - R^0$ (resp., $r \times s - R^1$), and 2^n is the total number of inputs. (1) Sensitivity of defective cells is the total number of inputs that give an uncorrected output (E^0 and E^1) divided by the total number of inputs (2^n), for each non-robust cell ($r \times s - R^0$ or $r \times s - R^1$). In the case of SA0 the metric can be expressed as: $S^0_C = E^0/(2^n(r \times s - R^0))$. The same reasoning can be done for SA1 defect sensitivity. (2) Sensitivity of lattice is the total number of inputs that give an uncorrected output divided by the total number of inputs for each cell, in the case of SA0 is: $S^0_L = E^0/(2^n(r \times s))$. The SA1 case is analogous.

6.2.1 Benchmarks and Simulations

The defect simulations have been run on a machine with two AMD Opteron 4274HE for a total of 16 CPUs at 2.5 GHz and 128 GByte of main memory, running Linux CentOS 7. The benchmarks functions are expressed in PLA form and are taken from a subset of LGSynth93 [82]. A total of about 580 functions were considered, and each output of a function is implemented as a separate Boolean function.

The software used for simulations is written in C++. We used ESPRESSO to implement the method described in [25], and a collection of Python scripts for computing minimum-area lattices by transformation to a series of SAT problems, to simulate the results reported in [26]. Each SAT execution is stopped after ten minutes.

Table 6.1: A sample of benchmark functions synthesized with [25] and [26] approaches and their sensitivity values

name	$r \times s$	n	E^0	S_C^0	S_L^0	$\%\frac{R^0}{r\times s}$	E^1	S_C^1	S_L^1	$\%\frac{R^1}{r\times s}$
Synthesis with Dual Method [25]										
add6(1)	6×6	4	19	0.06	0.03	47%	9	0.06	0.02	75%
alu2(2)	11×10	8	462	0.03	0.02	35%	121	0.02	0.01	80%
b11(1)	3×6	7	28	0.02	0.01	44%	73	0.03	0.03	6%
dc2(0)	4×6	7	117	0.05	0.04	17%	162	0.08	0.05	33%
exam(5)	6×11	9	1868	0.07	0.06	17%	131	0.02	0.01	74%
z4(2)	12×12	5	70	0.03	0.02	51%	14	0.03	0	90%
Synthesis with Quantified Boolean Logic [26]										
add6_G_1	5×3	4	31	0.15	0.13	13%	32	0.14	0.13	7%
alu2_G_2	7×3	8	464	0.1	0.09	14%	384	0.08	0.07	5%
b11_G_1	3×5	7	45	0.03	0.02	7%	128	0.07	0.07	7%
dc2_G_0	4×4	7	104	0.06	0.05	13%	132	0.07	0.06	13%

In Table 6.1, we report a sample of benchmark functions and their sensitivity values, according to the metrics presented before. In particular, Table 6.1 refers to lattice synthesized as described in [25] and [26]. The benchmarks that are present in Table 6.1 with dual method were stopped after ten minutes of SAT execution, but that was not the case for the rest.

More precisely, in both methods, the first column reports the name and the number of the considered output of each function. The following columns report dimension ($r \times s$) required for the synthesis of a given function according to each decomposition method, and the number of input variables n. Columns from 4 to 7 refers to SA0 defect metrics (resp., columns from 8 to 11 to SA1 metrics) showing the total number of errors E^0, the Sensitivity of defective cells S_C^0, the Sensitivity of lattice S_L^0 and the percentage of robust cells $\%R^0/(r \times s)$.

Table 6.2 describes the overall results for the benchmarks we have considered. It also shows the average values for the considered metrics. We can note that the percentage of cells that are considered robust according to our metrics is higher in the first approach [25]. This is due to the more constrained structure of the lattices produced by the first synthesis method. Indeed, the method proposed in [25] computes a lattice for f and its dual that is in general less compact than the lattice given by [26] (see, the column Average area in Table 6.2). Moreover, we can note that the sensitivity of the lattice to stuck-at-defects (SAD) is quite low for both methods. In fact, the experiments show that, in general, non-robust cells -in presence of a SAD- compute a defective output for a very limited number of inputs.

Table 6.2: Overall results of the simulations

Synthesis Method	Average area	Average n	S_C^0	S_L^0	$\%\frac{R^0}{r\times s}$	S_C^1	S_L^1	$\%\frac{R^1}{r\times s}$
[25]	30	6	0.05	0.05	20%	0.06	0.05	29%
[26]	15	7	0.07	0.06	9%	0.07	0.07	8%

6.3 Mitigation by Defect Avoidance

From the above results, it can be seen that the two analyzed mapping algorithm shows different sensitivities of the output of a given function. As a matter of fact, the more restrictive an algorithm is in terms of area (results closer to optimal solution), the higher the defect sensitivity of the output to cell defect of SA0 or SA1. It is mandatory to include in the mapping algorithm defect-avoidance heuristics.

In order to mitigate the sensitivity of a lattice to SAD, we propose the following possible strategy applied to the synthesis method proposed in [25] which has been proven as less sensitive to SAD impact on the output functions: (1) For a given mapped function, if a potential SA0, SA1 defect affects a robust cell identified by the defect injection campaign, the lattice still computes the correct output, thus we do not need any mitigation with defect tolerant design. (2) However, if an injected defect occurs in a multiple-choice cell, if a different literal can be chosen to make the cell robust, we change the literal with the new one. (3) Otherwise, if the injected SA0 defect is proven as being critical for the output value, the column that contains that defective cell has to be replaced by spare columns. In case of an SA1 the row that contains the defective cell has to be replaced by a spare row. Note that, in this case, the output still provides a correct function f from top to bottom, but the function from left to right could be changed and become a function which will not be dual of f anymore.

