POLITECNICO DI TORINO

**Master's Degree**
**in Electronic Engineering**

Master's Degree Thesis

# Quantization Reconstruction Based LDPC Decoders


**Supervisors**
prof. Guido Masera
prof. Guido Montorsi
dott. Vincenzo Petrolo

**Candidate**
Luca D'Elia

Academic Year 2023-2024

# Summary

Nowadays, digital communication requires extremely high standards in terms of latency for channel decoding. Therefore, different techniques are investigated to combine high frequency within limited hardware resources of FPGAs and ASICs and as a result, there is a preference for decoder with low bit width.

The Low-Density Parity-Check (LDPC) codes have found wide use in multiple implementation contexts, especially in wireless communication system for its reliability in correcting errors while maintaining a high code rate. The focus is concentrated on keep good correction capacity by using a reduced bit width, with the intention to meet the requirements of low latency and area.

Several design solutions to reduce bit width have been explored, and one in particular, which relies on the use of Quantization and Reconstruction units, has shown good results in terms of the trade-off between complexity and performances.

The work presented in this master thesis offers an in-depth study on the analysis of the Variable Node (VN) which build the LDPC decoder, in particular on the use of lookup tables (LUTs) or Thermometric blocks for the Quantization and Reconstruction block. All the results and the syntheses report are obtained using a 65 nm technology library.

The differences in the occupied area and the critical path offered by the two implementations were analyzed giving an overview on the trade-off that is observed on them when the parameters involved vary, such as internal bit width of VN, external bit width and degree. The study of Variable Node proceed by looking at the performances provided by different implementation methodologies, one with data handled sequentially, or a parallel representation with high throughput. Finally, the focus is on an complete architecture with all Variable Node instantiated according to the degree profile described by the interconnection matrix H, providing an overview of all VNs implemented at the complete decoder level.

# Contents

**List of Tables** 5

**List of Figures** 7

**1 Introduction** 11

**2 Background and Related Work** 15
2.1 Digital Transmission and Error Correction . . . 15
2.2 Basic of Error Correction - Hamming Codes . . . 16
2.3 Low-Density Parity-Check - LDPC codes . . . 18
2.4 LDPC Decoding Algorithm . . . 21
2.4.1 Hard Decision . . . 21
2.4.2 Soft Decision - Min-Sum Algorithm . . . 22
2.5 Variable Node: Reconstruction and Quantization . . . 24

**3 Model Architecture** 27
3.1 Serial Single . . . 28
3.2 Parallel Single . . . 31
3.3 LDPC WiFi codes - Degree Profile . . . 33
3.4 Serial Net . . . 35
3.5 Parallel Net . . . 37
3.6 LUT/THERMO architecture for R-Q block . . . 40
3.6.1 LUT . . . 40
3.6.2 THERMO . . . 42

**4 Experimental Setup** 47
4.1 ASIC Synthesis . . . 48
4.2 FPGA Synthesis . . . 51

**5 Experimental Results** 55
5.1 ASIC Synthesis Results . . . 55
5.1.1 Serial . . . 56
5.1.2 Parallel . . . 66
5.1.3 Trade-off between Serial and Parallel . . . 76
5.2 FPGA Synthesis Results . . . 85

5.2.1 Serial - Single . . . . . 86
5.2.2 Parallel - Single . . . . . 94
5.2.3 Trade-off between Serial Single - LUT and THERMO . . . . . 102
5.2.4 Trade-off between Parallel Single - LUT and THERMO . . . . . 105
5.2.5 Trade-off between Single LUT - Serial and Parallel . . . . . 108
5.2.6 Trade-off between Single THERMO - Serial and Parallel . . . . . 111

**6 Conclusion** 115

# List of Tables


3.1 Mapping relationship for quantization - *term_bin_DEC_3* . . . . . 44
5.1 Area ($\mu m^2$) of LUT_Q . . . . . 57
5.2 Area ($\mu m^2$) of THERMO_Q . . . . . 57
5.3 Area ($\mu m^2$) of LUT_R . . . . . 59
5.4 Area ($\mu m^2$) of THERMO_R . . . . . 59
5.5 Tot. Area ($\mu m^2$) - LUT blocks . . . . . 59
5.6 Tot. Area ($\mu m^2$) - Thermo blocks . . . . . 59
5.7 totLUTarea/totArea (%) . . . . . 60
5.8 totTHERMOarea/totArea (%) . . . . . 60
5.9 Critical Path ($ns$) LUT . . . . . 62
5.10 Critical Path ($ns$) THERMO . . . . . 62
5.11 Total Area for NET LUT . . . . . 64
5.12 Total Area for NET THERMO . . . . . 64
5.13 Critical path for NET LUT . . . . . 65
5.14 Critical path for NET THERMO . . . . . 65
5.15 Area ($\mu m^2$) of LUT_Q_par . . . . . 67
5.16 Area ($\mu m^2$) of Thermo_Q_par . . . . . 67
5.17 Area ($\mu m^2$) of LUT_R_par . . . . . 69
5.18 Area ($\mu m^2$) of Thermo_R_par . . . . . 69
5.19 Tot.Area ($\mu m^2$) - LUT_par . . . . . 70
5.20 Tot.Area ($\mu m^2$) - Thermo_par . . . . . 70
5.21 totLUTarea/totArea (%) . . . . . 70
5.22 totTHERMOarea/totArea (%) . . . . . 70
5.23 Critical Path ($ns$) LUT_par . . . . . 72
5.24 Critical Path ($ns$) Thermo_par . . . . . 72
5.25 Total Area for NET LUT_par . . . . . 73
5.26 Total Area for NET Thermo_par . . . . . 74
5.27 Crit-path for NET LUT_par . . . . . 75
5.28 Crit-path for NET Thermo_par . . . . . 75
5.29 Latency ($ns$) Serial LUT . . . . . 81
5.30 Latency ($ns$) Serial THERMO . . . . . 81
5.31 ALMs for Synthesis . . . . . 86
5.32 ALMs for Fitter . . . . . 86
5.33 ALUTs for Synthesis . . . . . 87

5.34 ALUTs for Fitter . . . 87
5.35 Logic Reg for Synthesis . . . 88
5.36 Logic Reg for Fitter . . . 88
5.37 $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model . . . 89
5.38 ALMs for Synthesis . . . 90
5.39 ALMs for Fitter . . . 90
5.40 ALUTs for Synthesis . . . 91
5.41 ALUTs for Fitter . . . 91
5.42 Logic Reg for Synthesis . . . 92
5.43 Logic Reg for Fitter . . . 92
5.44 $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model . . . 93
5.45 ALMs for Synthesis . . . 94
5.46 ALMs for Fitter . . . 94
5.47 ALUTs for Synthesis . . . 95
5.48 ALUTs for Fitter . . . 95
5.49 Logic Reg for Synthesis . . . 96
5.50 Logic Reg for Fitter . . . 96
5.51 $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model . . . 97
5.52 ALMs for Synthesis . . . 98
5.53 ALMs for Fitter . . . 98
5.54 ALUTs for Synthesis . . . 99
5.55 ALUTs for Fitter . . . 99
5.56 Logic Reg for Synthesis . . . 100
5.57 Logic Reg for Fitter . . . 100
5.58 $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model . . . 101

# List of Figures


2.1 Block scheme of a digital transmission . . . . . 15
2.2 Error correction using two redundant copies . . . . . 16
2.3 Correction of an erasure using a single parity bit . . . . . 17
2.4 Tanner Graph of the interconnection matrix H . . . . . 20
2.5 Generalized RCQ Architecture . . . . . 25
3.1 Possible architectures and combinations . . . . . 27
3.2 Block diagram of the Serial Single Variable Node . . . . . 29
3.3 Control Unit of the Serial Single Variable Node . . . . . 30
3.4 Block diagram of the Parallel Single Variable Node (Degree 2) . . . . . 32
3.5 Control Unit of the Parallel Single Variable Node . . . . . 33
3.6 H interconnection matrix [7] . . . . . 34
3.7 Degree profile. VN in blue, CN in red [7] . . . . . 34
3.8 VN Net with total degree distribution . . . . . 36
3.9 Block diagram of the Parallel Single Variable Node (Degree 3) . . . . . 38
3.10 Working principle of Quantization LUT and Reconstruction LUT . . . . . 41
3.11 Working principle of Quantization THERMO - *Comparator* . . . . . 43
3.12 Working principle of Reconstruction THERMO . . . . . 45
4.1 Tree diagram of code organization . . . . . 48
4.2 ALM Block Diagram for Cyclone V Devices [4] . . . . . 52
5.1 Relationship of Area between LUT and THERMO Quantization block . . . 57
5.2 Relationship of Area between LUT and THERMO Reconstruction block . 58
5.3 Relationship of Total Area between VN using LUT (left) and THERMO (right) blocks . . . . . 60
5.4 Relationship of the Critical path period ($ns$) between LUT and THERMO 61
5.5 Relationship of Total Area between NET VN using LUT (blue) and THERMO (orange) blocks . . . . . 63
5.6 Relationship of the Critical Path period ($ns$) between LUT (blue) and THERMO (orange) . . . . . 65
5.7 Relationship of Area between LUT (left) and THERMO (right) Parallel Quantization block . . . . . 67
5.8 Relationship of Area between LUT (left) and THERMO (right) Parallel Reconstruction block . . . . . 68
5.9 Relationship of Parallel Total Area between VN using LUT (left) and THERMO (right) blocks . . . . . 69

5.10 Relationship of the Critical path period ($ns$) between LUT and THERMO 71
5.11 Relationship of Total Area between NET VN using LUT (blue) and THERMO (orange) blocks . . . 73
5.12 Relationship of the Critical Path period ($ns$) between LUT (blue) and THERMO (orange) . . . 75
5.13 Relationship of Total Area between Serial (left) and Parallel (right) configurations - LUT block . . . 77
5.14 Relationship of Total Area between Serial (left) and Parallel (right) configurations - THERMO block . . . 77
5.15 Degree profile for Rate = 0.75. VN in blue, CN in red [7] . . . 78
5.16 Relationship of the Critical path period ($ns$) between Serial (left) and Parallel (right) - LUT block . . . 79
5.17 Relationship of the Critical path period ($ns$) between Serial (left) and Parallel (right) - THERMO block . . . 79
5.18 Relationship of Latency ($ns$) between Parallel (left) and Serial (right) - LUT block . . . 80
5.19 Relationship of Latency ($ns$) between Parallel (left) and Serial (right) - THERMO block . . . 81
5.20 Relationship of Total NET Area between Serial (blue) and Parallel (orange) configurations - LUT block . . . 82
5.21 Relationship of Total NET Area between Serial (blue) and Parallel (orange) configurations - THERMO block . . . 83
5.22 Relationship of NET Critical path period ($ns$) between Serial (blue) and Parallel (orange) - LUT block . . . 83
5.23 Relationship of NET Critical path period ($ns$) between Serial (blue) and Parallel (orange) - THERMO block . . . 84
5.24 Relationship of ALMs - Synthesis (left) and Fitter (right) - LUT block . . 86
5.25 Relationship of ALUTs - Synthesis (left) and Fitter (right) - LUT block . . 87
5.26 Relationship of Logic Reg - Synthesis (left) and Fitter (right) - LUT block 88
5.27 Timing Analyzer - $F_{max}$ Summary (MHz) - LUT block . . . 89
5.28 Relationship of ALMs - Synthesis (left) and Fitter (right) - THERMO block 90
5.29 Relationship of ALUTs - Synthesis (left) and Fitter (right) - THERMO block 91
5.30 Relationship of Logic Reg - Synthesis (left) and Fitter (right) - THERMO block . . . 92
5.31 Timing Analyzer - $F_{max}$ Summary (MHz) - THERMO block . . . 93
5.32 Relationship of ALMs - Synthesis (left) and Fitter (right) - LUT block . . 94
5.33 Relationship of ALUTs - Synthesis (left) and Fitter (right) - LUT block . . 95
5.34 Relationship of Logic Reg - Synthesis (left) and Fitter (right) - LUT block 96
5.35 Timing Analyzer - $F_{max}$ Summary (MHz) - LUT block . . . 97
5.36 Relationship of ALMs - Synthesis (left) and Fitter (right) - THERMO block 98
5.37 Relationship of ALUTs - Synthesis (left) and Fitter (right) - THERMO block 99
5.38 Relationship of Logic Reg - Synthesis (left) and Fitter (right) - THERMO block . . . 100
5.39 Timing Analyzer - $F_{max}$ Summary (MHz) - THERMO block . . . 101
5.40 ALMs - Serial Single - LUT (left) and THERMO (right) . . . 102

5.41 ALUTs - Serial Single - LUT (left) and THERMO (right) . . . . . 103
5.42 Logic Register - Serial Single - LUT (left) and THERMO (right) . . . . . 103
5.43 Timing Analyzer - Serial Single - LUT (left) and THERMO (right) . . . . 104
5.44 ALMs - Parallel Single - LUT (left) and THERMO (right) . . . . . 105
5.45 ALUTs - Parallel Single - LUT (left) and THERMO (right) . . . . . 106
5.46 Logic Register - Parallel Single - LUT (left) and THERMO (right) . . . . . 106
5.47 Timing Analyzer - Parallel Single - LUT (left) and THERMO (right) . . . 107
5.48 ALMs - Single LUT - Serial (left) and Parallel (right) . . . . . 108
5.49 ALUTs - Single LUT - Serial (left) and Parallel (right) . . . . . 109
5.50 Logic Register - Single LUT - Serial (left) and Parallel (right) . . . . . 109
5.51 Timing Analyzer - Single LUT - Serial (left) and Parallel (right) . . . . . 110
5.52 ALMs - Single THERMO - Serial (left) and Parallel (right) . . . . . 111
5.53 ALUTs - Single THERMO - Serial (left) and Parallel (right) . . . . . 112
5.54 Logic Register - Single THERMO - Serial (left) and Parallel (right) . . . . 112
5.55 Timing Analyzer - Single THERMO - Serial (left) and Parallel (right) . . . 113

# Chapter 1

# Introduction

In recent decades, digital communications have assumed a leading role in the world, bringing much attention to the study of different systems that would provide an appropriate response to the increasing demand for high-speed communication services and information capabilities. In particular, there has been a development of wireless communications, also due to new transmission technologies and technological innovations in the field of cellular network and beyond, leading to ever higher standards in the development of architectures that can handle high throughput while maintaining low latency.

To achieve this goal the evolution in the field of error correction has made many steps forward, useful not only to the correct transmission of information, but also to the increase of the number of data sent per unit time.

The transmission of information involves the sending of messages composed of a sequence of bits within a channel that can be more or less noisy. Having a bit flip or an uncertainty about its value leads to an incorrect reading and therefore an error in the communication of information. An error recognition and correction algorithm is required to restore the original data.

There are different types of algorithms that perform this function, differentiated according to the characteristics they must possess depending on the transmission channel in which the data are transmitted, or the bit-rate value they must maintain. The Hamming code was of great importance, which was able to recognize and correct single bit errors [6]. In the case of several bits involved, they were only recognized but not corrected.

The simplest strategy to correct any kind of error is to use redundant copies of the information, so that, in case one of them is damaged, it is enough to observe the others. The introduction of redundancy, however, is expensive, since more than 50% of the information bandwidth is used for redundancy, which does not offer the guarantee of reconstructing the original data. The Hamming code instead introduces parity control [6]: redundant bits are introduced that identify the parity of a subset of the information by being able to recognize the position of the error and finally restore the original data, but in the case of a greater number of errors, they will not be corrected.

In 1963, Robert Gallager introduced the Low Density Parity Check (LDPC) codes [5], which allow the correction of more bits and provide better communication performance. This approach is widely used in cellular networks but has also found widespread use in the field of data storage.
The goal is to use parity bits for intersecting subsets of data, allowing greater reliability and correction speed for messages with high bit-widths, while maintaining a code-rate (the ratio between the bits that constitute the information and the total bits transmitted) reduced. This algorithm, however, required computational costs too high for the technology of the time due to its iterative decoding, leading it to be ignored until their rediscovery in the 90s.

Using a single parity bit only fixes one deletion, in case of multiple errors, the code fails due to lack of information. For this reason it is possible to decrease the probability of failure by dividing the message into smaller subsets related to different parity bits. The parity check is done by the Check Node (CN).
The CNs receive data from Variable Nodes (VNs) and send the results back to them. The Variable Nodes have the scope to update the original data after the operation of correction has happened, with the objective to go back to the initial message after a fixed number of iterations.

In order to meet the requirements introduced by the new technologies in terms of low latency and occupied area, we focused on the study of solutions that could provide excellent identification and correction performance while maintaining short bit-width.

In this context, the main topic of this master's thesis arises: The study of the RCQ (Reconstruction-Computation-Quantization) paradigm applied to the VN for decoding LDPC codes that uses quantization and reconstruction blocks to reduce the bit-width of the data, but at the same time allows to obtain a good Frame Error Rate (FER) also with low precision messages.

The study focuses primarily on the development of the VN structure described in VHDL language in two types of configuration: Serial, which accepts input data one after the other, and Parallel, whose data enter all at the same time concurrently. After that, the VN model was expanded to study both the performance of a single VN and a more complex one, called *Net*, at the level of the complete decoder that includes a number of instances based on the parameter N (number of VNs) chosen outlined by the interconnect matrix H, which is necessary to define the connections between VN and CN.

Finally, two different methods of Quantization and Reconstruction have been defined in order to observe the performance in terms of area and critical path for all the above configurations. The first uses lookup tables (LUT) as unique association of a label to the input data, the second uses comparators that follow a thermometric code [3], called THERMO to make the association.
The synthesis for all these cases have been explored both on silicon for an ASIC (Application Specific Integrated Circuit), and on FPGA (Field Programmable Gate Array).

The chapters contained in this work are shown below in their organization:

- **Chapter 2 - Background and Related Work**: This chapter introduces the concept of error correction within digital transmissions. The decoding process is analyzed, with a particular focus on the LDPC algorithm and its operation, illustrating its algorithm. Subsequently, attention is directed to the concept of quantization and reconstruction, highlighting the benefits introduced by this structure.
- **Chapter 3 - Model Architecture**: This chapter presents all the structures of Variable Node that have been implemented and how they are realized in hardware. The functionalities of each module are specified, and differences between implementations are shown, with particular attention to serial and parallel architectures. Furthermore, the distribution of degrees in the complete VN is highlighted, considering all instances defined by the interconnection matrix H. Finally, the design and algorithmic differences of some quantization and reconstruction methods are examined, including the use of lookup tables (LUT) and a component called THERMO.
- **Chapter 4 - Experimental Setup**: This chapter specifies the software used to perform synthesis, the configurations adopted, and the scripts used to automate the process.
- **Chapter 5 - Experimental Results**: This chapter presents all the results obtained from synthesis, both for ASIC on silicon and FPGA, presenting the data through tables and graphs. Comparisons and evaluations are made among the results, carefully examining all possible trade-offs between all possible structures.
- **Chapter 6 - Conclusion**: This chapter summarizes the main results and relevant observations obtained from the conducted study, also introducing possible ideas for future research.

# Chapter 2

# Background and Related Work

## 2.1 Digital Transmission and Error Correction

Over time we have seen an increase in the standards of quantity and quality of information transmitted, bringing an evolution of technologies in the field of telecommunications. An attempt has been made to extend the number of data transmitted within the medium, for this reason the same importance has been given to the concept of error correction, which pursues the objective of increasing the signal strength over the noise. When we want to transmit a message, it is sent via a broadcast channel, which will always be affected by noise. The ratio between the strength of the signal sent and that of the noise present in the band is called signal-to-noise ratio (SNRr), and it is a common goal to increase its value.

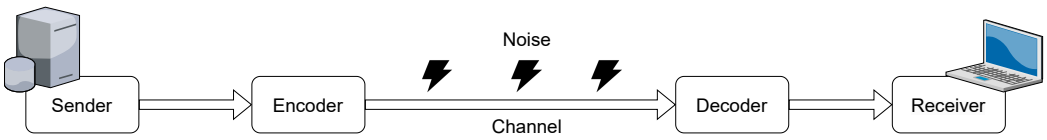


Figure 2.1: Block scheme of a digital transmission

To do so, the study on the correction of possible errors that the information may suffer during their path from the start point to the end point has been developed. The transmission channel is characterized by the band, that is the range of frequencies manageable by the medium, and by the noise that affects the channel itself that reduces the quality of the signal that travels inside. Defined these characteristics, it is possible to calculate the

capacity of the channel indicated by Shannon [14] that specifies the maximum rate called *Shannon limit* obtainable, in order to be able to send a data with zero error.