As an example, consider the lattice synthesized in Figure 6.2(a) with $f = x_4\overline{x_5}x_7 + \overline{x_4}x_6\overline{x_7} + \overline{x_4}x_5\overline{x_6}x_7 + x_4\overline{x_6x_7} + x_4x_6x_7$ by using synthesis method presented in [25]. The example shows one case of mitigation of 3 independent SAD affecting the crossbar implementing the function, yielding an approximative 10% defects. In Figure 6.2, SA1 cells are marked in blue and SA0 cells a remarked in red.

To avoid output errors due to these SAD we have used the following strategy:

1. Identify robust cells for a given function mapping. Example: the defect in first row, fifth column is non-influent on the value of the output (robust cell), the sensitivity map, obtained through defect injection campaign, shows that this cell is not sensitive to SA1 for the mapped function.

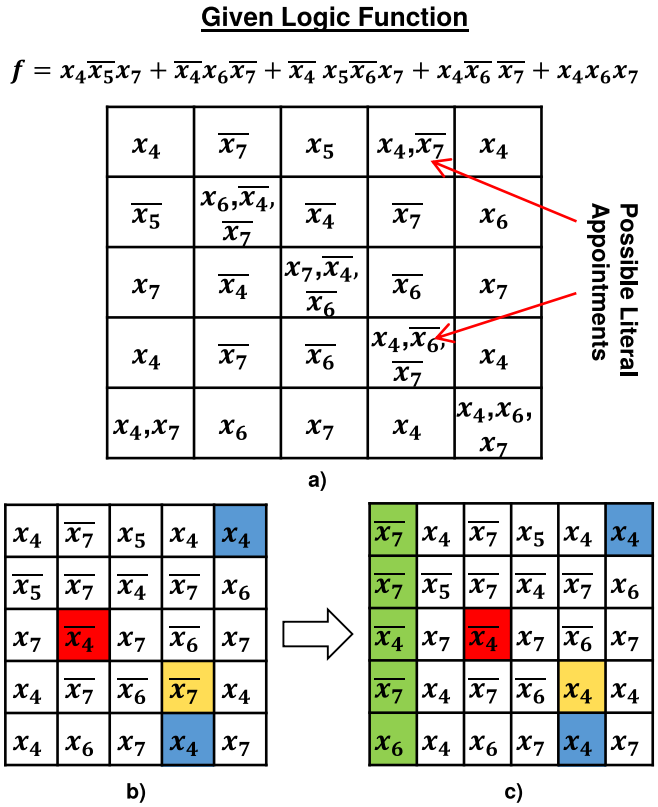


Figure 6.2: a) defect-free lattice; b) lattice with defects: SA0 in red and SA1 in blue; and c) lattice with the defect fixed.

2. Identify the swapping of literals during synthesis process on a column of a high sensitive cell. Example: the defect in fifth row, fourth column is sensitive to SA1. The former choice in the yellow cell was $\overline{x_7}$, choosing x_4 the fifth row, fourth column cell, if it will be affected by a SA1 at fabrication time, will not affect the output of the function.

3. Identify the critical cell for the output value and add a spare column per critical cell. Example: the defect in the third row and second column will influence the value of the output and no swapping operands is possible, thus the only solution remains to add a spare column (in green) identical to the column containing the SA0 defect, and perform the spare and replace strategy. By using spare columns, the mapping algorithm can eliminate columns of the crossbar susceptible to affect the output value of the function in case SA defect appear at fabrication or in the field.

Chapter 7

Conclusions

7.1 Concluding remarks

The first part of the thesis presented some design and synthesis techniques for a high density IC. After an analysis of the design issues typical of scaled technology nodes the applications and the architecture of the AMs chip were described. The AM chip is used inside the data acquisition chain of ATLAS.

The second part of the thesis presented the switching lattices that are one of the emerging post-CMOS technologies. Switching lattices use a 4-terminal switch that allows to implement logic functions in a very compact way. Logic synthesis of switching lattices is a necessary step for large scale adoption and it is important to optimize lattice size and synthesis computing time.

Part I presents a new AM cell, called KOXORAM, that uses a new comparison logic based on the propagation of a kill signal. KOXORAM consumes less energy with respect to previous AM cells (Figure 7.1), fulfilling the requirements of energy consumption and speed. This cell was implemented in the AM07 test chip. Measurements shown an energy consumption of $0.69\,\mathrm{fJ/bit}$ at a speed of $184.32\,\mathrm{MHz}$. During the test AM07 showed metastability on SRAM cells. A design methodology, based on SRAM analysis with butterfly diagrams, it is developed to avoid metastability.

A new memory line architecture, called KOXORAM+, was designed to match the aggressive speed requirements. Simulations of KOXORAM+ show an energy consumption of $0.42\,\mathrm{fJ/bit}$ at a working frequency of $400\,\mathrm{MHz}$. KOXORAM+ will be implemented in AM08 test chip.