In this scenario, the concept of Forward Error Correction (FEC) has emerged, that is a set of codes and techniques that allow the recognition and correction of errors within data transmitted in a more or less noisy channel.

There have been several codes that have developed over time, starting with Hamming codes that can correct errors with a single bit, up to LDPC codes that allow a more robust correction. These technologies have spread and are still used mainly in wireless connections such as WiFi and the latest generation connections for cellular network such as 5G.

## 2.2 Basic of Error Correction - Hamming Codes

In order to correct an error, the information will be transmitted along with other copies of the data, in this way, the decoder that will read the receiving file, will have to compare the data with its copies to understand if changes were made due to noise. This concept is known as *Repetition-Code* and is based on redundancy of information and turns out to be very simple and fast during the decoding.

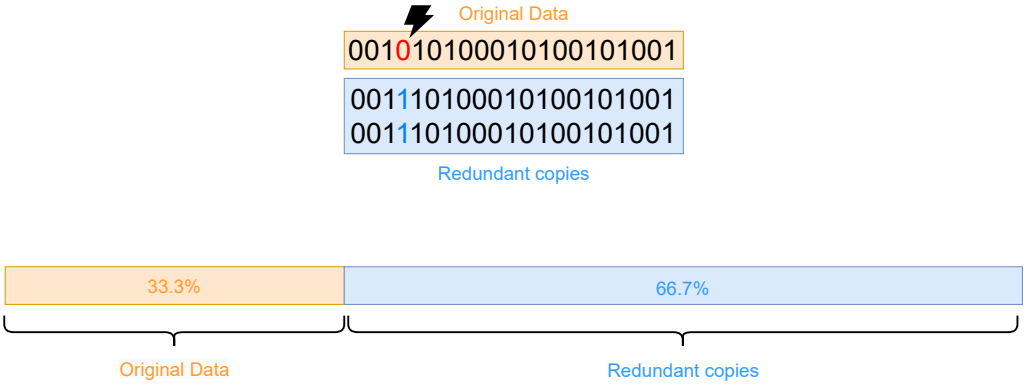


Figure 2.2: Error correction using two redundant copies

The main problem with this solution is that, for a data repeated two times, 66.7% of the bandwidth is used to send redundant copies of the information, and despite this there is also no guarantee to correct the error, since redundant copies are themselves subject to errors, and therefore are not reliable to make a correction if more than one bit is flipped.

To highlight the efficiency of this method is introduced the *Code-Rate*, defined as the ratio between the number of bits of the message and the number of bits transmitted. A high number of transmitted bits indicates a very high redundancy that leads to a greater security on the integrity of the data, but at the same time a level of Code-Rate much lower and therefore very expensive.

$$CodeRate = \frac{\#message\_bits}{\#trasmitted\_bits} \tag{2.1}$$

Having a low Code-Rate greatly complicates the design of the encoder and decoder due to the large number of bits to be transmitted unlike the actual length of the message. The goal is therefore to find a balanced strategy that provides security in restoring information, which ensures a high code-rate and therefore the use of a limited number of redundancy bits. This leads to simpler encoders and decoders that can perform operations quickly.

With the Repetition-Code, each bit is protected by its exact copy. To decrease the number of redundant bits Richard Hamming in 1940 introduced parity bits [6] that can protect a set of bits by observing the number of bits set to '1'. The encoder sets the value of the parity bit to '1' if the number of bits inside the information is even, '0' if odd. If an error is identified, the decoder will compare the number of '1' of the information with the value of the parity bit, in this way is possible to know the value of the uncertain bit. With this strategy, the Code-Rate increases considerably as the number of bits transmitted is near to the number of bits carrying the information. A numerical example is shown in figure 2.3.

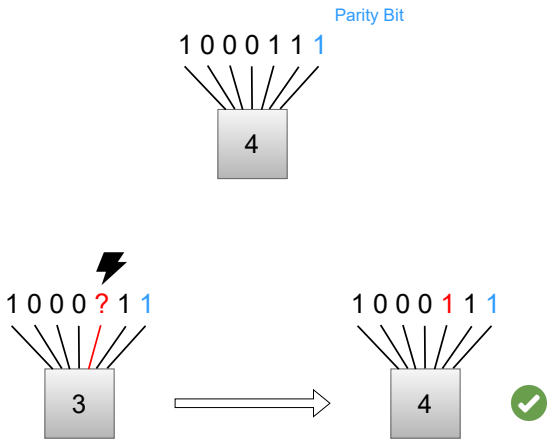


Figure 2.3: Correction of an erasure using a single parity bit

In this way, only one bit can be corrected due to the lack of information in case of double deletion. Is possible to add another parity bit and divide the information into small intervals, called *Check-Set* each linked to its own redundant bit, allowing you to decrease the probability of error. However, the problem is not completely solved, as we will still not have the possibility of correction in case cancellations are present within the same Check-Set, but we have reduced the probability that this will happen, by study smaller Check-sets.

The solution to get around the problem is to study overlapping Check-sets that are identified by several parity bits, where each of these covers more bits of the message. This allows to always identify and correct the error because is possible to have more parity bits that also observe other check-sets having a complete picture on the structure of the original data only if the decoding starts from check-set that has only one error and not more, otherwise the correction is impossible. In this case there is no longer an instant correction, but a decoding that takes place in multiple iterations by questioning the different parity bits for the different check-sets.

The main problem arises for very long data transmissions that involve the presence of many check-sets and therefore of many overlapping interconnections. This leads to multiple iterations in the corrections that take a long time to execute. The idea is therefore to find codes that can handle a large number of overlapping check-sets for very long messages to transmit, and at the same time carry out the correction iterations quickly maintaining a high code-rate. The LDPC codes are of great importance within this scenario, successfully responding to this request.

## 2.3 Low-Density Parity-Check - LDPC codes

LDPC codes were introduced in 1963 by Robert G. Gallager [5] as an improvement to the problem of error correction when transmitting information on a noise-affected channel. For the theory, these codes improved the decoding performance maintaining a high value of Code-Rate, also solving the problems explained in the section previously introduced by Hamming codes [6], but they were ignored until their rediscovery in 1996 because of the impossibility of realization in practice due to the technology of the time not suitable and too expensive to implement.

The fundamental ideas were to reduce the length of the check-sets so as to obtain an overlap of them large enough to maintain effective protection against errors, but not so large as to make the correction excessively time consuming and complex in terms of time and calculations. This concept is expressed by density, hence the term LDPC. Finally, to improve corrections and have greater reliability, it was observed that it is more effective to connect check-sets in a random and not sequential way, involving between the check-sets also parity bits used for decoding, resulting in a more solid structure. These links are described by the interconnection matrix H (example in [11]) as the one shown below:

$$H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

While the linear system that describe the matrix is represented like this:

$$\begin{aligned} x_1 + x_3 + x_4 + x_7 &= 0 \\ x_0 + x_1 + x_2 + x_5 &= 0 \\ x_2 + x_5 + x_6 + x_7 &= 0 \\ x_0 + x_3 + x_4 + x_6 &= 0 \end{aligned} \tag{2.2}$$

The matrix $n \times m$, in this case (8, 4), shows the connections that exist between two distinct blocks essential for decoding the LDPC code: the Variable-Node (VN) that contain the bits of the code-word, and the Check-Node (CN) that identify the parity controls. The number of rows in the matrix is the CN number, while the columns the VN number. The connection exists where the value '1' is present, while the number of them within the line corresponds to the density of the CN that defines the number of connected VN for each of them to perform the control. In fact, a low density code is called this way if the number of '1' in each row $W_r$ is much smaller than the number of rows $m$ ($W_r << m$), and the same must be said for the number of '1' in each column $W_c$, such that $W_c << n$ number of columns.

For a better visualization of the connections and to have a graphical representation of the matrix H, comes in aid the diagram designed by Tanner that bears his name [16]. This is shown in figure 2.4.

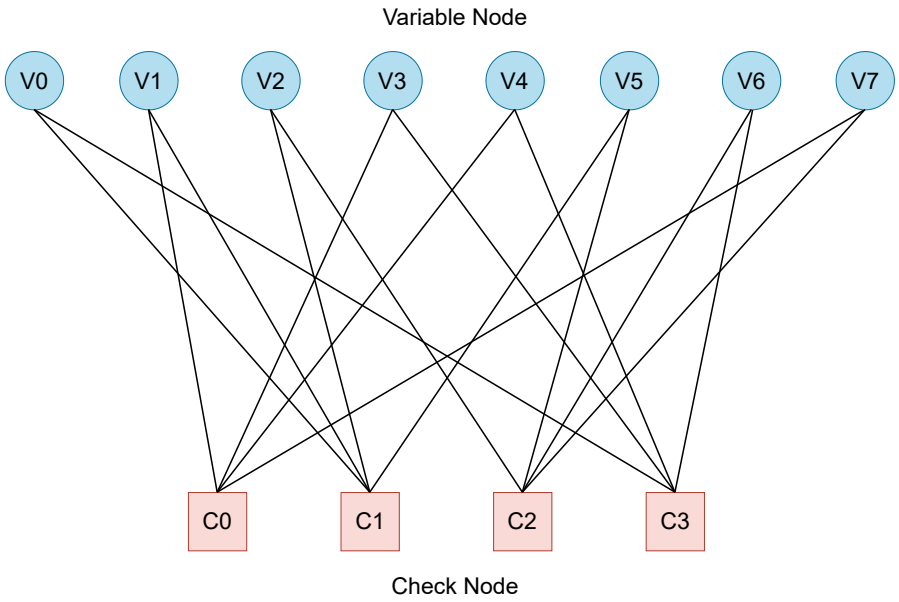


Figure 2.4: Tanner Graph of the interconnection matrix H

## 2.4 LDPC Decoding Algorithm

The decoding algorithm used carries several names such as *Belief Propagation* [9] [12], *Min-Sum Product* [15] [13] or *Message Passing* [10] but they all represent the same sequences of operations. In order to understand its functioning, a simplified version of the algorithm called *Hard Decision* is first described [15], which is not optimal to perform the decoding operation but allows to easily understand the main steps. Then it will be shown what is called *Soft Decision* that instead focuses on the concept of *Belief Propagation* which is optimal for decoding and more complete [13].

### 2.4.1 Hard Decision

It is supposed to have received a code-word composed of a string of zeros and ones, with the possibility of containing an error due to the flip of a digit.

The main steps are the following:

1. All VN send their information through the arcs to the CN to which they are connected. In this first step the only data that is sent corresponds to the code-word data that the VN see as correct.

$$CN_0 : c_1 \to 1, \quad c_3 \to 1, \quad c_4 \to 0, \quad c_7 \to 1 \quad Received \tag{2.3}$$
$$: 0 \to c_1, \quad 0 \to c_3, \quad 1 \to c_4, \quad 0 \to c_7 \quad Trasmitted \tag{2.4}$$
$$CN_3 : c_0 \to 1, \quad c_3 \to 1, \quad c_4 \to 0, \quad c_6 \to 0 \quad Received \tag{2.5}$$
$$: 1 \to c_0, \quad 1 \to c_3, \quad 0 \to c_4, \quad 0 \to c_6 \quad Trasmitted \tag{2.6}$$

   This must be $= 0$ for the parity condition.

2. At this point, each CN receives input data from the VN and calculates the resulting response in order to respect the check-set parity equation. If all equations are solved and all errors are corrected, the decoding operation ends at this point, otherwise it continues to the next iteration. For this reason, a threshold is set to the iterations that, once reached, interrupts the calculation loop.

3. VN receive processed messages from CN and update the data, make the decision on it. In this example the decision is taken for majority observing therefore the information coming from the connected CN and the original data already present in the VN.

$$VN_4 : \quad Receive \to 0; \quad CN_0 = 1; \quad CN_3 = 0; \quad DECISION \longrightarrow 0 \tag{2.7}$$

4. Repeat the loop back to step 2.

### 2.4.2 Soft Decision - Min-Sum Algorithm

This method corrects errors using the concept of probability of the data sent. Instead of observing a majority as in the *Hard* study, in this case the CN calculate the minimum value among those that get in input (Min-Sum algorithm). We then observe the sum of all the processed values coming from the CN and we evaluate the sign in opposition to the original value in order to understand if the flip has happened or not and to carry out therefore the eventual correction.

The hardware studied in this thesis has been described in VHDL starting from the study of the algorithm shown in the paper [13]. For this reason and to better understand the sequence of operations, the main steps of the *Min-Sum Decoding* are as follows:

1. **Initialization**: We introduce notations that will be used within formulas.

   - $L_n$ : A priori information of VN
   - $\overline{L_n}$ : A posteriori information of VN
   - $E_{m,n}$ : message from CN to VN
   - $F_{n,m}$ : message from VN to CN

   We set the following initializations as the first operation:

   - $L_n = -r_n$, a priori information
   - $F_{n,m} = L_n$, initialization of the data to be transmitted to the CN.

2. **Horizontal Step**: Check-Node processing of Sum-Product algorithm

$$E_{m,n} = log \frac{1 + \prod_{n' \in N(m) \backslash n} tanh(\frac{F_{n',m}}{2})}{1 - \prod_{n' \in N(m) \backslash n} tanh(\frac{F_{n',m}}{2})} \tag{2.8}$$

   Performing the tangent, however, is a very expensive and time-consuming operation of the sum-product algorithm, for this reason a simplification is used that allows us to transform the multiplication factor into a search for the minimum, hence the name Min-Sum. We can therefore proceed in this way using the relationship:

$$2\, tanh^{-1} p = log \frac{1+p}{1-p} \tag{2.9}$$

In this way the equation 2.8 can be rewritten as follows:

$$E_{m,n} = 2\,tanh^{-1} \prod_{n' \in N(m)\backslash n} tanh(\frac{F_{n',m}}{2}) \tag{2.10}$$

Which we can change again by explaining the sign:

$$E_{m,n} = 2\,tanh^{-1} \prod_{n' \in N(m)\backslash n} sgn(F_{n',m}) \prod_{n' \in N(m)\backslash n} tanh(\frac{|F_{n',m}|}{2}) \tag{2.11}$$

$$E_{m,n} = \prod_{n' \in N(m)\backslash n} sgn(F_{n',m})\, 2\,tanh^{-1} \prod_{n' \in N(m)\backslash n} tanh(\frac{|F_{n',m}|}{2}) \tag{2.12}$$

Using the min-sum algorithm we simplify the calculations of the 2.12 because it is converted the tangent product in the search for the minimum since the equation corresponds to the recognition of the smallest $F_{n',m}$.

For this reason is possible to write:

$$E_{m,n} = \prod_{n' \in N(m)\backslash n} sgn(F_{n',m}) \min_{n' \in N(m)\backslash n} |F_{n',m}| \tag{2.13}$$

3. **Vertical Step**: We now evaluate the information a posteriori $\overline{L_n}$: we calculate the total sum of all the values coming from the CN $E_{m,n}$, added further with the value of the VN a priori $L_n$:

$$\overline{L_n} = L_n + \sum_{m \in M(n)} E_{m,n} \tag{2.14}$$

And finally, from the total sum, the value of the minimum of that iteration is subtracted:

$$F_{n,m} = \overline{L_n} + \sum_{m' \in M(n) \setminus m} E_{m,n} \tag{2.15}$$

4. **Decoding**: At this point we check the sign of the total sum to see if is need to make the correction or not according to the following rule:

$$if \quad \overline{L_n} > 0, \quad \overline{c_n} = 0, \quad else \quad \overline{c_n} = 1 \tag{2.16}$$

If we get the $\overline{c_n} = 0$ then the algorithm stops, otherwise it continues processing other iterations until it reaches the maximum limit or a threshold set to exit the loop. Decoding performance increases with an increasing number of iterations, but it also increases the computational cost and resources used.

## 2.5 Variable Node: Reconstruction and Quantization

LDPC decoders have found wide use for their solid decoding capabilities as seen until now, managing to maintain a high code-rate. At the same time they were subject to extensive studies on maintaining high correction performance, but using a reduced bit-width. This is because decoders with reduced data lengths are preferable given the limited resources in terms of area of hardware components of ASIC and especially FPGA, but also to improve processing speed and, meeting the latency requirements imposed by communication standards.

At the same time, however, using messages with just a few bits degrades decoding performance.
It is here that the LDPC decoders using quantized messages have become important: messages starting from the VN to go to the CN are quantized, meaning that a representative label of the data is associated on a smaller number of bits. This allows faster decoding by the CN and at the same time saves resources, resulting in smaller component areas. The data that start from the CN and arrive at the VN meet the inverse operation: the result produced by the algorithm in the CN is reconstructed in its original value following the same laws of quantization. It has been observed that this method leads to excellent decoding performance, even using reduced bit-width.

In this thesis we will show the results of area and latency of different configurations that use the method of Quantization and Reconstruction observing the differences and providing the understanding of which architecture is better depending on the specific application

desired. The CN focuses on the implementation of the Min-Sum algorithm, and in this work is not taken into account for the study of R-Q effects.

The components that perform these operations are the lookup tables (LUT) which will be compared with blocks called THERMO that perform quantization following a thermometric conversion. These objects are used to replace the complex mathematical operations within VN.

The VN performs the visible operations in step 3 and step 4 of the algorithm shown in the section "Soft Decision - Min-Sum Algorithm". This corresponds to the Computation phase, which is preceded by the Reconstruction of the message from the CN and followed by the Quantization of the message to be sent to the CN. This defines the Reconstruction-Computation-Quantization (RCQ) paradigm [19] [18] that create the structure of the VN and can be displayed schematically as shown in figure 2.5.

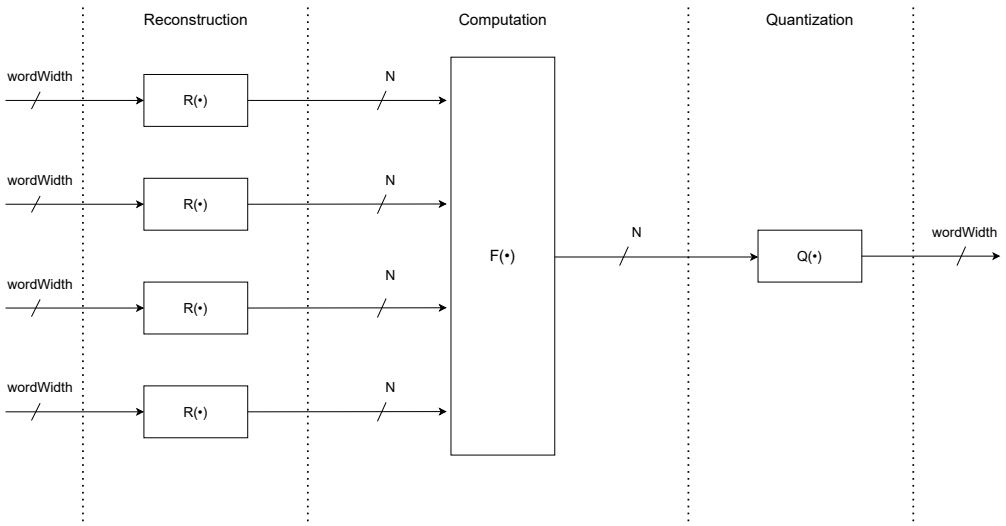


Figure 2.5: Generalized RCQ Architecture

The structure consists of three modules:

- **Reconstruction block**: takes the data sent by the CN, with which the VN is connected, on a bit-width equal to $wordWidth$ and reconstructs the message on $N$ bit such that $N > wordWidth$.
- **Computation block**: performs the operations described in the algorithm mentioned in the section above, corresponds to the internal structure of the VN.
- **Quantization block**: do the quantization of the internal messages of the structure described on N bits in labels on $wordWidth$ bits such that $wordWidth < N$.