In an AM chip digital circuits implemented using standard cells occupies a lot of area. Standard cells synthesis of digital circuits speeds up the design, but the resulting circuits occupy more silicon area and consume more power with respect

to a full custom digital circuit because they contains more transistors and metal interconnections. For this reason a full custom quorum logic has been presented. The quorum occupies, with respect to the same circuit designed with standard cells, about one fifth of the area and consumes $3.75\,\mathrm{aJ/bit}$.

AM chips present a very peaked current consumption at each clock rising edge. To mitigate it we spread clocks with different phases, derived from the input clock, to different chip cores. In this way the power consumption is distributed during the whole clock period. A DCO, that covers a frequency range between $2\,\mathrm{GHz}$ and $3.2\,\mathrm{GHz}$, is used to lock the external clock and to generate the clocks with different phases.

The next AM chip prototype (AM08), where will be included all the circuits presented in this thesis, will be submitted to the silicon foundry in May 2019. The test of AM08 will start in October 2019.

Part II presents the synthesis of switching lattices using an algorithmic approach based on Boolean function decompositions (P-circuit and EP-SOP) and regularities (D-reducible and autosymmetric functions).

The used approach consists on dividing the synthesis problem in smaller problems reducing computing time and lattice size using the presented decomposition techniques and function regularities. In particular we studied two approaches inter-

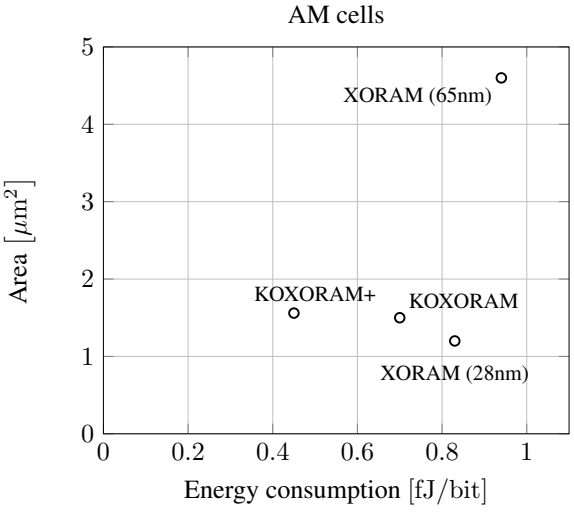


Figure 7.1: Graph that compare the energy consumption and the area of different AM cells. On x axis there is the energy consumption for each bit, on y axis the area of a single cell. The value for KOXORAM+ is obtained with a simulation with parasitics, the other values are obtained with measurements.

nal and external decomposition. Internal decomposition consists on implementing the output of the preprocessing process into a single lattice. External decomposition uses a two layer approach connecting the output of a lattice with one or more literals occurring in another lattice.

Simulation shows that these approaches lead to a reduction of synthesis time and lattice area. There are some cases of external decomposition where the sum of the lattices areas is smaller than the area of the minimal single lattice.

Nanocrossbars are produced using self assembly nanotechnologies that can lead to fabrication defects. For this reason we present a preliminary study on defect avoidance that considers single SA1 or SA0 events. We define a metric for sensitivity analysis and a simulation of the effects of defects on lattices. Moreover we present some techniques that can be used to mitigate the effect of faults.

7.2 Future developments

To reduce area occupation and power consumption of standard cell digital circuits it would be interesting to redesign the AM readout circuit using a full custom approach. However it is necessary to take into account possible drawbacks of this approach: the full custom design requires a great effort, due also to the need of exhaustive simulations, and once the design process is finished it is difficult to implement subsequent circuit modifications.

The study of nanocrossbars outlined some interesting connections between logic synthesis and defect tolerance. It would be interesting to use the properties of synthesis methods, Boolean functions decompositions and regularities to develop a synthesis method that can minimize the effects of lattice faults.

7.3 Publications

Part of the work presented in this thesis is published in articles and conference proceedings.

Part I:

- The first version of XORAM AM cell was designed in 2012 [60] and the first implementation of this cell was in designed in 64 nm and included in the AM06 [59]. The first version of XORAM in 28 nm is AM07 [31, 104].
- The specifications of AM08 are published in [28].
- The KOXORAM cell measurement results are published in [30] and, the characterization of the AM07 is published in [29].

- The full-custom pop-count circuit is published in [32], a former version that contains also the control logic is published in [69].
- The DCO circuit is published in [34].

Part II:

- Preliminary article on nanocrossbars are published in [105, 106].
- The internal decomposition methods use P-circuits, D-reducible functions and EP-SOP [35, 36]. The comparison between different decomposition methods is published in [37].
- The external decomposition results regarding autosymmetric functions are in [38]. The comparison between internal and external decomposition is published in [39]
- A preliminary study of nanocrossbar defectivity is published in [40].

Appendix A

AM chip specifications

A.1 Main requirements

- Bits for input word required: 16;
- Bits for internal CAM cell required: 18. The input word is 16, the other two bits are used to provide DC bits;
- Number of words per pattern: 8;
- Possibility to merge/split patterns: three options:
  - merge 2 patterns into a single 16 bit word pattern;
  - merge 4 patterns into a single 32 bit word pattern;
  - split a pattern into two 4 words patterns or one single pattern of 4 words.
- CAM energy consumption: $< 1\,\mathrm{fJ/bit}$ (including standby), if possible this specification has to be reduced as much as possible;
- Non-core power consumption: $500\,\mathrm{mW}$;
- Core minimum clock frequency: $250\,\mathrm{MHz}$;
- Max pass-through latency: $100\,\mathrm{ns}$ for AM09. ($50\,\mathrm{ns}$ for AM08);
- Minimum number of patterns: $3\times$128 Ki for AM09 (16 Ki for AM08);
- Working temperatures: $0\,^{\circ}\mathrm{C}$ to $120\,^{\circ}\mathrm{C}$;
- Quorum logic thresholds: $\{0; 1; 6; 7; 8; 9\}$;
- Data inputs/outputs compatible with LVDS18.