It is important to specify that, if you use data with limited bit-widths, the use of a uniform quantization type would lead to a deterioration in performance. This means that the data all have the same resolution for each quantization step.
For this reason, LUT and THERMO have been described following a non-uniform quantization that provides a better resolution for low value data, while being more approximate for high values. This increases performance even for data with limited bit-widths. Moreover, by performing the minimum operations, for subsequent iterations the results will tend to be of low value and therefore it makes more sense to pursue this path. This idea will be discussed in more detail in the next chapter.

Finally, it should be considered that to carry out R-Q operations as the main idea is the use of LUT, but they have an intensive use of resources as they are very expensive memories that depend heavily on the number of input bits to them, in particular the number of bits of N. This prevents its use in the case that the resources are very limited as in the case of FPGAs or even when the number of iterations required by the algorithm is too high. Another prohibitive case is described by the use of flooding-schedule [8] which, instead of reusing blocks, each iteration is carried out on a different block. This means duplicating the hardware for each iteration and even more affecting the cost in the area due to repeating LUT.

All these issues will be addressed in the next chapters, where we will also observe the differences and any improvements that the use of a block other than LUT, such as THERMO, can bring to the general architecture.

# Chapter 3

# Model Architecture

The key principle of this thesis focuses on the in-depth study of the performance of the Variable Node of a LDPC decoder using different implementation structures at the hardware level.

The general problem we aim to address is to relax the cost of the overall structure of the decoder, so as to decrease the required area and latency of operations. In this case we want to study alternative architectures for the VN in order to find improvements that can project the advantages also at a macroscopic level.

The architectures studied can be summarized through the following graph that we use to offer greater understanding of the analysis process:

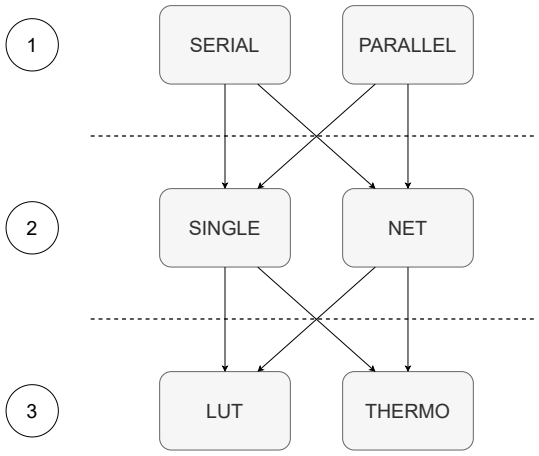


Figure 3.1: Possible architectures and combinations

The structures studied are different and applied at various levels of the VN:

1. **Serial/Parallel**: The two structures differ in how they handle incoming data to the VN. The *Serial* structure takes and manage input data serially, one at a time, while the *Parallel* structure duplicates the hardware to retrieve the data simultaneously and perform the calculations in a combinatorial way.

2. **Single/Net**: Unlike the *Single*, where the structure of the VN is studied individually, the *Net* structure identifies a complete Variable Node at the level of the Decoder with a number of single internal VN equal to the block-size N (in this case N = 1296). We study the H matrix to observe the distribution of degrees and the total number of connections based on the imposed rate R (in this case R = 0.5, so 648 CN).

3. **LUT/THERMO**: These two structures are specific to quantization and reconstruction blocks and define two different calculation strategies. The way they have been described allows them to be implemented in any configuration shown above, thus ensuring the modularity of the device.

The Single and Net architectures can contain both LUT and THERMO quantization and reconstruction blocks. Both cases will be studied, specifying in the results section the type of architecture combination being analyzed. In the same way the Serial and Parallel structures can be described both as Single and Net types.

In the following sections, we will illustrate the main hardware design choices, which have been implemented in VHDL language. They will then be tested for various combinations of parameters in the chapter dedicated to results, trying all possible combinations of the architectures described below.

## 3.1 Serial Single

The serial architecture [17] uses a more complex configuration for the data management. The structure involves memory elements such as register-file and counter to properly manage the sequencing of operations. However, it is easier from the point of view of quantization and reconstruction given the fact that this blocks are reused for all the input data whose quantity depends on the degree value.
The schematic of this VN is represented in figure 3.2.

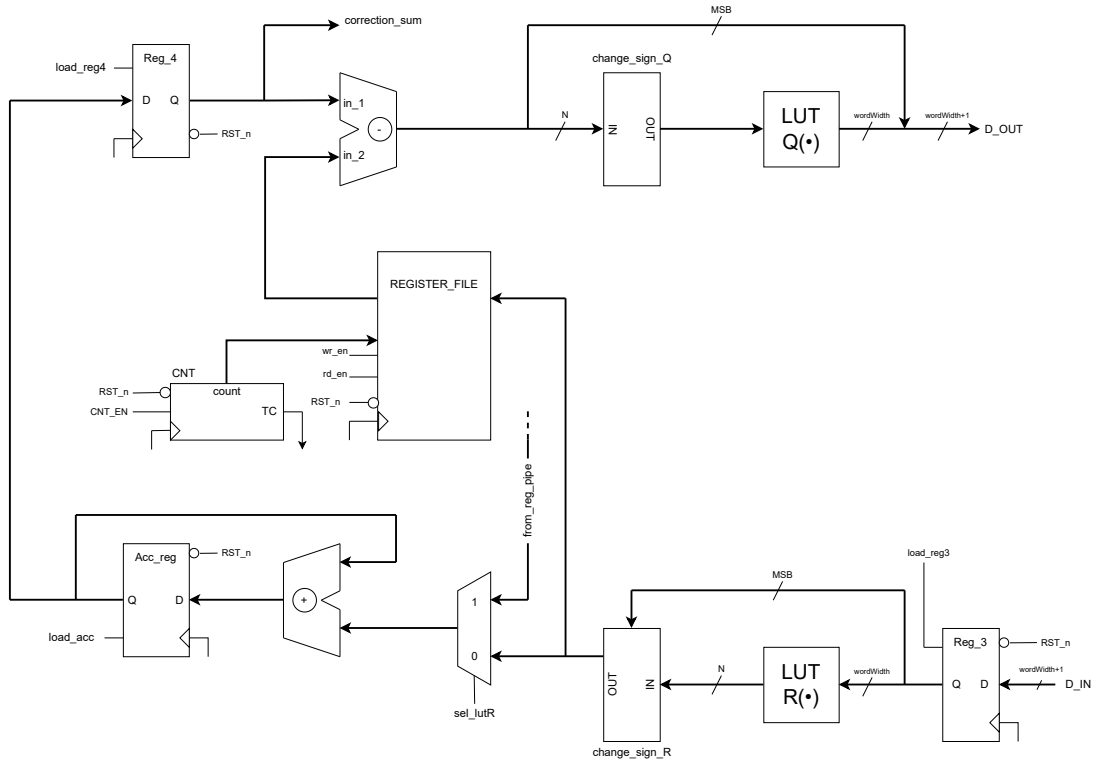


Figure 3.2: Block diagram of the Serial Single Variable Node

The data flow starts from the D_IN pin with information coming from the CN on wordWidth bits with an additional bit as MSB corresponding to the concatenated sign. These enter the Reconstruction block once the sign bit is separated. The data is reconstructed into an N-bit code-word, and the next block *change_sign_R* uses the sign to keep the data as it is if the sign is '0', otherwise, it flips the data.

Subsequently, a bifurcation divides the paths taken by the data: one section involves an accumulator that performs the sum of all incoming data plus the data coming from the channel thanks to the use of a multiplexer, while the other section proceeds to store all the data inside a register file using a counter to manage the addresses. This is necessary for the serial structure since the input data will be needed later, and as they arrive sequentially, it is necessary to save them inside a memory.

Once the number of inputs is finished (defined by the degree of the structure as better defined in the section dedicated to NET architectures), the Control Unit asserts the load signals for *Reg_4* and the *re_en* of the register file to proceed with the second section of the structure. The total sum passes through the register, becoming both an input to the subtractor and an output of the VN block. This sum will then be used to evaluate if the error correction is necessary.

To carry out the subsequent iterations of the algorithm, the subtractor will proceed to compute the difference between the total sum of the data and the input data that have been previously stored inside the register file. Each new result produced will be sent to the quantization block to be sent back to the CN. As happened for the reconstruction, the sign change will occur if the data is negative, and the sign will be concatenated as MSB to the quantized data.

In order to manage the flow of operations of the Serial Single structure, a simple Control Unit (CU) is implemented, mainly useful to manage data storage in the register file and the associated counter, as well as asserting signals for loading or resetting the machine. The process described by the CU is shown in figure 3.3.

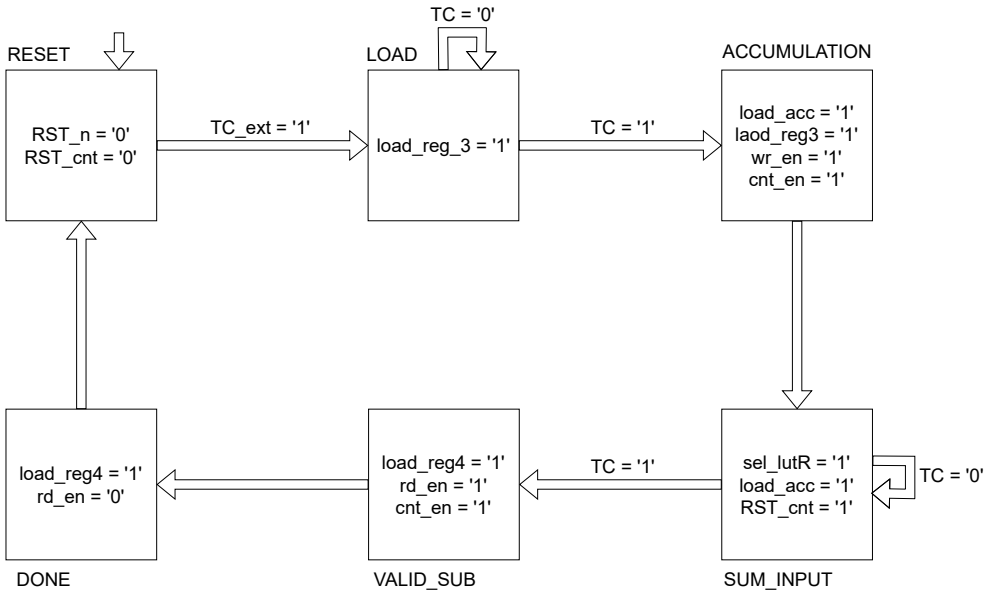


Figure 3.3: Control Unit of the Serial Single Variable Node

The serial structure, like the next ones presented later, have been described in a generic way in order to allow the modification of the data bit-width. This method enables the use of architecture with variable parameters without the need to entirely modify the block. For the same reason, the modularity of the architecture for quantization and reconstruction blocks has been ensured to provide the possibility of replacing blocks and testing new ones if necessary. This approach proves to be essential, as it allows testing structures using

different parameter values and conducting synthesis efficiently, taking part only on specific parameters without modify all components.

To facilitate these operations, a package called *constant* has been developed, which includes all variables that can be modified by the user interested in modifying the structure. This package contains the variables necessary for studying the structure. These may vary depending on the structure under consideration, but generally share some fundamental parameters:

- **N**: internal parallelism of the VN
- **wordWidth**: bit-width of the quantized data
- **GRADE**: degree of the VN under exam

Additional parameters necessary for the correct functionality of the structure can be inserted. Below is represented the code of the constant package for the Single Serial architecture, which studies degree 2 for the configuration [N, wordWidth] = [4, 3]:

```
-- constant.vhd (serial_single_LUT, Degree 2, [4,3])
package constants is
    constant N          : INTEGER := 4;  -- Internal parallelism   (4/5/6/7)
    constant wordWidth  : INTEGER := 3;  -- Quantized parallelism  (3/4/5)

    constant GRADE        : INTEGER := 2;  -- Degree of VN   (2/3/4/11)
    constant GRADE_length : INTEGER := 1;  -- num bits to represent the Degree

    constant MAX_N         : INTEGER := 7;
    constant MAX_wordWidth : INTEGER := 5;
end package constants;
```

## 3.2 Parallel Single

The parallel architecture is simpler than the serial one in terms of data flow. The Variable Node's structure receives inputs from the Check Nodes in a parallel way, which depends on the degree value. The degree indicates how many connections the Variable Node in exam has with the Check Nodes.

In this structure, the elements necessary for managing operations and maintaining proper timing in the serial architecture are no longer needed. For this reason, register file and counter are removed, and the control unit simply asserts load signals to maintain the desired timing. This introduces some interesting improvements as it simplifies the structure and reduces the occupied area of the data flow logic, but at the same time an additional complexity factor must be considered.

To execute simultaneous operations and to obtain a data flow that is mostly combinatorial, we need to repeat quantization and reconstruction blocks for each input data equal to the degree value.
In the next section we will discuss how the degree changes and what other structures will be analysed based on it. In the graph presented in 3.4 we showed a VN of degree 2.

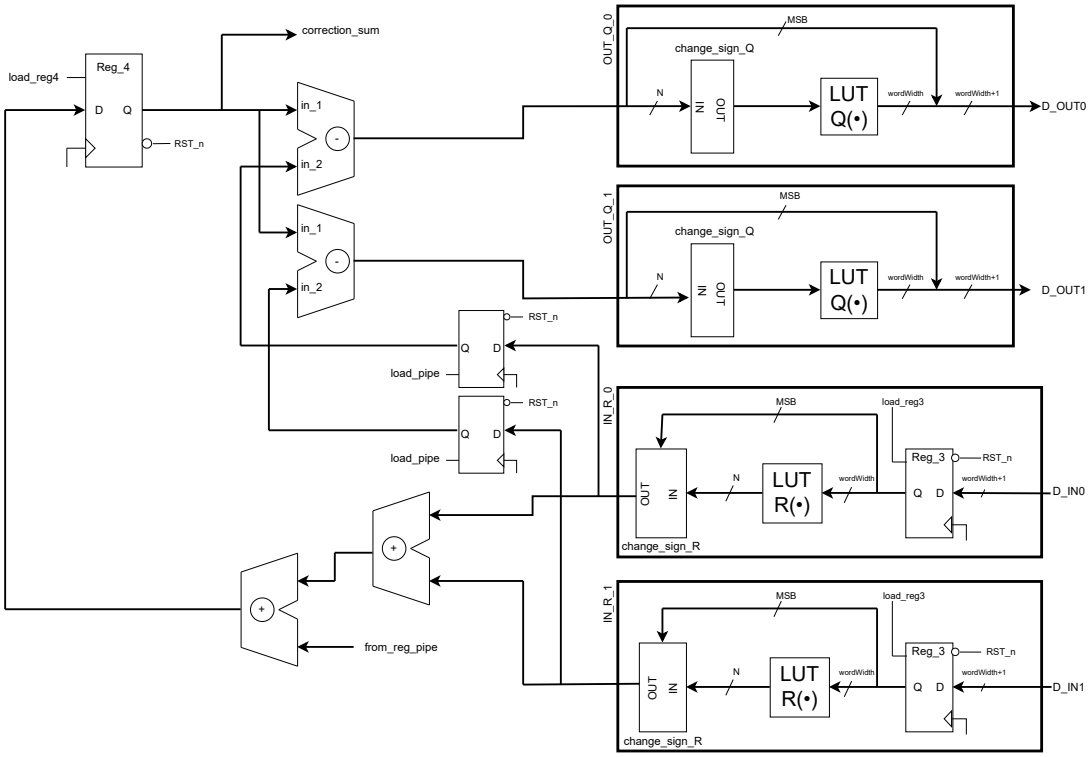


Figure 3.4: Block diagram of the Parallel Single Variable Node (Degree 2)

To maintain combinatorial operations, the accumulator used to obtain the total sum of inputs in the Serial configuration is replaced with a sum-blocks tree. This change increase the number of adders from one to a specified degree value. The same is true for reconstruction blocks: they are replicated in parallel matching the degree value, in order to provide data simultaneously to the CNs connected to the VN under examination. For this reason, the logic that precedes them also need a modification. Individual inputs are subtracted from the sum value of all inputs using subtractors which, as for adders, will increase by number from one to the specified degree value.

The sections dedicated to quantization and reconstruction are contained within blocks called $IN\_R\_X$ for quantization and $OUT\_Q\_X$ for reconstruction, where $X$ is the number of the input data to the structure. This substitution has been adopted to facilitate

the generation of components for the various degree distributions, as will be illustrated in the section dedicated to the NET architecture.

We decided to introduce the study of parallel architectures due to the trade-off mentioned: the high value of throughput and the simplicity of the structure is evident, but acquires a growing cost with the variation of the parameters under consideration. This aspect has been studied also for the serial configurations previously seen.

We show the resulting CU in figure 3.5 that is simpler than the Serial case. It is necessary to asserting some signals to maintain the correct timing, due to the combinatorial structure of the device.

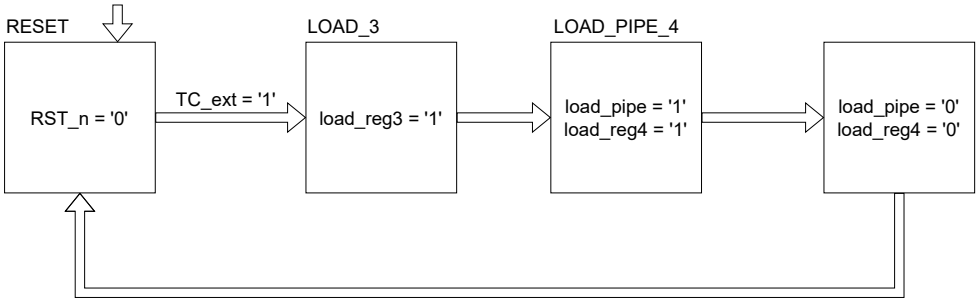


Figure 3.5: Control Unit of the Parallel Single Variable Node

## 3.3 LDPC WiFi codes - Degree Profile

To build an LDPC decoder that can handle error correction, it is necessary to employ a specific number of Variable Node and Check Node. This allocations are predetermined based on different parameters. In this scenario a number of VN equal to N = 1296 and a code-rate = 0.5 are selected, resulting in a corresponding number of Check Node equal to half the number of VN.

The singles Variable Node used in this structure are not uniform and differ according to the degree required for each individual element to properly encode the information. In other word, this indicates the number of connections that each individual VN must manage with the CNs.

To know the distribution of these degrees, we examine the matrix H, which defines all the connections between VN and CN based on the parameters set for N and code-rate. In our case, the interconnection matrix H used can be observed in figure 3.6 while the degree distribution is shown in figure 3.7 [7].