Aggressive goals

The AM ASIC will be considered in specifications also if these goals are not fully met. However meeting or getting closer to these goals improve performance.

- Core clock frequency: $400\,\mathrm{MHz}$ (typical and fast corners only)
- Number of patterns: 4×128 Ki for AM09.
- CAM energy consumption: $\leq 0.5\,\mathrm{fJ/bit}$.

Full Custom simulation Corners

- Worst Speed: ss, 0.9 V, 0 °C ($250\,\mathrm{MHz}$)
- Typical: tt, (1, 1.1) V, 25 °C ($400\,\mathrm{MHz}$)
- Worst Power: ff, 1,1 V, 0 °C ($400\,\mathrm{MHz}$)
- Crossed: fs, 0.9 V, 0 °C ($250\,\mathrm{MHz}$)
- Crossed: sf, 0.9 V, 0 °C ($250\,\mathrm{MHz}$)

The corner files are given by the silicon foundry and refers to the conductibility of nMOS and pMOS transistors, for example the fs corner is characterized by fast nMOS and slow pMOS transistors.

Interdisciplinary Applications

The AM chip can be used for interdisciplinary applications that require fast and efficient pattern matching.

Image Processing largely depends on pre-processing and post-processing steps.

- Direct FPGA interface: one AM chip will be connected with one FPGA
- Working as a hardware accelerator easily interfacing with PCs and HPCs (e.g. PCI express interface card)

DNA Processing compares long word.

- **Global quorum**: implemented in standard cells after the global readout tree; perform pop-count on hitmaps of a sequence of patterns then apply a global threshold;
- **Conditional writing:**

- bus by bus implemented in standard cells;
- bit by bit (masked write) made with 2 extra transistors per BL decoder.

A.1.1 Cores

AM08 will be composed of four "design-cores": The Table A.1 shows the features that will be present in each core:

Table A.1: Core features and options

Feature	Core			
	1	**2**	**3**	**4**
CAM block variable row number (64, 128, 256)				x
SRAM-like reading			x	x
full custom quorum circuit		x	x	x

The core 1 is similar to what designed in AM07, it will be placed to reduce risks if the full custom quorum does not work. The core 2 is the baseline, it contains the full custom quorum circuit. The core 3 and 4 will be used to test the new feature of SRAM-like reading. If this feature adds a power consumption lower than 5% with respect to cores 1 and 2 this new feature will be integrated in AM09. Core 4 will implement the variable row number architecture. In AM07 and AM06 we used blocks of 64 rows. However, blocks with more rows (128 and 256) can improve the memory density.

The possibility to read the bit content of CAM like SRAM will permit fast chip debug.

List of Terms

AM Associative Memory. i, 4–7, 16, 19–21, 24, 37, 107, 109

ASIC Application specific integrated circuit. 5

ATLAS A Toroidal LHC ApparatuS. ii, 5, 19, 107

BJT Bipolar Junction Transistor. 3

CAM Content Addressable Memory. 4, 20, 28

CERN European Organization for Nuclear Research. ii, 5

CMOS Complementary Metal-Oxide Semiconductor. i, ii, 3–5, 7, 11, 16

CMP Chemical Mechanical Polishing. 12, 14

D-FF Delay Flip-Flop. 23

DC don't care. 24, 25, 111

DCO Digitally Controlled Oscillator. 16, 38, 40, 41, 108

FPGA Field Programmable Gate Array. 5, 24

HEP High-Energy Physics. i, 4, 5, 19, 20

HL-LHC High Luminosity Large Hadron Collider. ii, 5

HTT Hardware Tracking for the Trigger. i, 5

IC Integrated Circuit. i, 1, 3, 4, 6, 107

ILD Inter Layer Dielectric. 12

IMPART Innovative Multi-chip system for multi-purpose PAttern Recognition Tasks. i, 5

LHC Large Hadron Collider. 19

LVBG Local Voltage Bias Generator. 38, 41

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor. 1, 3, 12

PLL Phase Locked Loop. 16, 37, 38, 40, 46

SA0 Stuck-At 0. 101–105, 109

SA1 Stuck-At 1. 101–105, 109

SR Set Reset. 23

VCO Voltage-Controlled Oscillator. 38

VLSI Very Large Scale Intergation. 20

Bibliography

[1] W. V. Quine, "The problem of simplifying truth functions," *Am. Math. Mon.*, vol. 59, no. 8, pp. 521–531, 1952. Available: http://www.jstor.org/stable/2308219

[2] W. V. Quine, "A way to simplify truth functions," *Am. Math. Mon.*, vol. 62, no. 9, pp. 627–631, 1955. Available: http://www.jstor.org/stable/2307285

[3] E. J. McCluskey, "Minimization of Boolean functions," *Bell Syst. Tech. J.*, vol. 35, no. 6, pp. 1417–1444, Nov 1956.