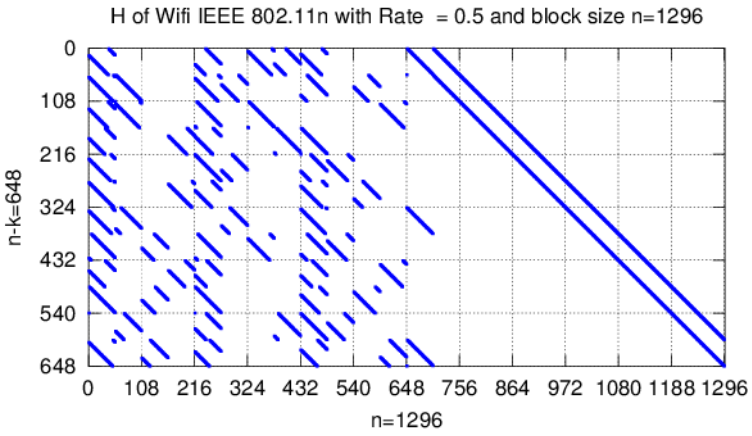


Figure 3.6: H interconnection matrix [7]

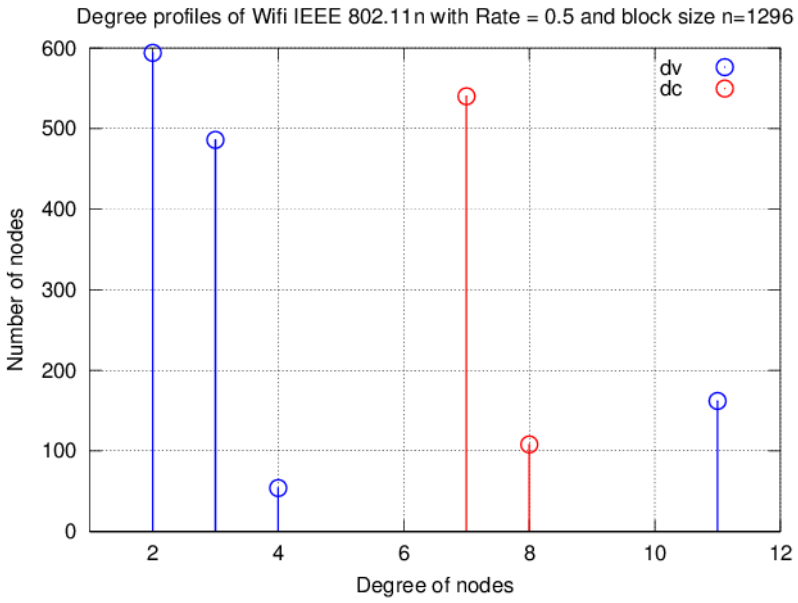


Figure 3.7: Degree profile. VN in blue, CN in red [7]

As previously studied for single VNs, the degree corresponds to 2,3,4 and 11 for VNs and 7 and 8 for CNs. In the analysis of the entire VN structure of our interest, 1296 instances of Variable Node will be instantiated, distributed in accordance with the observed graph 3.7 [7]. Therefore, we will use:

- 600 instances of degree 2
- 490 instances of degree 3
- 50 instances of degree 4
- 156 instances of degree 11

## 3.4 Serial Net

The previously analyzed structure corresponds to the fundamental building block, necessary in the structure of this application. The Variable Node has been described in a fully parametric manner, allowing not only the study of metrics as the parameters vary but also dynamic utilization of them within more complex structures, such as the mentioned Net.

As highlighted by the WiFi code specifications, the degrees required to define the VN Net at the Decoder level, with N and Rate specified, are four (2,3,4,11). In the Serial context, this does not provide heavy modifications to the structure of the individual VN, as the degree indicates the number of connections with the CNs and, consequently, the number of inputs in the structure.

This translates into the need to modify, through variables inserted in a package, the dimensions of the register file and the number of addresses pointed by the counter. A file is then created to generate the structure called *VN_net_GEN*, producing 1296 base VNs with the subdivision defined by the degree profile [7]. The block diagram representing the structure is shown in figure 3.8, in order to simplify the understanding of the final architecture.

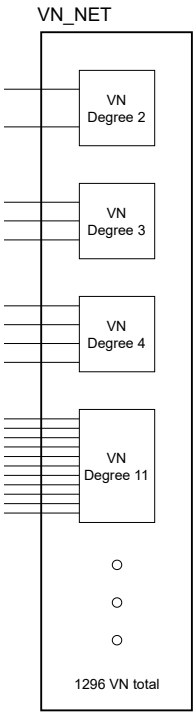


Figure 3.8: VN Net with total degree distribution

Through the constant *max_possible_grade*, the number of possible different configurations based on the degree is specified, which will be necessary for scrolling the *START* and *NUM_COMP* vectors containing the number of instances to generate for each degree. The code used for generation of the *Top_Entity* is shown below:

```
-- VN_net_GEN.vhd
GEN_BLOCK: FOR i IN 0 TO (max_possible_grade-1) GENERATE
    GEN_GRADE2: FOR j IN START(i) TO (NUM_COMP(i)-1) GENERATE
        GEN_VN_NET_Grade2 : TOP_ENTITY_VN_NET
            GENERIC MAP (NUM_GRADE(i), NUM_GRADE_length(i))
            PORT MAP(
                CLOCK          => CLOCK,
                RESETn         => RESETn,
                D_IN           => D_IN(j),
                D_OUT          => D_OUT(j),
                correction_sum => correction_sum(j),
                from_reg_pipe  => from_reg_pipe(j),
                TC_ext         => TC_ext);
    END GENERATE;
END GENERATE;
```

The code of the constants inside the package is shown below:

```
package constants is
    constant N          : INTEGER := 4;   -- Internal parallelism   (4/5/6/7)
    constant wordWidth  : INTEGER := 3;   -- Quantized parallelism  (3/4/5)

    TYPE IN_OUT_array IS ARRAY(INTEGER RANGE <>) OF UNSIGNED (wordWidth DOWNTO 0);
    TYPE SIGNED_array IS ARRAY(INTEGER RANGE <>) OF SIGNED   (N-1 DOWNTO 0);

    constant START      : GRADE_ARRAY := (0, 600, 1090, 1140);
    constant NUM_COMP   : GRADE_ARRAY := (600, 1090, 1140, 1296);

    constant NUM_VN             : INTEGER := 1296;  -- Total VN
    constant max_possible_grade : INTEGER := 4;

    type GRADE_ARRAY is array (natural range <>) of integer;

    constant NUM_GRADE          : GRADE_ARRAY := (2, 3, 4, 11); -- Grade of VN
    constant NUM_GRADE_length   : GRADE_ARRAY := (1, 2, 2, 4);
end package constants;
```

## 3.5 Parallel Net

The Net structure developed for the parallel architecture follows the same principle as the Serial one. The *Top_entity*, which includes the datapath and the control unit of the single VN, must be replicated a number of times determined by the variable N. This variable is configured based on the desired WiFi code and so the H matrix used.

However, the implementation in this configuration is more complex. The parallel structure has the advantage of simultaneously processing all input data, therefore speeding up the process because the results are calculated in a combinatorial way, leading to a very fast processing.
This advantage, however, is counterbalanced by the increase in the occupied area, necessary to manage the simultaneous processing of all input data. This results lead an increase in the number of elements required to handle such the inputs, proportional to the value of the degree.
Furthermore, it is important to note that in the Net structure, four different configurations will be generated, depending on the distribution of the degree into four values. These structures will have the number of inputs proportional to the degree, as well as the number of adders and subtractors within each VN.

In the section dedicated to the Parallel Single block, an example with a degree 2 has been studied (as highlighted in figure 3.4), which involves two quantization and reconstruction blocks for each of the two input data. The logic necessary for data memory has been

removed as well as the accumulator loop. Instead, trees of adders and subtractors, equal to the degree value, have been introduced to perform the operations in a combinatorial way.
The positioning of the adders within the tree structure has been carefully considered. Operations were scheduled to give symmetry to the tree, in order to avoid potential inconsistencies in the data, while still respecting the timing. This is evident from the fact that the sums are not carried out in cascade, but between parallel adders that operate on different data.

Also, in the block scheme, the R-Q blocks have been included in larger structures to simplify their generation, especially for high degree. To enhance the understanding of the block diagrams, these elements are represented as black boxes, named like the latter. Knowing that we are analyzing cases with degree distribution equal to 2, 3, 4, and 11, the respective parallel VN have been described in hardware.

The architecture for the degree equal to tree is illustrated in figure 3.9. The structures of degree 4 and degree 11 follow the same principle: they include a number of adders and subtractors corresponding to the degree value, as well as the inputs and outputs, which replicate the black boxes illustrated for the case of degree equal to tree.

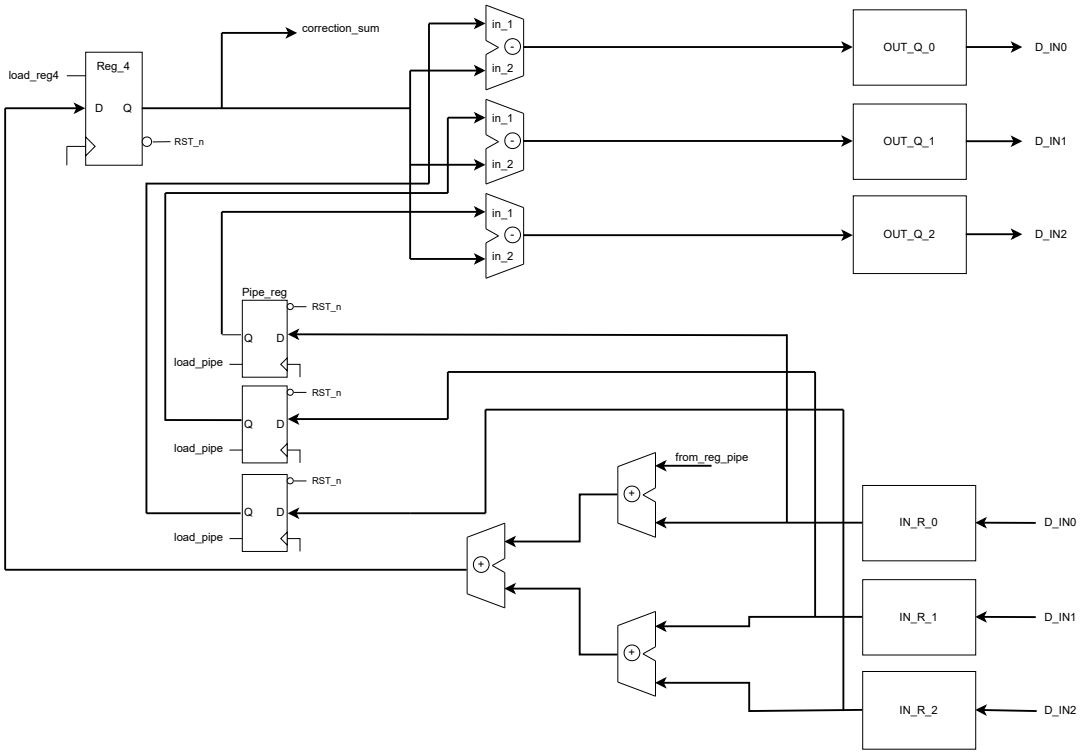


Figure 3.9: Block diagram of the Parallel Single Variable Node (Degree 3)

To create the structure called *Parallel Net*, it will be necessary to generate the structure using the blocks previously shown as fundamental elements. In this case it should be noted how the complexity of the code will increase, as the number of structures to be generated will vary depending on the configuration. The block diagram of this architecture is identical to that already illustrated in figure 3.8.

Below is an extract of the code used to generate the complete VN structure using the appropriate architectures depending on the grade (*VN_net_GEN_parallel.vhd*).

```
-- GRADE 2
GEN_GRADE2: FOR i IN START(0) TO (NUM_COMP(0)-1) GENERATE
    GEN_VN_NET_Grade2 : TOP_ENTITY_par_2
        PORT MAP(
            CLOCK          => CLOCK,
            RESETn         => RESETn,
            D_IN(0)        => D_IN(2*i),
            D_IN(1)        => D_IN((2*i)+1),
            D_OUT(0)       => D_OUT(2*i),
            D_OUT(1)       => D_OUT((2*i)+1),
            correction_sum => correction_sum(i),
            from_reg_pipe  => from_reg_pipe(i),
            TC_ext         => TC_ext);
END GENERATE;

-- GRADE 3
GEN_GRADE3: FOR j IN START(1) TO (NUM_COMP(1)-1) GENERATE
    GEN_VN_NET_Grade3 : TOP_ENTITY_par_3
        PORT MAP(
            D_IN(0)  => D_IN((3*(j-START(1)))+START(1)),
            D_IN(1)  => D_IN((3*(j-START(1)))+(START(1)+1)),
            [...]
            D_OUT(0) => D_OUT((3*(j-START(1)))+START(1)),
            D_OUT(1) => D_OUT((3*(j-START(1)))+(START(1)+1)),
            [...]);
END GENERATE;

-- GRADE 4
GEN_GRADE4: FOR k IN START(2) TO (NUM_COMP(2)-1) GENERATE
    GEN_VN_NET_Grade4 : TOP_ENTITY_par_4
        PORT MAP(
            D_IN(0)  => D_IN((4*(k-START(2)))+START(2)),
            D_IN(1)  => D_IN((4*(k-START(2)))+(START(2)+1)),
            [...]
            D_OUT(0) => D_OUT((4*(k-START(2)))+START(2)),
            D_OUT(1) => D_OUT((4*(k-START(2)))+(START(2)+1)),
            [...]);
END GENERATE;
```

```
-- GRADE 11
GEN_GRADE11: FOR z IN START(3) TO (NUM_COMP(3)-1) GENERATE
    GEN_VN_NET_Grade11 : TOP_ENTITY_par_11
        PORT MAP(
            D_IN(0)  => D_IN((11*(z-START(3)))+ START(3)),
            D_IN(1)  => D_IN((11*(z-START(3)))+(START(3)+1)),
            [...]
            D_OUT(0) => D_OUT((11*(z-START(3)))+ START(3)),
            D_OUT(1) => D_OUT((11*(z-START(3)))+(START(3)+1)),
            [...]);
END GENERATE;
```

## 3.6 LUT/THERMO architecture for R-Q block

In this section, we will focus on showing how the blocks related to quantization and reconstruction have been implemented. The architectures for this study are the lookup table (LUT) and a component called THERMO, which uses a set of comparators adopting an approach based on a thermometric scale for value assignment.

The main focus of this work is on analyzing the impact that blocks like LUT and THERMO have on the area and overall performance of the system. This implies the need to easily modify the structure both to adapt the desired block and to simplify future investigations, considering the possibility of introducing new blocks for evaluation.
For this reason, has been dedicated time to making the structure as modular as possible, focusing in particular on these blocks. About the use of LUT or THERMO, the quantization or reconstruction block maintains the same parallelism for input data and another one constant for output data.
Furthermore, the choice between serial or parallel architectures does not affect the components, which will always be the same in their description, but it modifies their number. In the parallel configuration, the number of components corresponds to the input data defined by the degree value.

The THERMO block has been proposed as a potential solution instead the LUT to improve quantization and reconstruction performance. This work aims to evaluate whether this approach offers significant advantages, and if confirmed, an actual improvement. Also, other architectures could be explored to optimize the performance of the entire decoder.

### 3.6.1 LUT

The use of the lookup table represents the simplest method. The quantization and reconstruction functions simply map the input message entering the block to an output message. This function can also be executed using a ROM memory.

However, it is important to note how the size of the LUT increases exponentially with the number of input bits. The input data is used as an address within the LUT, where

each value is associated with a unique value. The value associated with the input data is then returned at the desired bit-width. This means that to represent a complete LUT, it must contain a number of addresses equal to $2^N$, where N is the parallelism of the data internal to the structure, and a bit-width of the output data equal to wordWidth. In case a shorter length is desired, the value of the data will be truncated.
The reconstruction block implemented with LUT is identical to quantization, with the only difference that the inputs have a parallelism of wordWidth and the outputs of N, performing the inverse operation of quantization. The structure of the process for quantization and reconstruction is easily visualized in the block diagram shown in figure 3.10.

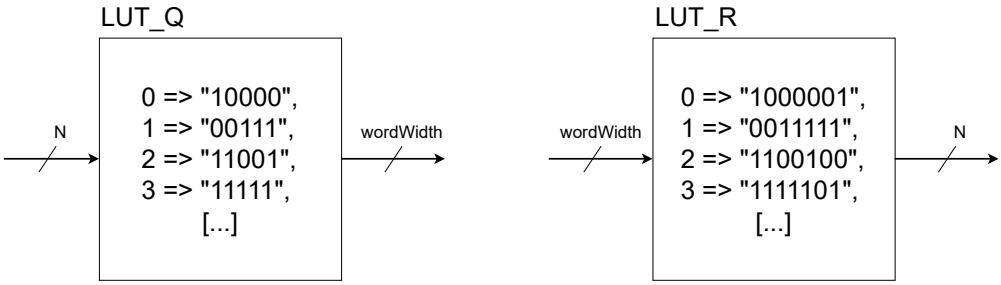


Figure 3.10: Working principle of Quantization LUT and Reconstruction LUT

In the code developed for the study reported in this thesis, the mappings performed by the LUT are stored within a package that is called by the component whenever quantization needs to be performed. This grants a greater simplicity in managing the blocks, allowing data modification without directly intervening on the component.

An extract of the code used to realize the LUT block and the package needed to do the association, are reported below:

```
-- LUT_Q_rom.vhd
[...]
TYPE LUT_rom IS ARRAY (0 TO (2**N - 1)) OF UNSIGNED((wordWidth - 1) DOWNTO 0);
    SIGNAL LUT_Q : LUT_rom;
    BEGIN
        g1: FOR i IN 0 TO ((2**N) - 1) GENERATE
            LUT_Q(i) <= Data(i)(wordWidth-1 DOWNTO 0);
        END GENERATE;
```

```
-- Package for LUT_Q (data_LUT_ROM_Q.vhd)
TYPE Data_Array IS ARRAY(0 TO (2**MAX_N-1)) OF UNSIGNED((MAX_wordWidth-1)
DOWNTO 0);
    CONSTANT Data : Data_Array :=(
    0 => "10000",
    1 => "00111",
    2 => "11001",
    3 => "11111",
    [...]

-- LUT_R_rom.vhd
[...]
TYPE LUT_rom IS ARRAY (0 TO (2**wordWidth - 1)) OF UNSIGNED((N - 1) downto 0);
    SIGNAL LUT_R : LUT_rom;
    BEGIN
        g1: FOR i IN 0 TO ((2**wordWidth) - 1) GENERATE
            LUT_R(i) <= Data(i)(N-1 DOWNTO 0);
        END GENERATE;

-- Package for LUT_R (data_LUT_ROM_R.vhd)
TYPE Data_Array IS ARRAY(0 TO (2**MAX_wordWidth - 1)) OF UNSIGNED((MAX_N-1)
DOWNTO 0);
    CONSTANT Data : Data_Array :=(
    0 => "1000001",
    1 => "0011111",
    2 => "1100100",
    3 => "1111101",
    [...]
```

For this reason, although they are relatively simple to study due to their unique association with data, they are also difficult to use due to their high cost. The goal, that also correspond to the scope of this thesis, has been to find valid alternatives to this quantization method, demonstrating how it is possible to achieve appreciable advantages even at a macroscopic level.
In this work, the study of LUT will be accompanied by the analysis of the THERMO block, which will perform the Q-R operation following different approaches.

### 3.6.2 THERMO

The solution called THERMO represents another approach that performs the same quantization function but differs in its implementation method. The resources used are different, and there are slight variations between the quantization and reconstruction blocks. For this reason, it is interesting to observe the impact of this method on performance and resource usage. The idea of this implementation was presented in the study conducted by [19] [17].

The quantization branch involves the use of two distinct blocks: the first, called *Comparator*, is responsible for comparing the input data with threshold values using comparators.

An iterative cycle scrolls through a number of values representable by $2^{wordWidth}$, comparing the input data with the unsigned representation of the pointer. The result of this comparison is assigned with '1' if the threshold is overcome, otherwise '0'. This approach is called *thermometric code* [3], where each threshold is sequentially compared, and each threshold passed yields '1', remembering the mercury column. The structure of this process is easily visualized in the block diagram shown in figure 3.11.

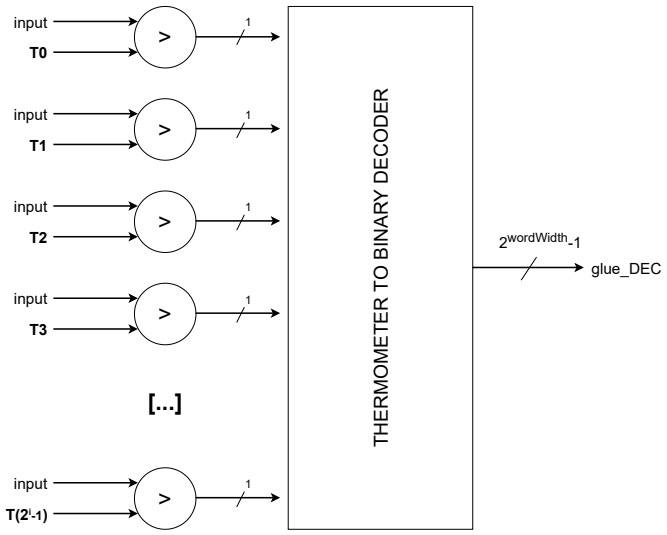


Figure 3.11: Working principle of Quantization THERMO - *Comparator*

The encoded value obtained corresponds to the new input for the second block that composes the structure, called *term_bin_dec*. This converts the thermometric code into a binary representation coded on wordWidth bits, shown in a table 3.1. These two elements are then grouped within the *comparator_Q* module.

However, to perform the comparisons, a specific *term_bin_dec* component is required for each wordWidth value. This is necessary to maintain the correctness of the hardware as the parameters vary. For this reason, three blocks corresponding to the values 3, 4, 5 have been developed and instantiated depending on the parameters being studied for that specific iteration.

| Thermometer Code | Binary form |
|---|---|
| 01111111 | 111 |
| 00111111 | 110 |
| 00011111 | 101 |
| 00001111 | 100 |
| 00000111 | 011 |
| 00000011 | 010 |
| 00000001 | 001 |
| 00000000 | 000 |

Table 3.1: Mapping relationship for quantization - *term_bin_DEC_3*

The following is an extract of the code for the *comparator* and the *term_bin_dec* (in this case the wordWidth = 3 is represented):

```
-- comparator.vhd
input  : IN  unsigned (N-1 DOWNTO 0);
output : OUT unsigned ((2**wordWidth)-1 DOWNTO 0));
[...]
PROCESS (input)
    BEGIN
        FOR i IN ((2**wordWidth)-1) DOWNTO 0 LOOP
            if  input  > to_unsigned((2**i)-1, N) then
                output(i) <= '1';
                else
                output(i) <= '0';
            end if;
        END LOOP;
END PROCESS;

-- term_bin_DEC_3.vhd -> for wordWidth = 3
input   : IN  unsigned ((2**wordWidth)-1 DOWNTO 0);
output  : OUT unsigned (wordWidth-1  DOWNTO 0));
[...]
process(input)
    begin
        if  input = "01111111" then
            output <= "111" ;
        elsif
            input = "00111111" then
            output <= "110" ;
        elsif
            input = "00011111" then
            output <= "101" ;
        [...]
```

The reconstruction block will be implemented using a multiplexer. The input data to be reconstructed will be associated with the selector of the multiplexer and will be compared with the corresponding value casted as Unsigned. A package will assign this value to the respective reconstructed data, which will then be provided as output. The package will provide unique associations for each value, representing $2^{wordWidth}$ combinations, and will return in output the reconstructed data on N bits. The structure is shown more clearly in figure 3.12. Additionally, an extract of the code describing the previously mentioned operation is included in the following.
For the implementation of the THERMO component, instead of the memories used by the LUTs, comparators are instantiated, which could offer high efficiency although very intensive.

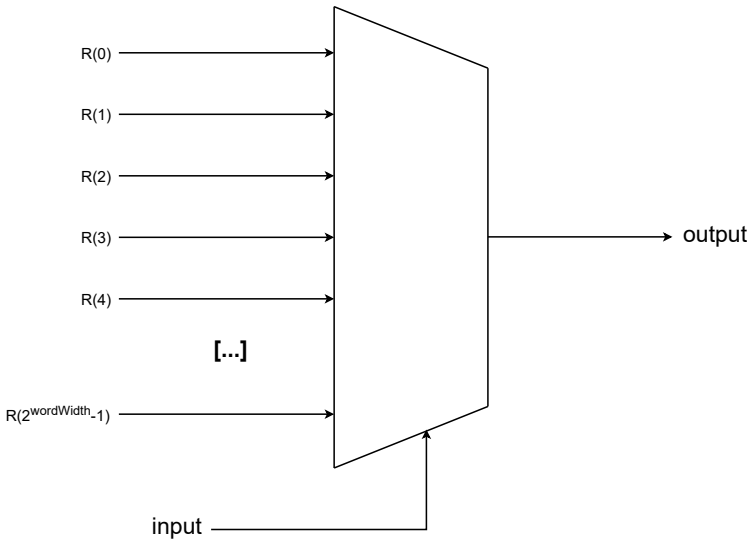


Figure 3.12: Working principle of Reconstruction THERMO

```
-- reconstructor.vhd
sel    : IN  unsigned ( wordWidth-1  downto 0 );
output : OUT unsigned ( N-1 downto 0 ));
[...]
PROCESS (sel)
    BEGIN
        FOR i IN 0 TO ((2**wordWidth)-1) LOOP
            IF sel = to_unsigned(i, wordWidth) THEN
                output <= Data(i)(N-1 DOWNTO 0);
            END IF;
        END LOOP;
END PROCESS;
```

```
-- data_reconstructor.vhd
package data_reconstructor is
    TYPE Data_Array IS ARRAY(0 TO (2**MAX_wordWidth - 1)) OF
                                unsigned((MAX_N-1) DOWNTO 0);
    CONSTANT Data : Data_Array :=(
    0  => "0000000",
    1  => "0000001",
    2  => "0000011",
    3  => "0000111",
    4  => "0001111",
    5  => "0011111",
    6  => "0111111",
    [...]
```

# Chapter 4

# Experimental Setup

In this chapter, we will present the strategies and software used to simulate and synthesize the circuits developed for all the combinations of interest, in order to obtain all significant results that demonstrate the efficiency of the various architectures, highlighting the different trade-offs that characterize their performance.

The development of these strategies followed a hierarchical approach, where each component was described starting from its fundamental elements. Attention was paid to verifying the correct functionality of each component using test-benches to validate the expected behavior. Subsequently, the complete Variable Node was developed by connecting all the components to create the Data-path of the structure. The Control-Unit was also defined, and finally, the entire structure was encapsulated within the Top-Entity, which describes the desired block to be synthesized.
To verify the correct operation of individual components and the hierarchical structure as a whole, ModelSim was used, a software developed by Siemens for simulating hardware description languages like VHDL and Verilog. This tool was used to validate the test-benches created specifically to verify the functionalities of the components.

Our goal is to conduct various synthesis of the described circuits, adopting different technologies and parameters. The synthesis of a logic circuit is the process by which, starting from the reference files of the algorithm (described more or less specifically depending on how much you want to influence design decisions), the logical expressions necessary to realize the circuit capable of executing the desired function are written.

The description files of the various architectures to be tested were then organized in an orderly manner within folders, in order to simplify navigation and facilitate the operation by the scripts during synthesis. Special attention was paid to the *constants* files containing the parameters to be modified to test all the combinations in play.
To have a better understanding of the organization of the synthesis, the tree diagram visible in figure 4.1 was created, showing the structure of the code.

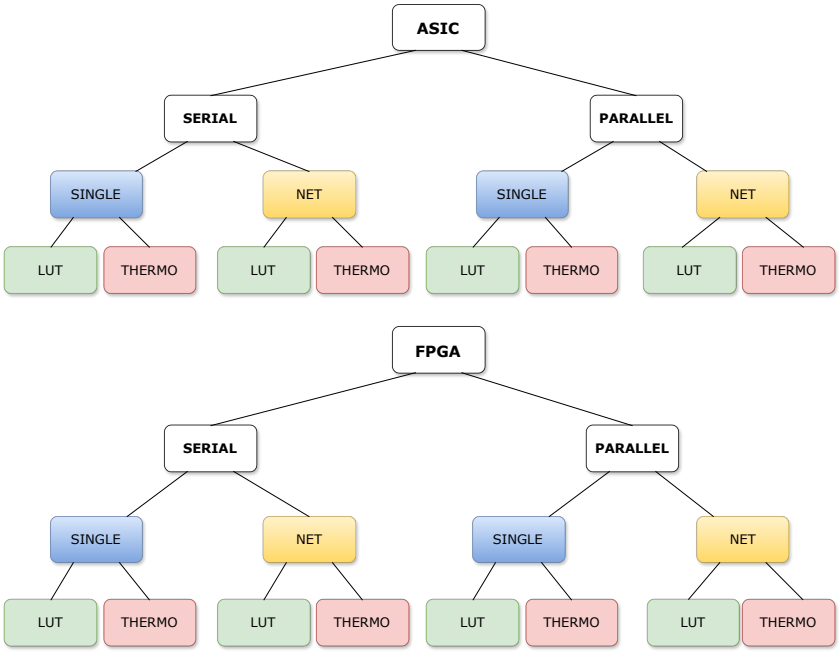


Figure 4.1: Tree diagram of code organization

During this study, synthesis will be performed on two different platforms in order to evaluate their performance and conduct a feasibility study. Initially, ASIC (Application Specific Integrated Circuit) synthesis will be examined, using a 65nm library for silicon implementation. Subsequently, synthesis will be carried out on FPGA, using a Cyclone V family board.

## 4.1 ASIC Synthesis

For ASIC silicon synthesis, the Synopsys software was used, employing libraries implementing the 65nm technology. In particular, libraries providing the worst-case scenario were utilized. This approach is crucial as it provides the upper limits of performance, allowing understanding of the extreme limits achievable by the device.

In order to automate the synthesis process for all possible configurations, scripts were developed to handle loading the design of the entire structure, processing the complete structure of the Top-Entity, performing compilation, and saving resulting area and timing reports within files generated by the script itself and placed in the correct folders for an easy access.
Furthermore, to perform a complete evaluation of the multiple combinations of internal and external bit-widths of the structure together with the degree value, as will be shown in the chapter dedicated to results, it was necessary to generate modifications in the files named *constants*. This was done so that the script, using nested loops, could acquire the

appropriate files for each synthesis combination.

A section of the script is provided for better understanding of the workflow (in this case a script for the Serial-Single-LUT configuration is reported).

```
# Create a procedure to execute the analysis of VHDL scripts.
proc run_analysis {grado interni esterni} {
    # Path to the directory containing the VHDL scripts.
    set vhdl_dir "../src/serial_single_LUT/$grado/${interni}_${esterni}/"

    # Perform the analysis for each of the VHDL codes.
    analyze -f vhdl -lib WORK $vhdl_dir/constants.vhd
    analyze -f vhdl -lib WORK ../src/serial_single_LUT/add.vhd
    analyze -f vhdl -lib WORK ../src/serial_single_LUT/reg.vhd
    [...]
    analyze -f vhdl -lib WORK ../src/serial_single_LUT/datapath_VN_net.vhd
    analyze -f vhdl -lib WORK ../src/serial_single_LUT/CU_VN_net.vhd
    analyze -f vhdl -lib WORK ../src/serial_single_LUT/TOP_ENTITY_VN_NET.vhd

    # Instantiate the top entity with the "Structure" architecture.
    elaborate TOP_ENTITY_VN_NET -arch Structure -lib WORK

    # Link and compile
    link
    compile

    # Report generation of timing and area
    report_timing > report/serial_single_LUT/
                    $grado/${interni}_${esterni}/report_timing.txt
    report_area -hierarchy > report/serial_single_LUT/
                    $grado/${interni}_${esterni}/report_area.txt

    # Necessary command to analyze the next architecture
    remove_design -all
}

# Iterate over various configurations of degree, in, and ext parameters.
foreach grado {2 3 4 11} {
    foreach interni {4 5 6 7} {
        foreach esterni {3 4 5} {
            # Skip the case where external >= internal.
            if {$esterni >= $interni} {
                continue
            }
            # Perform the analysis with the current configuration
            run_analysis $grado $interni $esterni
        }
    }
}
```

Regarding the use of THERMO components for R-Q operations, as mentioned earlier in the previous chapter, the quantization phase is divided into two blocks. One of these blocks, called *term_bin_DEC*, depends on the number of bits used for quantization. In order to properly adapt the architecture to the required specifications, a modification has been made to the script. This modification allows loading the correct configuration of the component based on the desired number of external bits (wordWidth). We show the isolated modification below:

```
if {$esterni == 3} {
    analyze -f vhdl -lib WORK ../src/serial_single_THERMO/
    term_bin_DEC_3/term_bin_DEC.vhd
} elseif {$esterni == 4} {
    analyze -f vhdl -lib WORK ../src/serial_single_THERMO/
    term_bin_DEC_4/term_bin_DEC.vhd
} elseif {$esterni == 5} {
    analyze -f vhdl -lib WORK ../src/serial_single_THERMO/
    term_bin_DEC_5/term_bin_DEC.vhd
}
```

A large number of *constant* files were generated using a second script written in Python, which differ only in the values of the variables used. The results reports are saved within directories, divided based on the degree and number of bits set for internal and external values.

In the reports related to area, the *Total Cell Area* value is recorded, indicating the total sum of the area of all components in square micrometers ($\mu m^2$). We chose to present a hierarchical area report, as indicated in the script, to analyze the area occupancy of all fundamental blocks in detail. This approach was useful for studying parallel applications as it allows easy evaluation of the contribution of each repeated element, considering the contribution of a specific block.

Regarding timing reports, the *Data arrival time* value is reported, defining the maximum period required by the critical path of the structure to complete the operation in nanoseconds ($ns$). A longer period defines a lower operating frequency, highlighting the point where intervention is needed to optimize the speed of the structure.

The synthesis was conducted using Synopsys software installed on a server provided by the university, accessed using a terminal provided by the X2GO client. Synthesis times were prolonged especially for *Net* circuits, indicating the considerable complexity of the structure, as expected given the large number of base VN involved. Additionally, synthesis times may have been influenced by the limited resources allocated by the server for each connected session.

Microsoft Excel was used for analyzing the data provided by the reports and creating graphs included in the results chapter, useful also for evaluating trade-offs between different architectures due to its ability to represent data in tabular form.

## 4.2 FPGA Synthesis

This section focuses on the study of all the configurations, already listed and studied on silicon for ASIC, on FPGA in order to observe the performance and costs in terms of area, with particular attention to Adaptive Logic Module (ALM), Adaptive LUT (ALUT) and Dedicated Logic Registers, to understand if the boards under analysis can be used to create the structure take in consideration. Unlike Synopsys synthesis, which have place the elements in software without resource restrictions, on FPGA these are limited, so optimizing them is necessary.

We used Intel Quartus Prime software for synthesis, selecting the Cyclone V family of boards and all related devices (E/GX/GT/SX/SE/ST) offered by sellers such as Altera, Terasic and Arrow. In particular, the *5CGXFC7C7F23C8* device is tested, which contains up to 56480 ALM units.
The operating conditions were set with a core voltage of 1.1V and a junction temperature range of 0 to 85 ºC (with the worst-case study focused on 0 ºC).
We have prepared scripts readable by the software to automate operations and save synthesis reports in appropriate folders based on the parameters under consideration. The reports we have analysed are the following:

- **<entity_name>.map.rpt**: summarizes the results produced by *"Analysis and synthesis"*, where in particular we observe the estimation of ALMs blocks utilization, ALUT usage for logic and dedicated logic registers used for the realization of the architecture.
- **<entity_name>.fit.rpt**: contains information on resources instantiated by the *"Place and route"*, and therefore the final cost, after the Fitter has made the performance optimizations.
- **<entity_name>.sta.rpt**: contains timing information to determine the maximum working frequency. We look at the case of *"Slow 1100 mV 0 °C Model"* as the worst case and, to force an optimization, we set an external clock with 10 ns period to force the synthesizer to stay within this parameter.

As defined in [1], the Adaptive Logic Module (ALM) blocks constitute the fundamental modules used by various families of boards, including the Cyclone V of our interest, to implement the required circuits. Each ALM can support up to eight inputs and eight outputs and contains two or four logic register cells, two cells for combinatorial logic, two dedicated full-adders with carry chains, and an adaptive LUT. Multiple functions can be implemented within a single ALM, and they are interconnected to achieve the overall function desired by the circuit. Figure 4.2 shows the block diagram of an ALM.

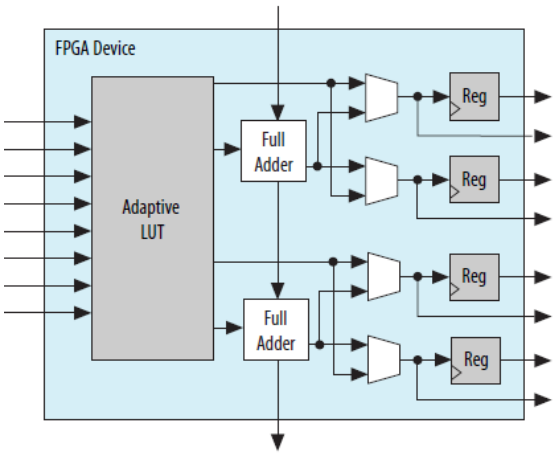


Figure 4.2: ALM Block Diagram for Cyclone V Devices [4]

In [1], is also possible to understand the meaning of Adaptive Look-Up Table (ALUT), that is a logical construct that illustrate the functionalities achievable through the combinational logic hardware of an Adaptive Logic Module (ALM) with the compatible device families. About the ALUT utilization, the physical resources represented differ based on device family.

The Fitter operation in Intel Quartus Prime is responsible for the Place & Route of the circuit. This procedure, as the name suggests, is divided into two phases, where the first concerns the initial placement of components and logic within the FPGA. Once that all elements have been loaded, the subsequent phase follows, aiming to connect all instantiated elements, creating all the wiring that link them. Additionally, during this phase, an optimization is performed to maintain the correct logic of the circuit with the expected connections, respecting the resources available on the board and minimizing their usage.
As also specified by the Quartus software instructions in [2], the Fitter operation begins with a random placement of components, subsequently optimizing space by connecting all components in the most effective manner. All these information and data will be discussed in the Results chapter.

The general observations on the advantages and disadvantages of the different implementations of the circuits follow the same ones made previously on ASIC, with the difference that in this case the resources are real and therefore limited. For this reason, we focus the study mainly on the number of resources used and the variation of them according to the architecture used.

It's important to highlight a synthesis result for the configuration of the complete Variable Node at the decoder level called *Net*. Even if a single VN is synthesized, this configuration with 1296 instances is too large for any FPGA board proposed, both the Cyclone V family and other proposals from Quartus Prime.

From the synthesis on ASIC it was evident that an occupation of that area was not possible because of the order of the couple of millimetres, that is too big to be able to use in general, but impossible to insert inside of a FPGA, or at least for those studied. This problem is therefore even more evident in the FPGA, making the *Net* analysis impossible in this section.

The Timing Analyzer checks that the required timing relationships between signals match the design and guarantee the correct functionality of signals observing that data arrives within the specified times for the inserted constraints. In this context, the objective was to determine the maximum achievable frequency of the architecture. Therefore, a lighter constraint of 10 $ns$ was defined to allow evaluation without excessively limiting the circuit structure, while still preventing the synthesis of a circuit without clock timing restrictions.

```
# Create Clock
create_clock -name {CLOCK} -period 10.000 [get_ports {CLOCK}]
```

FPGA synthesis was performed quickly using a local system, with defined performance specifications:

- Operative System Windows 10 Pro
- Intel Core i5 7500, 3.40 GHz
- 4 DRAM DDR4, 24 GBytes, Dual Channel
- Disk Unit SSD

This probably made FPGA synthesis operations faster, while it was not possible to compare it with the time that need the VN Net operations, which was highly time consuming for the ASIC synthesis.

# Chapter 5

# Experimental Results

The aim of the master's thesis is to explore the complexity of the Variable Node (VN) focusing on its key metrics, such as area and critical path, and therefore frequency. In order to achieve this goal we focused on examining all possible combinations between the following VN implementations:

- Serial and parallel architecture configuration
- Internal VN bit-width: 4/5/6/7
- Bit-width after quantization: 3/4/5
- Degree (by following the degree profiles [7] of WiFi IEEE 802.11 with Rate = 0.5 and block size N = 1296): 2/3/4/11
- Implementation of quantization and reconstruction: Look-up table (LUT) and THERMO blocks

In this chapter we show all the results produced by the synthesis of two distinct platforms: ASIC (application specific integrated circuit) using the 65nm library for the implementation on silicon via Synopsys and on FPGA (Field Programmable Gate Array), in particular the family of Cyclone V board, by using Intel Quartus Prime.
For a better understanding of the results and comparisons of the metrics, it can be useful to observe the tree diagram shown previously in figure 4.1, which illustrates the code structure and thus the order of the studied outcomes.

## 5.1 ASIC Synthesis Results

As mentioned above, we have synthesized the circuits of the two different architectures, in all possible combinations of the listed parameters on silicon using Synopsys software and using scripts to automate the process and save reports. The library used offers a channel length of 65nm and in particular the worst-case one was used, useful to understand what are the performance limitations under the worst conditions or implementation.