[4] J. S. Kilby, "Miniaturized electronic circuits," 02 1959. Available: http://www.patentlens.net/patentlens/patent/US_7062320/

[5] G. Moore, "Cramming more components onto Integrated Circuits," *Journal of Electronics*, pp. 114–117, April 1965.

[6] G. Moore, "Progress in digital integrated electronics," in *1975 International Electron Devices Meeting*, vol. 21, 1975, pp. 11–13.

[7] G. Moore, "The M0S transistor as an individual device and in integrated arrays," in *1958 IRE International Convention Record*, vol. 13, March 1965, pp. 44–52.

[8] M. Bohr, "A 30 year retrospective on Dennard's MOSFET scaling paper," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 1, pp. 11–13, Winter 2007.

[9] M. T. Bohr and I. A. Young, "CMOS scaling trends and beyond," *IEEE Micro*, vol. 37, no. 6, pp. 20–29, November 2017.

[10] D. James, "Intel's 14-nm Parts are Finally Here!" *chipworks*, 2014, accessed: December 20, 2018. Available: https://www.chipworks.com/about-chipworks/overview/blog/intel%E2%80%99s-14-nm-parts-are-finally-here

[11] "International Technology Roadmap for Semiconductors (ITRS) reports," http://www.itrs2.net, 2015, accessed: December 20, 2018.

[12] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct 1974.

[13] M. Dell'Orso and L. Ristori, "VLSI structures for track finding," *Nucl. Instrum. Methods Phys. Res. A*, vol. Nucl. Instrum. Methods Phys. Res., no. 2, pp. 436 – 440, 1989. Available: http://www.sciencedirect.com/science/article/pii/0168900289908620

[14] A. Andreani *et al.*, "The FastTracker Real Time Processor and its impact on Muon Isolation, Tau and b-Jet online selections at ATLAS," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 2, pp. 348–357, April 2012.

[15] V. Cavaliere *et al.*, "Design of a hardware track finder (Fast Tracker) for the ATLAS trigger," *J. Instrum.*, vol. 11, no. 02, p. C02056, 2016. Available: http://stacks.iop.org/1748-0221/11/i=02/a=C02056

[16] "IMPART: A Multi-Purpose chip for PAttern Recognition Tasks," http://albertostabile3.wixsite.com/impart, accessed: December 20, 2018.

[17] ATLAS Collaboration, "Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System," CERN, Tech. Rep. ATL-COM-DAQ-2017-160, 2017.

[18] The ATLAS Collaboration et al., "The ATLAS Experiment at the CERN Large Hadron Collider," *J. Instrum.*, vol. 3, no. 08, p. S08003, 2008. Available: http://stacks.iop.org/1748-0221/3/i=08/a=S08003

[19] H. Yan, H. S. Choe, S. Nam, Y. Hu, S. Das, J. F. Klemic, J. C. Ellenbogen, and C. M. Lieber, "Programmable nanowire circuits for nanoprocessors," *Nature*, vol. 470, pp. 240 EP –, Feb 2011. Available: http://dx.doi.org/10.1038/nature09749

[20] G. M. Whitesides and B. Grzybowski, "Self-assembly at all scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002. Available: http://science.sciencemag.org/content/295/5564/2418

[21] W. Lu and C. M. Lieber, "Nanoelectronics from the bottom up," *Nat. Mater.*, vol. 6, pp. 841 EP –, Nov 2007, review Article. Available: http://dx.doi.org/10.1038/nmat2028

[22] K. Ariga, M. V. Lee, T. Mori, X.-Y. Yu, and J. P. Hill, "Two-dimensional nanoarchitectonics based on self-assembly," *Adv. Colloid and Interface Sci.*, vol. 154, no. 1, pp. 20 – 29, 2010. Available: http://www.sciencedirect.com/science/article/pii/S0001868610000072

[23] Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits," *Nanotechnology*, vol. 14, no. 4, p. 462, 2003. Available: http://stacks.iop.org/0957-4484/14/i=4/a=311

[24] L. Amarú, P. E. Gaillardon, S. Mitra, and G. D. Micheli, "New Logic Synthesis as Nanotechnology Enabler," *Proc. IEEE*, vol. 103, no. 11, pp. 2168–2195, Nov 2015.

[25] M. Altun and M. D. Riedel, "Logic Synthesis for Switching Lattices," *IEEE Trans. Comput.*, vol. 61, no. 11, pp. 1588–1600, Nov 2012.

[26] G. Gange, H. Søndergaard, and P. J. Stuckey, "Synthesizing Optimal Switching Lattices," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, no. 1, pp. 6:1–6:14, Nov. 2014. Available: http://doi.acm.org/10.1145/2661632

[27] "NANOxCOMP," http://www.nanoxcomp.itu.edu.tr, accessed: December 20, 2018.

[28] A. Stabile *et al.*, "Phase-II Associative Memory ASIC Specifications," INFN, CERN; LPNHE; UNIMI, Milano, Paris, Melbourne, London, Pisa, Perugia, Bergamo, Tech. Rep. CERN-OPEN-2018-003, May 2018, preprint. Available: https://cds.cern.ch/record/2320701

[29] A. Annovi *et al.*, "Characterization of an associative memory chip in 28 nm CMOS technology," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[30] A. Annovi, L. Frontini, V. Liberali, and A. Stabile, "Design and characterization of new content addressable memory cells," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[31] A. Annovi *et al.*, "A low-power and high-density associative memory in 28 nm CMOS technology," in *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2017, pp. 1–4.