### 5.1.1 Serial

Below we conduct an analysis on the changes to the parameters within serial configurations. The structure denoted as **Single** identifies the examination of an individual VN, dependent on the degree under consideration. With **Net** we study the case of a complete VN block at the decoder level composed of N = 1296 instances of VN. These instances follow a specific distribution of degree described by the WiFi code parity check matrix H, with a code rate of 50% and N instances.

#### Single

In this section we present the resources needed in terms of the critical path, the area occupied by the quantization and reconstruction blocks and the total area of the entire VN. A comparison is carried forward to analyze the differences between the use of the look-up tables and THERMO for R-Q blocks in order to study the trade-off between the two options when changing the afore mentioned parameters.

In figure 5.1 we observe the area occupied by the two configurations of the quantization block. The complete data are represented in table 5.1 for LUTs and table 5.2 for THERMO. The table rows indicate the number of bits studied in the form [N, wordWidth] with N is the bit-width inside the VN and wordWidth the bit-width of the quantized data, while the columns indicate the degree under consideration.

The complexity shown in the graph matches the description of the hardware architecture and, therefore, the elements that have been instantiated. The complexity of the LUT depends on the parameter N, which is the bit length of the input to the block. The THERMO component requires a number of comparisons determined by the value of wordWidth, which represents the bit length of the quantized value output from the block. This implies different advantages of the two structures depending on the parameters used: for low values of N, the use of LUT remains preferable because, even with reduced values of wordWidth, the comparators negatively affect performance in terms of area. With high values of N, the use of comparators improves performance.

In particular, it is observed that for N equal to 4 and 5, the LUT architecture will always be more advantageous than THERMO, while starting from N = 7, the use of LUT becomes less advantageous. It is interesting to note that for N = 6, the THERMO configuration will be preferable unless the value of wordWidth exceeds the 5 bits, in which case the LUT configuration will become advantageous again.
The purpose of quantization is to reduce the number of bits used; therefore, the value of wordWidth will always be less than N.

In general, we can infer that for very small internal bit lengths, the use of look-up tables is advantageous, especially when the values of wordWidth approach those of N, while for significantly larger N values compared to wordWidth, it will be advantageous to use THERMO, with some exceptions to consider when the bit lengths are similar.

This can be deducted from the fact that the use of THERMO eliminates the dependence on N, which usually takes on high values, while wordWidth remains contained and always lower than N, transferring the area dependence to a smaller parameter.

The result is of particular interest because it allows to understand the cost in resources according to the parameters under observation, and therefore, to optimize the use of area based on specific applications that use a precise number of bits. For the serial case, no differences arises in terms of degree because it affects only the size of the register file and the counter inside the VN, therefore the R and Q blocks do not change.

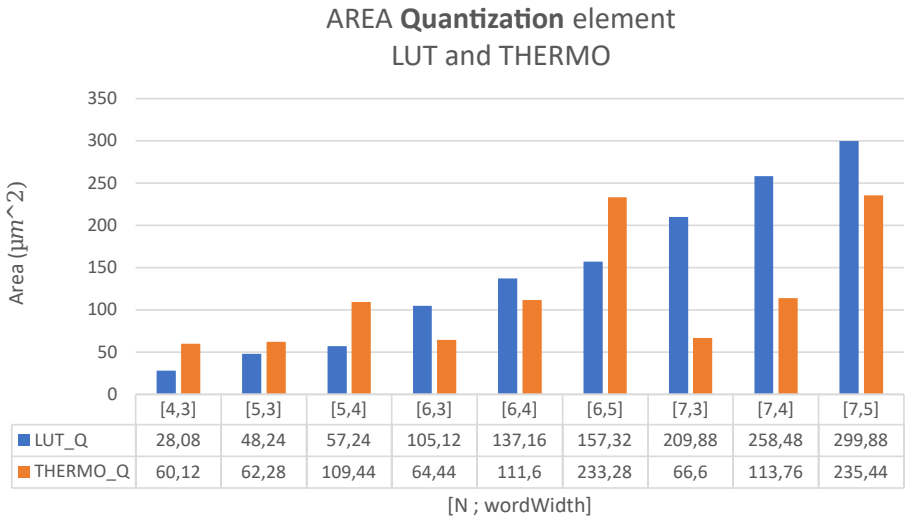


Figure 5.1: Relationship of Area between LUT and THERMO Quantization block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 28,08 | 28,08 | 28,08 | 28,08 |
| **[5,3]** | 48,24 | 48,24 | 48,24 | 48,24 |
| **[5,4]** | 57,24 | 57,24 | 57,24 | 57,24 |
| **[6,3]** | 105,12 | 105,12 | 105,12 | 105,12 |
| **[6,4]** | 137,16 | 137,16 | 137,16 | 137,16 |
| **[6,5]** | 157,32 | 157,32 | 157,32 | 157,32 |
| **[7,3]** | 209,88 | 209,88 | 209,88 | 209,88 |
| **[7,4]** | 258,48 | 258,48 | 258,48 | 258,48 |
| **[7,5]** | 299,88 | 299,88 | 299,88 | 299,88 |

Table 5.1: Area ($\mu m^2$) of LUT_Q

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 60,12 | 60,12 | 60,12 | 60,12 |
| **[5,3]** | 62,28 | 62,28 | 62,28 | 62,28 |
| **[5,4]** | 109,44 | 109,44 | 109,44 | 109,44 |
| **[6,3]** | 64,44 | 64,44 | 64,44 | 64,44 |
| **[6,4]** | 111,6 | 111,6 | 111,6 | 111,6 |
| **[6,5]** | 233,28 | 233,28 | 233,28 | 233,28 |
| **[7,3]** | 66,6 | 66,6 | 66,6 | 66,6 |
| **[7,4]** | 113,76 | 113,76 | 113,76 | 113,76 |
| **[7,5]** | 235,44 | 235,44 | 235,44 | 235,44 |

Table 5.2: Area ($\mu m^2$) of THERMO_Q

The study proceeds with the analysis of the reconstruction block. In figure 5.2 we observe the impact on the area of the two configurations while the complete data have been brought back in table 5.3 for the lookup table and in table 5.4 for the THERMO.

The LUT, like in the quantization branch, increases in area exponentially with the bit-width in input. In this context, the block size will depend on the wordWidth bit-width since we are in the context of reconstruction. We will then analyze, through the graph, how its size will increase in relation to the wordWidth parameter.
On the contrary, the THERMO block remains stable, even when the parameters change, due to the fact that it is implemented using a multiplexer. As the wordWidth width changes, only the number of wires connecting the reconstruction data changes, showing a slight increase in area and consequently a constant cost in terms of overall area.
It is interesting to note that for parameters up to [6,4], it is advantageous to use lookup tables in terms of area. The use of the multiplexer becomes advantageous for high values of N, but especially for wordWidth.

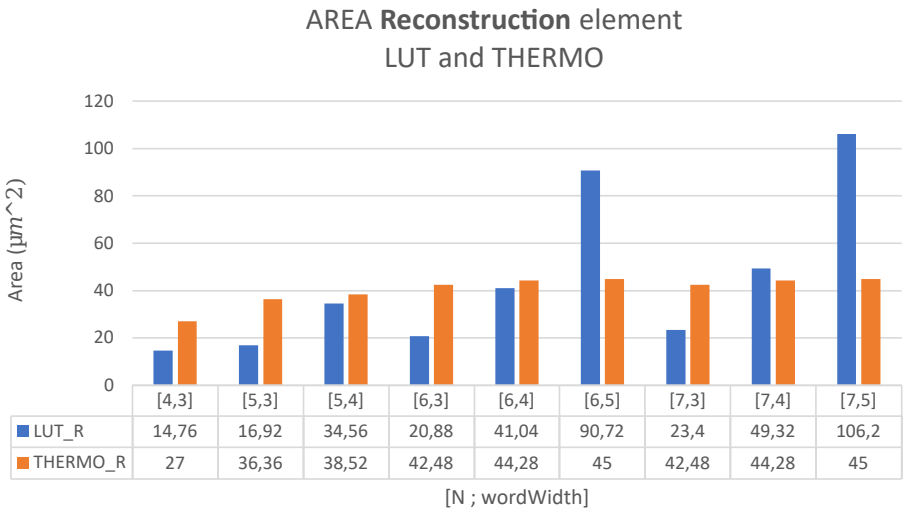


Figure 5.2: Relationship of Area between LUT and THERMO Reconstruction block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 14,76 | 14,76 | 14,76 | 14,76 |
| **[5,3]** | 16,92 | 16,92 | 16,92 | 16,92 |
| **[5,4]** | 34,56 | 34,56 | 34,56 | 34,56 |
| **[6,3]** | 20,88 | 20,88 | 20,88 | 20,88 |
| **[6,4]** | 41,04 | 41,04 | 41,04 | 41,04 |
| **[6,5]** | 90,72 | 90,72 | 90,72 | 90,72 |
| **[7,3]** | 23,4 | 23,4 | 23,4 | 23,4 |
| **[7,4]** | 49,32 | 49,32 | 49,32 | 49,32 |
| **[7,5]** | 106,2 | 106,2 | 106,2 | 106,2 |

Table 5.3: Area ($\mu m^2$) of LUT_R

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 27 | 27 | 27 | 27 |
| **[5,3]** | 36,36 | 36,36 | 36,36 | 36,36 |
| **[5,4]** | 38,52 | 38,52 | 38,52 | 38,52 |
| **[6,3]** | 42,48 | 42,48 | 42,48 | 42,48 |
| **[6,4]** | 44,28 | 44,28 | 44,28 | 44,28 |
| **[6,5]** | 45 | 45 | 45 | 45 |
| **[7,3]** | 42,48 | 42,48 | 42,48 | 42,48 |
| **[7,4]** | 44,28 | 44,28 | 44,28 | 44,28 |
| **[7,5]** | 45 | 45 | 45 | 45 |

Table 5.4: Area ($\mu m^2$) of THERMO_R

At this point it is interesting to know the differences in terms of total area. In this case it fits not only the contribution of the quantization and reconstruction elements but also the rest of the logic that build the variable node, discriminating as always the two architectures. Figure 5.3 shows the result in a graph so that it can be easily consulted, while the complete data are given in tables 5.5 for LUT and 5.6 for THERMO.

In this situation, the contribution of the degree becomes relevant. It mainly affects the memory block (register file and counter for addressing) as the degree increases. That will boost linearly the size occupied by the VN. In general, the utilization of comparators in the THERMO blocks helps to manage the space efficiently.
For reduced bit-width, employing the LUTs remains possible, but on the overall configuration they turn out to be area consuming and therefore impractical for resources conservation.

This final study shows how, in the case of using a single VN in serial configuration, it is more convenient to replace R-Q lookup table blocks with alternative solutions due to their dependency of the area that increases exponentially with the number of input bits to the block. A first possible solution of this type has been proven to be the use of THERMO block, implemented with comparators.

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 495,72 | 564,48 | 626,4 | 1047,59 |
| **[5,3]** | 612,72 | 694,44 | 769,68 | 1280,16 |
| **[5,4]** | 650,16 | 731,88 | 807,12 | 1317,59 |
| **[6,3]** | 745,56 | 841,68 | 930,24 | 1529,99 |
| **[6,4]** | 808,56 | 904,68 | 993,24 | 1593 |
| **[6,5]** | 889,2 | 985,32 | 1073,88 | 1673,64 |
| **[7,3]** | 943,2 | 1052,28 | 1154,16 | 1843,91 |
| **[7,4]** | 1028,52 | 1137,6 | 1239,48 | 1929,24 |
| **[7,5]** | 1137,6 | 1246,68 | 1348,56 | 2038,32 |

Table 5.5: Tot. Area ($\mu m^2$) - LUT blocks

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 540 | 608,76 | 670,68 | 1091 |
| **[5,3]** | 646,2 | 727,92 | 803,16 | 1313,64 |
| **[5,4]** | 706,32 | 788,04 | 863,28 | 1373,76 |
| **[6,3]** | 726,48 | 822,6 | 911,16 | 1510,84 |
| **[6,4]** | 786,24 | 882,36 | 970,92 | 1570,68 |
| **[6,5]** | 919,44 | 1015,56 | 1104,12 | 1703,88 |
| **[7,3]** | 819 | 928,08 | 1029,96 | 1719,72 |
| **[7,4]** | 878,76 | 987,84 | 1089,72 | 1779,48 |
| **[7,5]** | 1011,96 | 1121,04 | 1222,92 | 1912,68 |

Table 5.6: Tot. Area ($\mu m^2$) - Thermo blocks

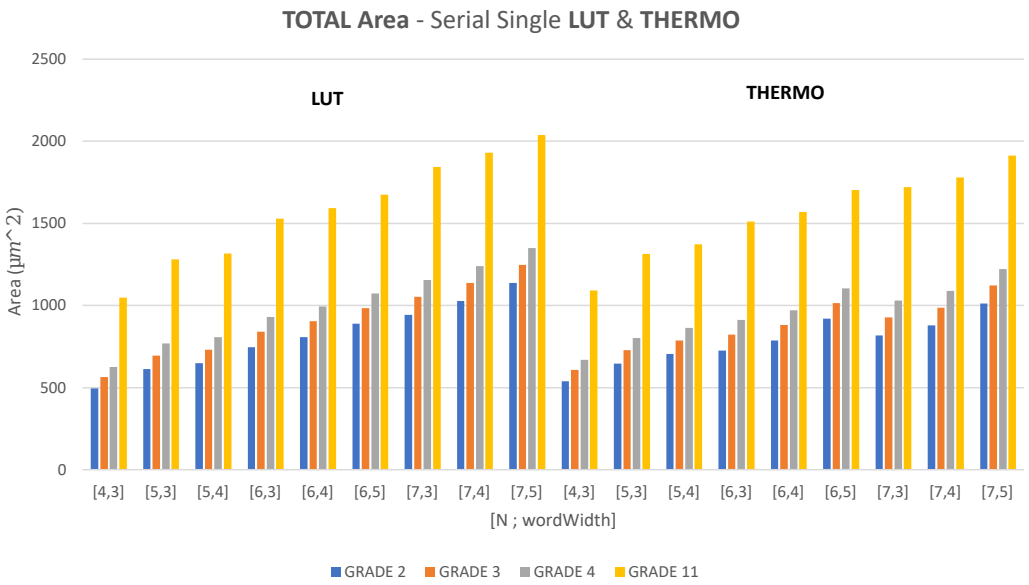


Figure 5.3: Relationship of Total Area between VN using LUT (left) and THERMO (right) blocks

To provide a complete overview of the ongoing study, we have examined the contribution of quantization and reconstruction on the total area, expressed as a percentage value. This supplementary data can be found in tables 5.7 for LUT and 5.8 for THERMO. These data provide the comprehension of the significance of the area occupied by both R-Q elements within the total area, offering insight into various configurations.

The percentage is derived by considering the sum of the area occupied by the quantization block Q and the reconstruction block R, relative to the total area of the corresponding configuration.

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 8,6% | 7,6% | 6,8% | 4,1% |
| **[5,3]** | 10,6% | 9,4% | 8,5% | 5,1% |
| **[5,4]** | 14,1% | 12,5% | 11,4% | 7,0% |
| **[6,3]** | 16,9% | 15,0% | 13,5% | 8,2% |
| **[6,4]** | 22,0% | 19,7% | 17,9% | 11,2% |
| **[6,5]** | 27,9% | 25,2% | 23,1% | 14,8% |
| **[7,3]** | 24,7% | 22,2% | 20,2% | 12,7% |
| **[7,4]** | 29,9% | 27,1% | 24,8% | 16,0% |
| **[7,5]** | 35,7% | 32,6% | 30,1% | 19,9% |

Table 5.7: totLUTarea/totArea (%)

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 16,1% | 14,3% | 13,0% | 8,0% |
| **[5,3]** | 15,3% | 13,6% | 12,3% | 7,5% |
| **[5,4]** | 20,9% | 18,8% | 17,1% | 10,8% |
| **[6,3]** | 14,7% | 13,0% | 11,7% | 7,1% |
| **[6,4]** | 19,8% | 17,7% | 16,1% | 9,9% |
| **[6,5]** | 30,3% | 27,4% | 25,2% | 16,3% |
| **[7,3]** | 13,3% | 11,8% | 10,6% | 6,3% |
| **[7,4]** | 18,0% | 16,0% | 14,5% | 8,9% |
| **[7,5]** | 27,7% | 25,0% | 22,9% | 14,7% |

Table 5.8: totTHERMOarea/totArea (%)

Finally, we observe the resulting critical path that we get in the different configurations. The critical path is identified across the quantization branch. This means that, in this study, the degree value does not affect performance as the only elements depending on it (register file and counter) are not part of the path under examination. This conclusion is supported by the data collected and presented in tables 5.9 for the LUT and 5.10 for the THERMO.

As the bit-width increases, there is an increase in the period using LUT blocks, while a reduction in period is observed using THERMO blocks. This result can be explained by the fact that LUT blocks associate a specific quantization with each value, which can generate significant internal fan-out, slowing down encoding operations. On the other hand, the THERMO block utilizes simple comparisons on a reduced number of values, allowing for faster operations. The values have been inserted in a graph and are observable in figure 5.4.

In general, this demonstrates an advantage on the statistics for the THERMO block, with reduced circuit area occupation and operates with lower period, and so, with a higher frequencies.

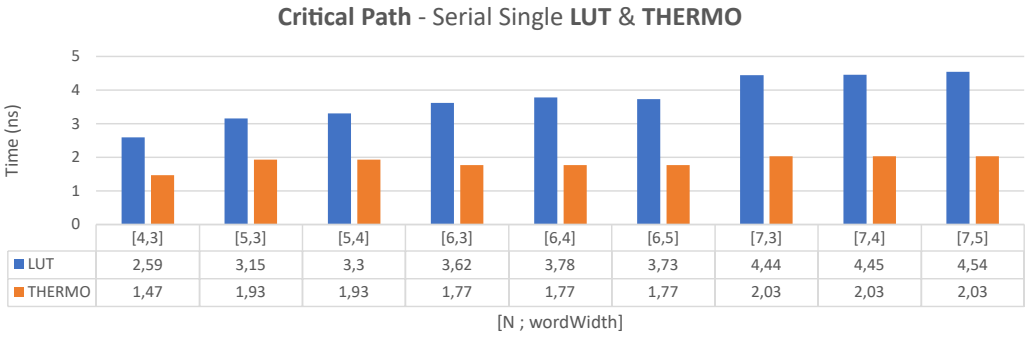


Figure 5.4: Relationship of the Critical path period ($ns$) between LUT and THERMO

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 2,59 | 2,59 | 2,59 | 2,59 |
| **[5,3]** | 3,15 | 3,15 | 3,15 | 3,15 |
| **[5,4]** | 3,3 | 3,3 | 3,3 | 3,3 |
| **[6,3]** | 3,62 | 3,62 | 3,62 | 3,62 |
| **[6,4]** | 3,78 | 3,78 | 3,78 | 3,78 |
| **[6,5]** | 3,73 | 3,73 | 3,73 | 3,73 |
| **[7,3]** | 4,44 | 4,44 | 4,44 | 4,44 |
| **[7,4]** | 4,45 | 4,45 | 4,45 | 4,45 |
| **[7,5]** | 4,54 | 4,54 | 4,54 | 4,54 |

Table 5.9: Critical Path ($ns$) LUT

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 1,47 | 1,47 | 1,47 | 1,47 |
| **[5,3]** | 1,93 | 1,93 | 1,93 | 1,93 |
| **[5,4]** | 1,93 | 1,93 | 1,93 | 1,93 |
| **[6,3]** | 1,77 | 1,77 | 1,77 | 1,77 |
| **[6,4]** | 1,77 | 1,77 | 1,77 | 1,77 |
| **[6,5]** | 1,77 | 1,77 | 1,77 | 1,77 |
| **[7,3]** | 2,03 | 2,03 | 2,03 | 2,03 |
| **[7,4]** | 2,03 | 2,03 | 2,03 | 2,03 |
| **[7,5]** | 2,03 | 2,03 | 2,03 | 2,03 |

Table 5.10: Critical Path ($ns$) THERMO

### Net

In this section, we will focus on analyzing a more realistic application of the Variable Node. The configuration of an LDPC decoder is determined by the interconnection matrix H, which establishes connections between Variable Nodes (VN) and Check Nodes (CN) within the entire structure. The configuration varies depending on the type of codes we want to examine; in our case, it corresponds to N = 1296 instances with a code-rate of 50%.

Through this study, we identify the distribution of degrees necessary to realize a complete VN with all generated instances, as shown in the chapter dedicated to the architecture. The main objective is to highlight the impact that this structure has on the occupied area and the differences that arise depending on what LUT or THERMO blocks are used for quantization and reconstruction.

The results of the synthesis are shown in figure 5.5 where we can observe the cost in terms of area occupied by the two configurations under examination. The numerical data are collected within table 5.11 for LUT and in table 5.12 for THERMO.

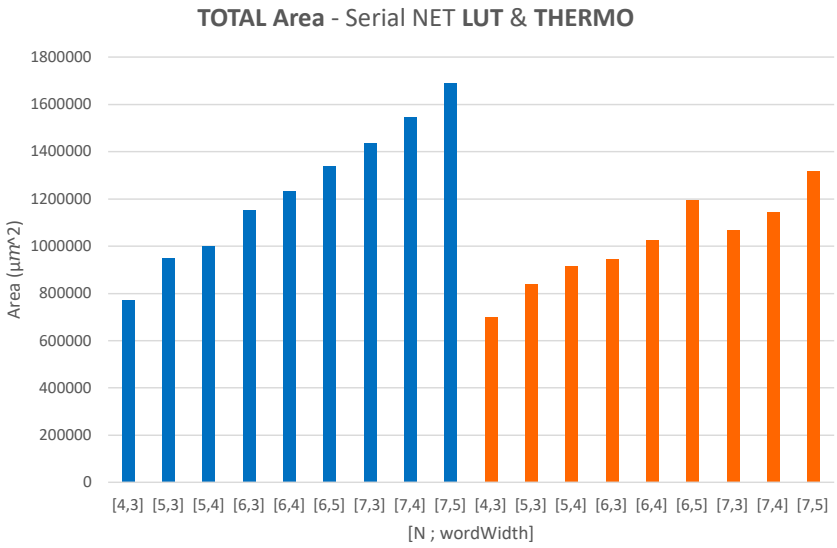


Figure 5.5: Relationship of Total Area between NET VN using LUT (blue) and THERMO (orange) blocks

| [N, wordWidth] | Total Area ($\mu m^2$) |
|---|---|
| [4,3] | 773858,88 |
| [5,3] | 951539,77 |
| [5,4] | 1000062,01 |
| [6,3] | 1152563,05 |
| [6,4] | 1234211,05 |
| [6,5] | 1338720,49 |
| [7,3] | 1437299,29 |
| [7,4] | 1547874,01 |
| [7,5] | 1689241,69 |

Table 5.11: Total Area for NET LUT

| [N, wordWidth] | Total Area ($\mu m^2$) |
|---|---|
| [4,3] | 701706,24 |
| [5,3] | 839341,45 |
| [5,4] | 917256,97 |
| [6,3] | 947116,81 |
| [6,4] | 1024565,77 |
| [6,5] | 1197192,97 |
| [7,3] | 1067022,73 |
| [7,4] | 1144471,69 |
| [7,5] | 1317098,89 |

Table 5.12: Total Area for NET THERMO

Considering the previous research that highlighted an advantage in using THERMO blocks for low-cost applications in terms of occupied space and high frequency, it is observed that this advantage also extends to a higher level, showing the overall benefit of using THERMO blocks.

It is important to note, however, that despite improvements in resource efficiency, in both the use of LUTs and THERMO blocks, there is an excessively high area occupation about two square millimeters.
This data is unsatisfactory for the realization of a complete VN structure, especially considering that in a fully unrolled architecture [8], the decoding operation will be carried out iteratively by repeated structures, resulting in the increase in area proportional to the number of iterations.

In conclusion, it is represented for the Net structure the resulting critical path that determines the period and so the working frequency. As already shown in the study of individual elements, there is an improvement in performance thanks to the use of THERMO blocks that lead to reduced periods remaining almost stable to the change of parameters.

The graph showing the results obtained is represented in figure 5.6. To better consult the numerical values, we report below the tables for the LUT in 5.13 and for the THERMO in 5.14.

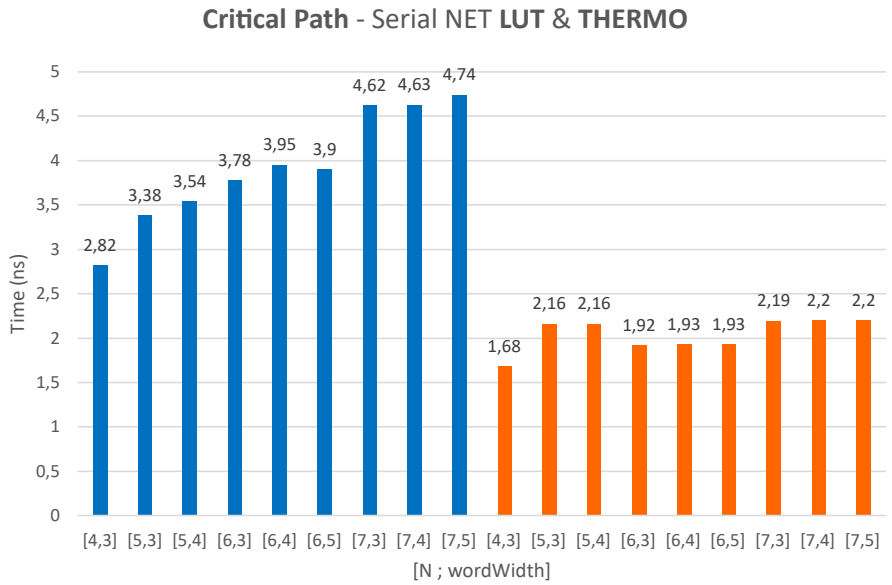


Figure 5.6: Relationship of the Critical Path period ($ns$) between LUT (blue) and THERMO (orange)

| [N, wordWidth] | Critical path ($ns$) |
|---|---|
| **[4,3]** | 2,82 |
| **[5,3]** | 3,38 |
| **[5,4]** | 3,54 |
| **[6,3]** | 3,78 |
| **[6,4]** | 3,95 |
| **[6,5]** | 3,9 |
| **[7,3]** | 4,62 |
| **[7,4]** | 4,63 |
| **[7,5]** | 4,74 |

Table 5.13: Critical path for NET LUT

| [N, wordWidth] | Critical path ($ns$) |
|---|---|
| **[4,3]** | 1,68 |
| **[5,3]** | 2,16 |
| **[5,4]** | 2,16 |
| **[6,3]** | 1,92 |
| **[6,4]** | 1,93 |
| **[6,5]** | 1,93 |
| **[7,3]** | 2,19 |
| **[7,4]** | 2,2 |
| **[7,5]** | 2,2 |

Table 5.14: Critical path for NET THERMO

### 5.1.2 Parallel

In this section, we will analyze parallel architecture, considering the various configurations previously mentioned for the Serial case, as the parameters vary.

The structure involves the replacing of the necessary elements for serial data management with a parallel structure that allows simultaneous and combinatorial management of input data. This will introduce improvements in terms of throughput; however, it will leads to a significant increase in area, as it will be necessary to replicate the quantization and reconstruction blocks for each input data to the structure. It becomes therefore of interest to evaluate how expensive is the increase in area and how many resources are required for the various parameters under examination, in order to observe the trade-off between the different structures, depending on the parameters considered.

For this reason, the results provided by the area and critical path syntheses of parallel structures for architectures that mount both LUT and THERMO blocks will be studied below. In the next section we will focus on analyzing the differences between a serial and parallel implementation, always discriminating between the two possibilities listed to see which configurations are most convenient depending on the desired specific application.

The study is carried out both for the VN taken individually (**Single**) and for the decoder structure with the N = 1296 instances (**Net**) and with the same distribution of degree shown in figure 3.7.

#### Single

In this section we analyze the structure of the Variable Node in terms of area and critical path. The study begins with the analysis of the quantization block observing the differences between the implementation with lookup tables and the THERMO block build using comparators.
It is important to note that in this parallel case, unlike the serial one, the degree affects the area occupied by this stage since the structure is replicated a number of times equal to the value of the degree itself.

We report the data of the area occupied in the two representations in table 5.15 for the LUT and in table 5.16 for the THERMO solution. In order to better understand the differences, the data is shown in the diagram in figure 5.7.

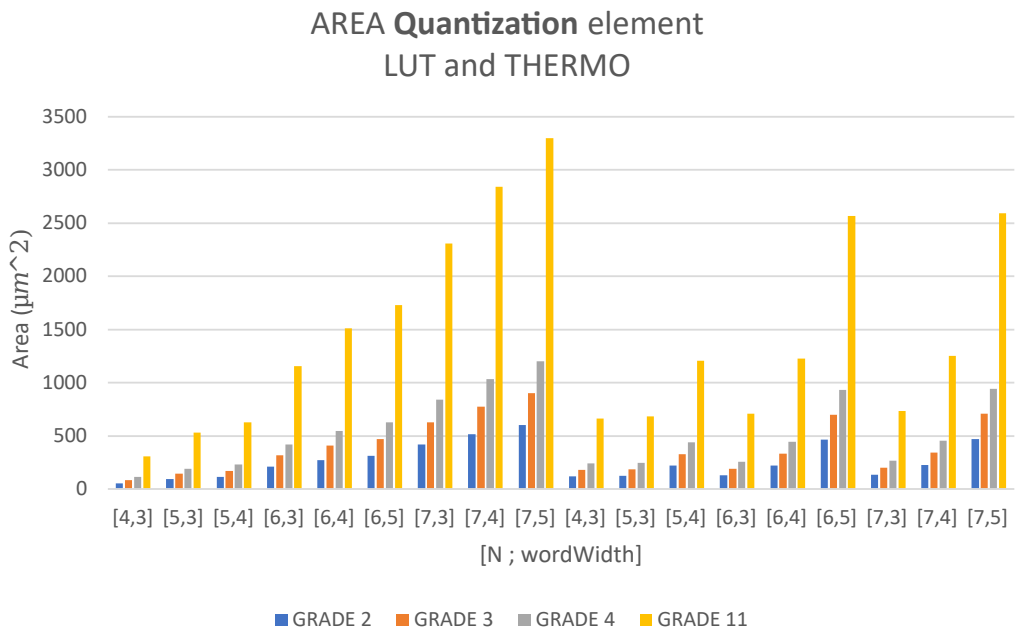


Figure 5.7: Relationship of Area between LUT (left) and THERMO (right) Parallel Quantization block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 56,16 | 84,24 | 112,32 | 308,88 |
| **[5,3]** | 96,48 | 144,72 | 192,96 | 530,64 |
| **[5,4]** | 114,48 | 171,72 | 228,96 | 629,64 |
| **[6,3]** | 210,24 | 315,36 | 420,48 | 1156,32 |
| **[6,4]** | 274,32 | 411,48 | 548,64 | 1508,76 |
| **[6,5]** | 314,64 | 471,96 | 629,28 | 1730,52 |
| **[7,3]** | 419,76 | 629,64 | 839,52 | 2308,68 |
| **[7,4]** | 516,96 | 775,44 | 1033,92 | 2843,28 |
| **[7,5]** | 599,76 | 899,64 | 1199,52 | 3298,68 |

Table 5.15: Area ($\mu m^2$) of LUT_Q_par

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 120,24 | 180,36 | 240,48 | 661,32 |
| **[5,3]** | 124,56 | 186,84 | 249,12 | 685,08 |
| **[5,4]** | 218,88 | 328,32 | 437,76 | 1203,84 |
| **[6,3]** | 128,88 | 193,32 | 257,76 | 708,84 |
| **[6,4]** | 223,2 | 334,8 | 446,4 | 1227,6 |
| **[6,5]** | 466,56 | 699,84 | 933,12 | 2566,08 |
| **[7,3]** | 133,2 | 199,8 | 266,4 | 732,6 |
| **[7,4]** | 227,52 | 341,28 | 455,04 | 1251,36 |
| **[7,5]** | 470,88 | 706,32 | 941,76 | 2589,84 |

Table 5.16: Area ($\mu m^2$) of Thermo_Q_par

We can observe from the tables that the values of the indicated area are proportional to the values shown in table 5.1 where the area of the single element in serial configuration was studied. This data is consistent with the structure because the parallel architecture does not modify the quantization and reconstruction blocks that are instead duplicated according to the value of the degree. For this reason, it is observable as for grade 2 the area doubles, triples for grade 3 and so on.

As in the Serial case, it is observed that there is a correlation that highlight a greater area consumption by the THERMO component when the number of parameter bits is reduced.

When the bit-width value indicated by N is contained, the LUT maintain a limited area. However, as the value of wordWidth increases, the impact of comparators becomes more significant, but remains less relevant compared to the growth of LUTs with N.
Starting from $N = 6$, the THERMO block proves to be more efficient for all wordWidth configurations, except for the configuration equal to [6,5]. In this case, there is a significant deterioration in the area, and the use of LUTs becomes relevant again. For $N = 7$ or higher values, it is convenient to use THERMO blocks for every wordWidth value.

We proceed with the same study on the reconstruction block that also in this case shows a predictable trend with the observations already explained. The stability of the THERMO model turns out to be an excellent general purpose solution and that keeps a complexity almost constant, but not always the best, especially for low parameter values where the configuration with lookup table remains preferable.
We have reported the graph in figure 5.8 while the data are available in tables 5.17 for LUT and 5.18 for THERMO.

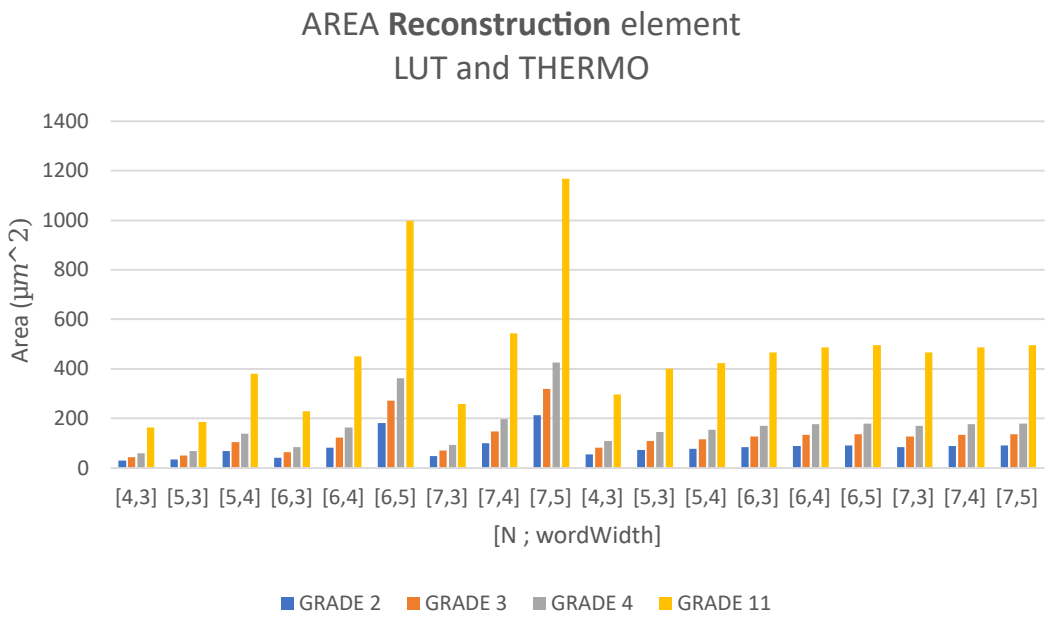


Figure 5.8: Relationship of Area between LUT (left) and THERMO (right) Parallel Reconstruction block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 29,52 | 44,28 | 59,04 | 162,36 |
| **[5,3]** | 33,84 | 50,76 | 67,68 | 186,12 |
| **[5,4]** | 69,12 | 103,68 | 138,24 | 380,16 |
| **[6,3]** | 41,76 | 62,64 | 83,52 | 229,68 |
| **[6,4]** | 82,08 | 123,12 | 164,16 | 451,44 |
| **[6,5]** | 181,44 | 272,16 | 362,88 | 997,92 |
| **[7,3]** | 46,8 | 70,2 | 93,6 | 257,4 |
| **[7,4]** | 98,64 | 147,96 | 197,28 | 542,52 |
| **[7,5]** | 212,4 | 318,6 | 424,8 | 1168,2 |

Table 5.17: Area ($\mu m^2$) of LUT_R_par

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 54 | 81 | 108 | 297 |
| **[5,3]** | 72,72 | 109,08 | 145,44 | 399,96 |
| **[5,4]** | 77,04 | 115,56 | 154,08 | 423,72 |
| **[6,3]** | 84,96 | 127,44 | 169,92 | 467,28 |
| **[6,4]** | 88,56 | 132,84 | 177,12 | 487,08 |
| **[6,5]** | 90 | 135 | 180 | 495 |
| **[7,3]** | 84,96 | 127,44 | 169,92 | 467,28 |
| **[7,4]** | 88,56 | 132,84 | 177,12 | 487,08 |
| **[7,5]** | 90 | 135 | 180 | 495 |

Table 5.18: Area ($\mu m^2$) of Thermo_R_par

At this point, we will examine the contribution of each element within the VN in order to calculate the value of the total area. This evaluation will take into account both the quantization and reconstruction blocks for both architectures.
The graph is observable figure 5.9, the corresponding tables are the 5.19 for the LUT and the 5.20 for the THERMO.

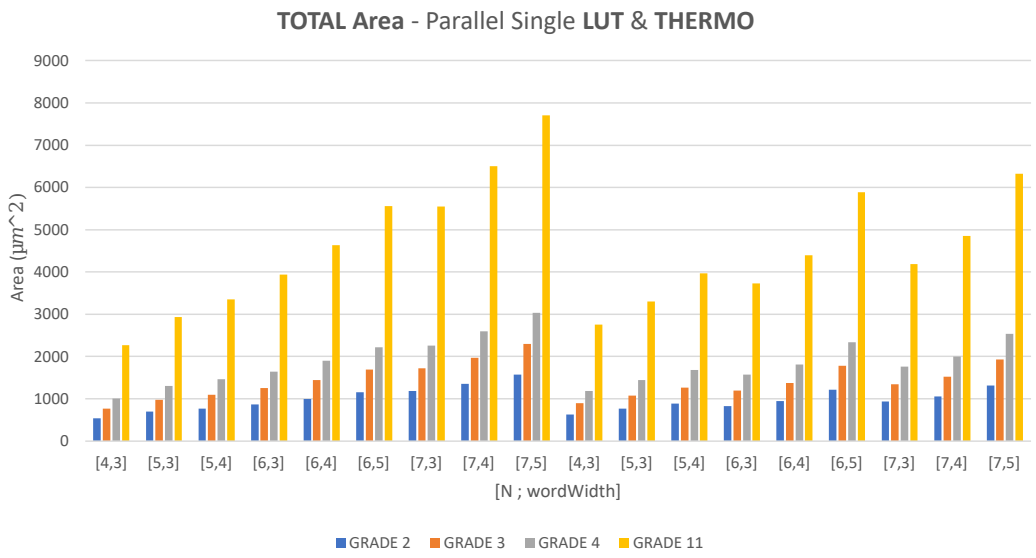


Figure 5.9: Relationship of Parallel Total Area between VN using LUT (left) and THERMO (right) blocks

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 536,76 | 762,84 | 1008 | 2268 |
| **[5,3]** | 695,52 | 977,76 | 1307,88 | 2937,24 |
| **[5,4]** | 770,4 | 1091,16 | 1457,64 | 3350,16 |
| **[6,3]** | 867,24 | 1254,6 | 1644,12 | 3939,48 |
| **[6,4]** | 993,24 | 1444,68 | 1896,12 | 4637,16 |
| **[6,5]** | 1154,52 | 1686,6 | 2218,68 | 5557,68 |
| **[7,3]** | 1179 | 1716,84 | 2257,2 | 5549,4 |
| **[7,4]** | 1349,64 | 1973,88 | 2598,48 | 6504,12 |
| **[7,5]** | 1567,8 | 2302,2 | 3034,8 | 7705,8 |

Table 5.19: Tot.Area ($\mu m^2$) - LUT_par

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 626,04 | 895,68 | 1185,12 | 2757,96 |
| **[5,3]** | 762,48 | 1078,2 | 1441,8 | 3305,52 |
| **[5,4]** | 882,72 | 1259,64 | 1682,28 | 3967,92 |
| **[6,3]** | 829,08 | 1197,36 | 1567,8 | 3729 |
| **[6,4]** | 948,6 | 1377,72 | 1806,84 | 4391,64 |
| **[6,5]** | 1215 | 1777,32 | 2339,64 | 5890,32 |
| **[7,3]** | 930,6 | 1344,24 | 1760,4 | 4183,2 |
| **[7,4]** | 1050,12 | 1524,6 | 1999,44 | 4856,76 |
| **[7,5]** | 1316,52 | 1925,28 | 2532,24 | 6323,76 |

Table 5.20: Tot.Area ($\mu m^2$) - Thermo_par

Looking at the complete picture it is possible to observe how the THERMO and LUT configuration influence the study of the VN, considering also all the other elements of which it is composed, in terms of resources spent when the parameters grow. Starting from [6,3] it has a smaller area than the lookup tables, while below this value, the latter offer a better optimization of the space than the comparators. The isolated case remains [6,5] which, as the graph shows, prefers the use of LUTs due to the strong dependence of the area of comparators for the wordWidth parameter. However this trend disappear in the case [7,5] where, with the same wordWidth, the area of the LUT explodes due to its exponential dependence with the bit-width of input shown by N.