[32] L. Frontini, V. Liberali, and A. Stabile, "A very compact population count circuit for associative memories," in *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2018, pp. 1–3.

[33] L. Frontini, A. Stabile, and V. Liberali, "Population count circuits for associative memories: A comparison study," in *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2017, pp. 1–4.

[34] S. Capra, F. Crescioli, L. Frontini, M. Garci, and V. Liberali, "A digitally-controlled ring oscillator in 28 nm CMOS technology," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[35] A. Bernasconi, V. Ciriani, L. Frontini, V. Liberali, G. Trucco, and T. Villa, "Logic synthesis for switching lattices by decomposition with P-Circuits," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 423–430.

[36] A. Bernasconi, V. Ciriani, L. Frontini, and G. Trucco, "Synthesis on switching lattices of dimension-reducible boolean functions," in *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Sept 2016, pp. 1–6.

[37] A. Bernasconi, V. Ciriani, L. Frontini, V. Liberali, G. Trucco, and T. Villa, "Enhancing logic synthesis of switching lattices by generalized Shannon decomposition methods," *Microprocessors and Microsystems*, vol. 56, pp. 193–203, 2018.

[38] A. Bernasconi, V. Ciriani, L. Frontini, and G. Trucco, "Composition of Switching Lattices and Autosymmetric Boolean Function Synthesis," in *2017 Euromicro Conference on Digital System Design (DSD)*, Aug 2017, pp. 137–144.

[39] A. Bernasconi, V. Ciriani, L. Frontini, and G. Trucco, "Composition of switching lattices for regular and for decomposed functions," *Microprocessors and Microsystems*, vol. 60, pp. 207 – 218, 2018. Available: http://www.sciencedirect.com/science/article/pii/S0141933118300449

[40] M. C. Morgül, L. Frontini, I. Vatajelu, and L. Anghel, "Integrated Synthesis Methodology for Crossbar Arrays," in *IEEE NANOARCH'2018*. Athens, Greece: Academic Press, London, UK, Jul. 2018. Available: https://hal.archives-ouvertes.fr/hal-01898674

[41] K. J. Kuhn, "Moore's Law Past 32nm: Future Challenges in Device Scaling," in *2009 13th International Workshop on Computational Electronics*, May 2009, pp. 1–6.

[42] H. Wong and H. Iwai, "On the scaling issues and high-κ replacement of ultrathin gate dielectrics for nanoscale MOS transistors," *Microelectronic Engineering*, vol. 83, no. 10, pp. 1867 – 1904, 2006. Available: http://www.sciencedirect.com/science/article/pii/S016793170600253X

[43] S. H. Lo, D. A. Buchanan, Y. Taur, and W. Wang, "Quantum-mechanical modeling of electron tunneling current from the inversion layer of ultra-thin-oxide nMOSFET's," *IEEE Electron Device Lett.*, vol. 18, no. 5, pp. 209–211, May 1997.

[44] L. P. B. Lima, H. F. W. Dekkers, J. G. Lisoni, J. A. Diniz, S. Van Elshocht, and S. De Gendt, "Metal gate work function tuning by Al incorporation in TiN," *Journal of Applied Physics*, vol. 115, no. 7, p. 074504, 2014. Available: https://doi.org/10.1063/1.4866323

[45] "A Review of TSMC 28 nm Process Technology," https://www.chipworks.com/about-chipworks/overview/blog/review-tsmc-28-nm-process-technology, accessed: December 20, 2018.

[46] H. Levinson, *Principles of Lithography*, ser. SPIE Press Monograph. Society of Photo Optical, 2005.

[47] S. Chii-ming, H. Yi-yu, T. Yu-cheng, L. Hung-yueh, T. Kao-tsai, and W. Jong-bor, "Alternating phase shift mask," 12 2005. Available: http://www.freepatentsonline.com/6977127.pdf

[48] Z. Krivokapic and C. A. Spence, "Attenuated phase shift mask," 07 1999. Available: http://www.freepatentsonline.com/5928813.pdf

[49] G. Boselli, G. Trucco, and V. Liberali, "Effects of digital switching noise on analog circuits performance," in *2007 18th European Conference on Circuit Theory and Design*, Aug 2007, pp. 160–163.

[50] A. Afzali-Kusha, M. Nagata, N. K. Verghese, and D. J. Allstot, "Substrate Noise Coupling in SoC Design: Modeling, Avoidance, and Validation," *Proceedings of the IEEE*, vol. 94, no. 12, pp. 2109–2138, Dec 2006.

[51] S. Donnay and G. Gielen, *Substrate Noise Coupling in Mixed-Signal ASICs*. Springer, 2003.

[52] L. Frontini, A. Stabile, and V. Liberali, "Power Distribution Network optimization for Associative Memories," in *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2017, pp. 1–4.

[53] N. Kimura and A. Collaboration, "Atlas ftk a - very complex - custom super computer," *Journal of Physics: Conference Series*, vol. 762, no. 1, p. 012005, 2016. Available: http://stacks.iop.org/1742-6596/762/i=1/a=012005

[54] G. Apollinari, A. I. Béjar, O. Brüning, P. Fessia, M. Lamont, L. Rossi, and L. Tavian, *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1*, ser. CERN Yellow Reports: Monographs. Geneva: CERN, 2017. Available: http://cds.cern.ch/record/2284929

[55] J. Gradin, M. Mårtensson, and R. Brenner, "Comparison of two hardware-based hit filtering methods for trackers in high-pileup environments," *Journal of Instrumentation*, vol. 13, no. 04, p. P04019, 2018. Available: http://stacks.iop.org/1748-0221/13/i=04/a=P04019

[56] M. M. Del Viva, G. Punzi, and D. Benedetti, "Information and perception of meaningful patterns," *PloS one*, vol. 8, no. 7, p. e69154, 2013.

[57] P. Luciano, C. . Sotiropoulou, S. Gkaitatzis, M. Viti, S. Citraro, A. Retico, P. Giannetti, and M. Dell'Orso, "A Hardware Implementation of a Brain Inspired Filter for Image Processing," *IEEE Transactions on Nuclear Science*, vol. 64, no. 6, pp. 1374–1381, June 2017.

[58] M. A. Mirzaei, F. Crescioli, S. Viret, W. Tromeur, G. Calderini, G. Marchiori, G. Baulieu, and G. Galbit, "A Novel Associative Memory Based Architecture for Sequence Alignment," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2016, pp. 473–478.

[59] A. Annovi, M. M. Beretta, G. Calderini, F. Crescioli, L. Frontini, V. Liberali, S. Shojaii, and A. Stabile, "AM06: the Associative Memory chip for the Fast TracKer in the upgraded ATLAS detector," *J. Instrum.*, vol. 12, no. 04, p. C04013, 2017. Available: http://stacks.iop.org/1748-0221/12/i=04/a=C04013

[60] L. Frontini, S. Shojaii, A. Stabile, and V. Liberali, "A new XOR-based content addressable memory architecture," in *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*, Dec 2012, pp. 701–704.

[61] P. Fischer, "First implementation of the MEPHISTO binary readout architecture for strip detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 461, no. 1, pp. 499 – 504, 2001, 8th Pisa Meeting

on Advanced Detectors. Available: http://www.sciencedirect.com/science/article/pii/S0168900200012833

[62] A. Annovi *et al.*, "A new variable-resolution Associative Memory for high energy physics," in *2011 2nd International Conference on Advancements in Nuclear Instrumentation, Measurement Methods and their Applications*, June 2011, pp. 1–6.

[63] A. T. Do, C. Yin, K. Velayudhan, Z. C. Lee, K. S. Yeo, and T. T. Kim, "0.77 fJ/bit/search Content Addressable Memory Using Small Match Line Swing and Automated Background Checking Scheme for Variation Tolerance," *IEEE J. Solid-State Circuits*, vol. 49, no. 7, pp. 1487–1498, July 2014.

[64] M. Wieckowski, D. Sylvester, D. Blaauw, V. Chandra, S. Idgunji, C. Pietrzyk, and R. Aitken, "A black box method for stability analysis of arbitrary SRAM cell structures," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, March 2010, pp. 795–800.

[65] J. U. Horstmann, H. W. Eichel, and R. L. Coates, "Metastability behavior of CMOS ASIC flip-flops in theory and test," *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 146–157, Feb 1989.

[66] J. Lohstroh, "Static and dynamic noise margins of logic circuits," *IEEE J. Solid-State Circuits*, vol. 14, no. 3, pp. 591–598, June 1979.

[67] A. Dalalah, S. Baba, and A. Tubaishat, "New Hardware Architecture for Bit-counting," in *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, ser. ACOS'06. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2006, pp. 118–128. Available: http://dl.acm.org/citation.cfm?id=1973598.1973623

[68] V. A. Pedroni, "Compact Hamming-Comparator-based rank order filter for digital VLSI and FPGA implementations," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 2, May 2004, pp. II–585–8 Vol.2.

[69] L. Frontini, A. Stabile, and V. Liberali, "Population count circuits for Associative Memories: A comparison study," in *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2017, pp. 1–4.

[70] M. Maymandi-Nejad and M. Sachdev, "A digitally programmable delay element: design and analysis," *IEEE Trans. VLSI Syst.*, vol. 11, no. 5, pp. 871–878, Oct 2003.

[71] M. Maymandi-Nejad and M. Sachdev, "A monotonic digitally controlled delay element," *IEEE J. Solid-State Circuits*, vol. 40, no. 11, pp. 2212–2219, Nov 2005.

[72] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "On Decomposing Boolean Functions via Extended Cofactoring," in *Design Automation and Test in Europe (DATE)*, 2009, pp. 1464–1469.

[73] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Logic Synthesis by Signal-Driven Decomposition," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, K. Gulati, Ed. Springer New York, 2011, pp. 9–29.

[74] A. Bernasconi, V. Ciriani, V. Liberali, G. Trucco, and T. Villa, "Synthesis of P-Circuits for Logic Restructuring," *Integration*, vol. 45, no. 3, pp. 282–293, 2012.

[75] A. Bernasconi, V. Ciriani, and R. Cordone, "EXOR Projected Sum of Products," in *14th International Conference on Very Large Scale Integration*, 2006.

[76] A. Bernasconi, V. Ciriani, and R. Cordone, "On Projecting Sums of Products," in *11th Euromicro Conference on Digital Systems Design: Architectures, Methods and Tools*, 2008, pp. 787–794.

[77] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Projected Don't Cares," in *Euromicro Conference on Digital Systems Design (DSD12)*, 2012, pp. 57–64.

[78] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Using Flexibility in P-Circuits by Boolean Relations," *IEEE Trans. Computers*, vol. 64, no. 12, pp. 3605–3618, 2015.

[79] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Minimization of EP-SOPs via Boolean relations," in *IFIP/IEEE VLSI-SoC 2013 - International Conference on Very Large Scale Integration of System-on-Chip*, 2013, pp. 112–117.

[80] J. C. Bioch, "The Complexity of Modular Decomposition of Boolean Functions," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 1–13, 2005.

[81] V. Kravets, "Constructive Multi-Level Synthesis by Way of Functional Properties," Ph.D. dissertation, Computer Science Engineering, University of Michigan, 2001.

[82] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Microelectronic Center, User Guide, 1991.

[83] A. Bernasconi and V. Ciriani, "DRedSOP: Synthesis of a New Class of Regular Functions." in *Euromicro Conference on Digital Systems Design (DSD)*, 2006, pp. 377–384.

[84] V. Ciriani, "Synthesis of SPP -Level Logic Networks using Affine Spaces," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 10, pp. 1310–1323, 2003.

[85] A. Bernasconi and V. Ciriani, "Dimension-reducible boolean functions based on affine spaces," *ACM Trans. Design Autom. Electr. Syst.*, vol. 16, no. 2, p. 13, 2011.

[86] F. Luccio and L. Pagli, "On a New Boolean Function with Applications," *IEEE Trans. Comput.*, vol. 48, no. 3, pp. 296–310, 1999.

[87] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Fast Three-Level Logic Minimization Based on Autosymmetry," in *ACM/IEEE 39th Design Automation Conference (DAC)*, 2002, pp. 425–430.

[88] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Implicit Test of Regularity for Not Completely Specified Boolean Functions," in *IEEE/ACM 11th International Workshop on Logic & Synthesis (IWLS)*, 2002, pp. 345–350.

[89] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Three-Level Logic Minimization Based on Function Regularities," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1005–1016, 2003.

[90] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Exploiting Regularities for Boolean function synthesis," *Theory Comput. Syst.*, vol. 39, no. 4, pp. 485–501, 2006.

[91] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Synthesis of Autosymmetric Functions in a New Three-Level Form," *Theory Comput. Syst.*, vol. 42, no. 4, pp. 450–464, 2008.

[92] A. Bernasconi, V. Ciriani, and G. Trucco, "Biconditional-BDD Ordering for Autosymmetric Functions," in *2015 Euromicro Conference on Digital*

System Design, DSD 2015, Madeira, Portugal, August 26-28, 2015, 2015, pp. 211–217.

[93] P. Cohn, *Algebra Vol. 1*. John Wiley & Sons, 1981.

[94] V. Ciriani, "A New Approach to Three-Level Logic Synthesis," Computer Science Department, University of Pisa, Technical Report TR-02-03, 2002.

[95] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Addison-Wesley Publishing Company, 1993.

[96] A. Bernasconi and V. Ciriani, "Logic synthesis and testability of D-reducible functions," in *IFIP/IEEE VLSI-SoC 2010 - International Conference on Very Large Scale Integration of System-on-Chip,*, 2010, pp. 280–285.

[97] R. Bryant, "Graph Based Algorithm for Boolean Function Manipulation," *IEEE Trans. Comput.*, vol. 35, no. 9, pp. 667–691, 1986.

[98] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," in *ACM/IEEE 30th Design Automation Conference (DAC)*, 1993, pp. 272–277.

[99] A. Bernasconi, V. Ciriani, and L. Lago, "On the error resilience of ordered binary decision diagrams," *Theor. Comput. Sci.*, vol. 595, pp. 11–33, 2015.

[100] A. Bernasconi and V. Ciriani, "Index-Resilient Zero-Suppressed BDDs: Definition and Operations," *ACM Trans. Design Autom. Electr. Syst.*, vol. 21, no. 4, pp. 72:1–72:27, 2016.

[101] D. Bañeres, J. Cortadella, and M. Kishinevsky, "A Recursive Paradigm to Solve Boolean Relations," *IEEE Trans. Comput.*, vol. 58, no. 4, pp. 512–527, 2009.

[102] M. C. Morgul and M. Altun, "Logic Circuit Design with Switching Nano Arrays and Area Optimization (in Turkish)," in *Elektrik, Elektronik, Bilgisayar ve Biyomedikal Mühendisligi Sempozyumu (ELECO)*, 2014.

[103] C. Chen, H. Shih, C. Wu, C. Lin, P. Chiu, S. Sheu, and F. T. Chen, "RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 180–190, Jan 2015.

[104] A. Annovi *et al.*, "A XOR-based associative memory block in 28 nm CMOS for interdisciplinary applications," in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec 2015, pp. 392–395.

[105] D. Alexandrescu, M. Altun, L. Anghel, A. Bernasconi, V. Ciriani, L. Frontini, and M. Tahoori, "Logic synthesis and testing techniques for switching nano-crossbar arrays," *Microprocessors and Microsystems*, vol. 54, pp. 14–25, 2017.

[106] D. Alexandrescu, M. Altun, L. Anghel, A. Bernasconi, V. Ciriani, L. Frontini, and M. Tahoori, "Synthesis and performance optimization of a switching nano-crossbar computer," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 334–341.