In the parallel configuration there is a greater number of quantization and reconstruction elements than the serial architecture, due to the degree under observation. The study of overhead is interesting to analyze since it shows how important the contributions of R-Q are compared with the rest of the logic and how much they are more predominant than the serial case, as shown in the table 5.7 and 5.8. For the parallel case the tables are listed below, the 5.21 for the LUT and 5.22 for the THERMO.

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 16,0% | 16,8% | 17,0% | 20,8% |
| **[5,3]** | 18,7% | 20,0% | 19,9% | 24,4% |
| **[5,4]** | 23,8% | 25,2% | 25,2% | 30,1% |
| **[6,3]** | 29,1% | 30,1% | 30,7% | 35,2% |
| **[6,4]** | 35,9% | 37,0% | 37,6% | 42,3% |
| **[6,5]** | 43,0% | 44,1% | 44,7% | 49,1% |
| **[7,3]** | 39,6% | 40,8% | 41,3% | 46,2% |
| **[7,4]** | 45,6% | 46,8% | 47,4% | 52,1% |
| **[7,5]** | 51,8% | 52,9% | 53,5% | 58,0% |

Table 5.21: totLUTarea/totArea (%)

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 27,8% | 29,2% | 29,4% | 34,7% |
| **[5,3]** | 25,9% | 27,4% | 27,4% | 32,8% |
| **[5,4]** | 33,5% | 35,2% | 35,2% | 41,0% |
| **[6,3]** | 25,8% | 26,8% | 27,3% | 31,5% |
| **[6,4]** | 32,9% | 33,9% | 34,5% | 39,0% |
| **[6,5]** | 45,8% | 47,0% | 47,6% | 52,0% |
| **[7,3]** | 23,4% | 24,3% | 24,8% | 28,7% |
| **[7,4]** | 30,1% | 31,1% | 31,6% | 35,8% |
| **[7,5]** | 42,6% | 43,7% | 44,3% | 48,8% |

Table 5.22: totTHERMOarea/totArea (%)

As expected, the insertion of additional R-Q blocks has significantly increased the overall area, especially with the increasing of degree levels, up to reaching a 50% occupation of the entire structure. This result is significant as it allows some trade-off analysis and permit to do specific design choices based on desired functionality objective and the area we wish to allocate, depending on the available space.

Finally, we observe the results provided by the study of timing, identifying the period of the critical path, and therefore the working frequency of the different structures. This path is the same of the Serial architecture, this means that the quantization branch is involved.
Differently from the Serial configuration, it is observed that the degree variation causes an increase in logic and critical path. In particular, since the quantization path coincides with the critical path, we can notice an increase in the number of subtractors, necessary for performing operations, which however results in a higher fan-out for the register containing the data to be subtracted. This leads to an increase in the period required to do the operations, especially for structures with high degrees. The use of LUT is associated with an increase in period as the parameters grow, while for the THERMO solution, a more stable yet significantly faster period is observed compared to LUTs.
The results can be consulted from the graph in figure 5.10 and from the tables 5.23 for LUT and 5.24 for THERMO.

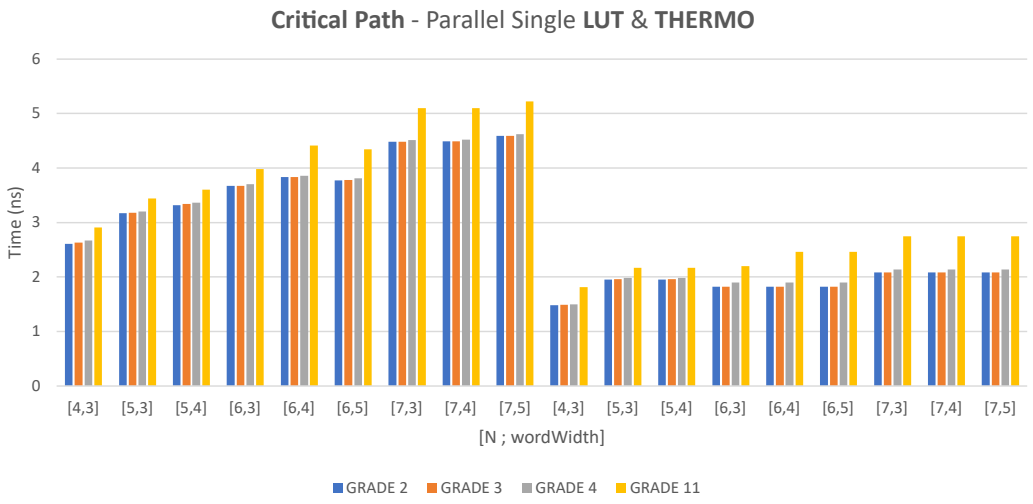


Figure 5.10: Relationship of the Critical path period ($ns$) between LUT and THERMO

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 2,61 | 2,63 | 2,67 | 2,91 |
| **[5,3]** | 3,17 | 3,18 | 3,2 | 3,44 |
| **[5,4]** | 3,32 | 3,34 | 3,36 | 3,6 |
| **[6,3]** | 3,67 | 3,67 | 3,7 | 3,98 |
| **[6,4]** | 3,83 | 3,83 | 3,86 | 4,41 |
| **[6,5]** | 3,77 | 3,78 | 3,81 | 4,34 |
| **[7,3]** | 4,48 | 4,48 | 4,51 | 5,1 |
| **[7,4]** | 4,49 | 4,49 | 4,52 | 5,1 |
| **[7,5]** | 4,59 | 4,59 | 4,62 | 5,22 |

Table 5.23: Critical Path ($ns$) LUT_par

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 1,48 | 1,49 | 1,5 | 1,81 |
| **[5,3]** | 1,95 | 1,96 | 1,98 | 2,17 |
| **[5,4]** | 1,95 | 1,96 | 1,98 | 2,17 |
| **[6,3]** | 1,82 | 1,82 | 1,9 | 2,2 |
| **[6,4]** | 1,82 | 1,82 | 1,9 | 2,46 |
| **[6,5]** | 1,82 | 1,82 | 1,9 | 2,46 |
| **[7,3]** | 2,08 | 2,08 | 2,14 | 2,75 |
| **[7,4]** | 2,08 | 2,08 | 2,14 | 2,75 |
| **[7,5]** | 2,08 | 2,08 | 2,14 | 2,75 |

Table 5.24: Critical Path ($ns$) Thermo_par

### Net

The Net architecture is also studied for the previously shown parallel configurations, following the same definition of the interconnection matrix.
The results of the synthesis are shown in figure 5.11 where we can observe the cost in terms of area occupied by the two configurations under examination. The numerical data are collected within table 5.25 for LUT and in table 5.26 for THERMO.

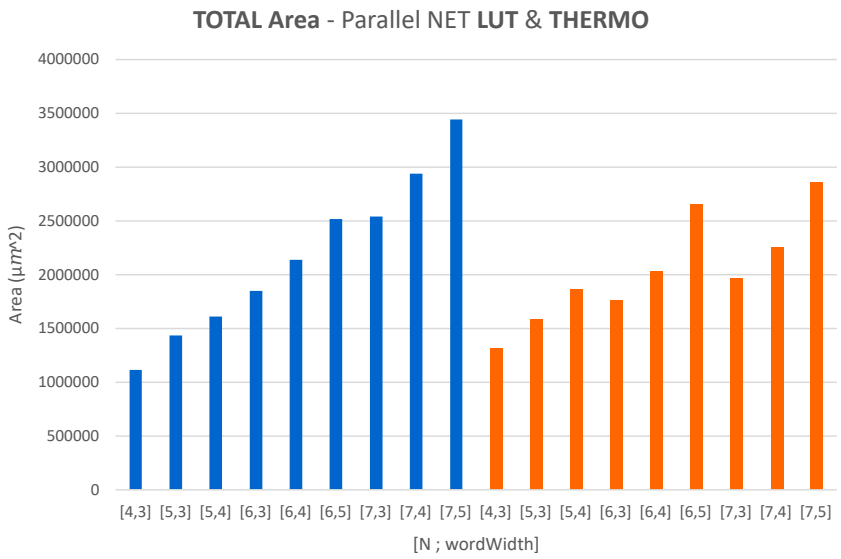


Figure 5.11: Relationship of Total Area between NET VN using LUT (blue) and THERMO (orange) blocks

| [N, wordWidth] | Total Area ($\mu m^2$) |
|---|---|
| [4,3] | 1115446,33 |
| [5,3] | 1435408,58 |
| [5,4] | 1611074,9 |
| [6,3] | 1848229,94 |
| [6,4] | 2140250,43 |
| [6,5] | 2518557,15 |
| [7,3] | 2542159,47 |
| [7,4] | 2939762,2 |
| [7,5] | 3444081,88 |

Table 5.25: Total Area for NET LUT_par

| [N, wordWidth] | Total Area ($\mu m^2$) |
|---|---|
| [4,3] | 1319395,69 |
| [5,3] | 1588947,85 |
| [5,4] | 1868624,66 |
| [6,3] | 1760729,05 |
| [6,4] | 2037890,89 |
| [6,5] | 2657237,78 |
| [7,3] | 1972578,2 |
| [7,4] | 2252962,8 |
| [7,5] | 2867896,8 |

Table 5.26: Total Area for NET Thermo_par

The results highlight a proof of the trend observed in previous architectures, with an advantage in using LUT compared to THERMO comparators for low values of N, which affect the weight of the LUT. As N increases, the THERMO block demonstrates greater effectiveness in resource utilization, highlighting the value [6,5], which represents the transition point of the two trends. For N equal to 6, it is advantageous to use the THERMO block , but the value of wordWidth needs to remain below 5.
The analysis of overall occupation reveals excessively high data, with area values over the three square millimeters, making it challenging to apply the Net architecture, especially for a flooded approach for decoding.

Finally, it is represented for the Net structure the resulting critical path that determines the period. As shown in the study of individual parallel elements, there is an improvement in performance thanks to the use of THERMO blocks that lead to reduced periods remaining almost stable to the change of parameters, going against the increase given by the LUTs.

The graph showing the results is represented in figure 5.12. To better consult the numerical values, we report below the tables for the LUT in 5.25 and for the THERMO in 5.26.

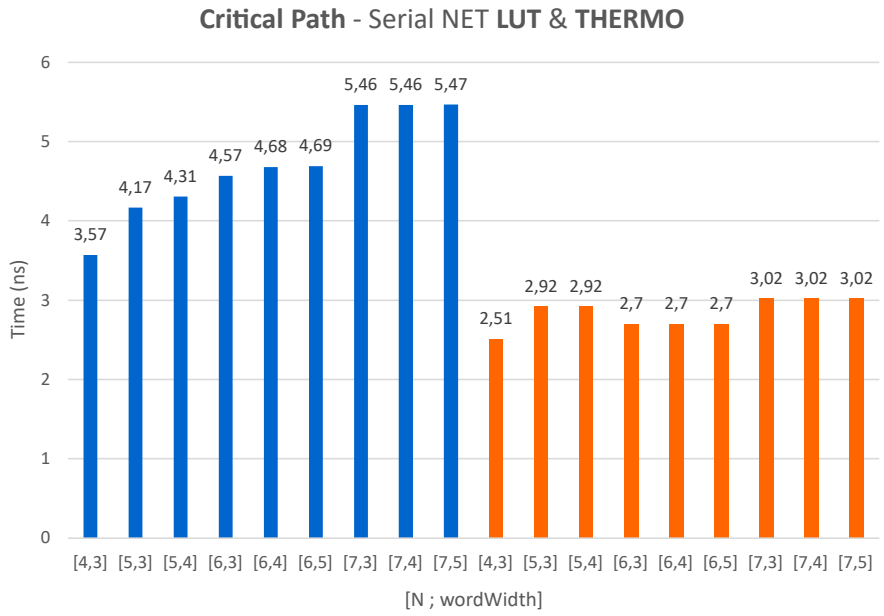


Figure 5.12: Relationship of the Critical Path period ($ns$) between LUT (blue) and THERMO (orange)

| [N, wordWidth] | Critical path ($ns$) |
|---|---|
| **[4,3]** | 3,57 |
| **[5,3]** | 4,17 |
| **[5,4]** | 4,31 |
| **[6,3]** | 4,57 |
| **[6,4]** | 4,68 |
| **[6,5]** | 4,69 |
| **[7,3]** | 5,46 |
| **[7,4]** | 5,46 |
| **[7,5]** | 5,47 |

Table 5.27: Crit-path for NET LUT_par

| [N, wordWidth] | Critical path ($ns$) |
|---|---|
| **[4,3]** | 2,51 |
| **[5,3]** | 2,92 |
| **[5,4]** | 2,92 |
| **[6,3]** | 2,7 |
| **[6,4]** | 2,7 |
| **[6,5]** | 2,7 |
| **[7,3]** | 3,02 |
| **[7,4]** | 3,02 |
| **[7,5]** | 3,02 |

Table 5.28: Crit-path for NET Thermo_par

### 5.1.3 Trade-off between Serial and Parallel

So far we have evaluated the performance of various Variable Node configurations, focusing on the differences between the use of lookup tables and THERMO comparators. In this section we want to observe the results of these R-Q blocks, but comparing the serial and parallel implementation method, in order to observe their strengths and weaknesses.

The serial configuration has a slight dependency of the area with the grade, which only changes the size of the memory element. This dependence on the area becomes more marked with the parallel configuration given the multiple instantiation of the quantization and reconstruction elements.

This study collides with the timing defined by the critical path that define the working frequency, which may be better for low degree values given by the greater simplicity of the structure than the serial, or worse if the degree increase too much, leading to an growth in combinatorial logic and fan-out of some circuit components. However, it is important to underline the value of latency since this corresponds to a great advantage of the parallel configuration that allows its use above the serial one despite the huge increase in area. Performing operations in parallel allows to obtain a high value of throughput and especially low latency, which balances the problem of the area. On the contrary, the serial architecture involves a high latency, having to carry out the operations in series.

Depending on the goal you want to achieve from the structure, it may be useful to evaluate what type of approach is worth following. Alternatively, the trade-off of the listed structures is studied in order to choose the most suitable configuration.

#### Single

The first comparison that is possible to observe is the total area occupied between the Serial and Parallel architecture for LUT in figure 5.13 and THERMO in figure 5.14.

The comparison in general terms are predictable for the observations made during this chapter, despite this, the data are reported to highlight the orders of magnitude. The resources used grow in a linear way according to the degree studied, and is possible to see how expensive is the space used for high-grade elements.

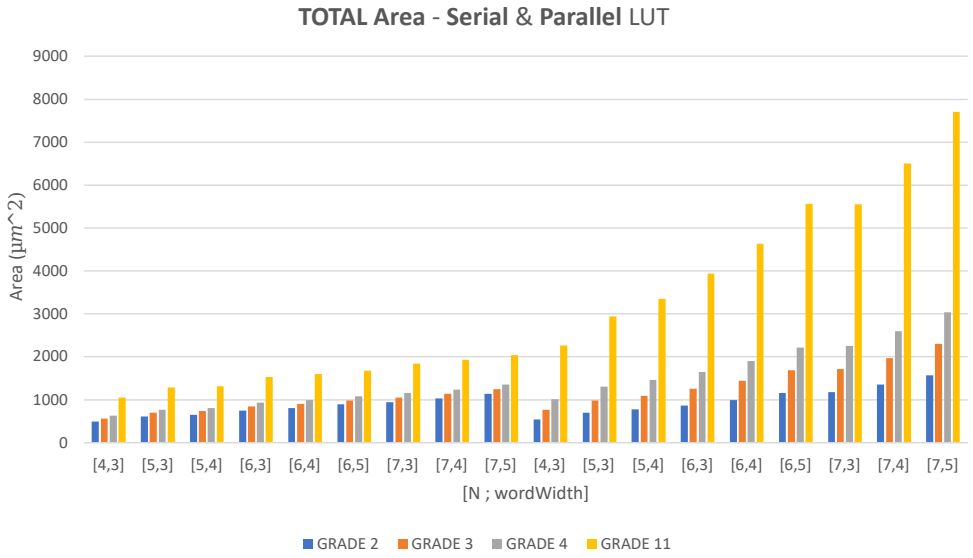


Figure 5.13: Relationship of Total Area between Serial (left) and Parallel (right) configurations - LUT block

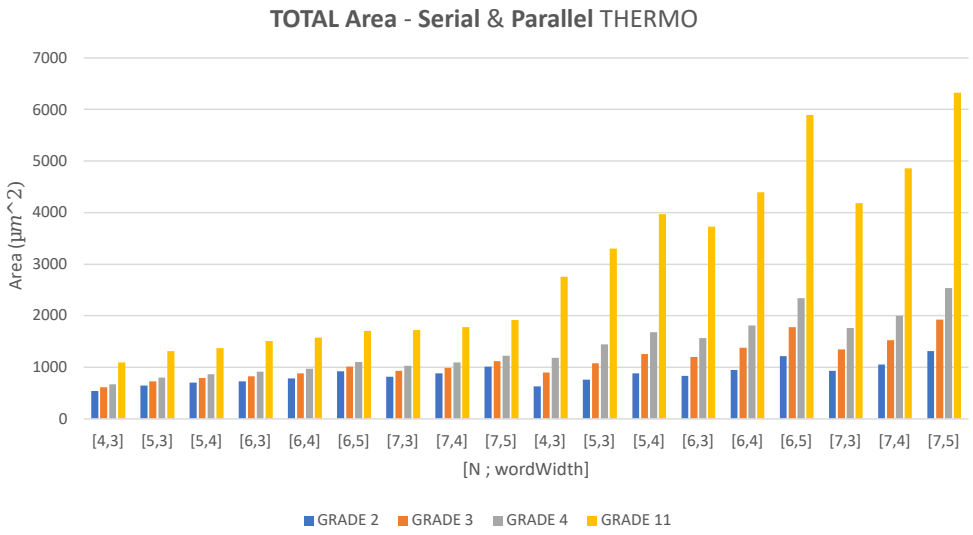


Figure 5.14: Relationship of Total Area between Serial (left) and Parallel (right) configurations - THERMO block

This result can also lead to another possibility. The idea may be to use another degree profile, using a parity check matrix with a different code rate from $R = 0.5$.
Keeping the number of Variable Node always equal to $N = 1296$, it is possible to modify the code rate, and put it for example equal to $R = 0.75$, that allows to get a distribution of degrees [7] as observable in figure 5.15.

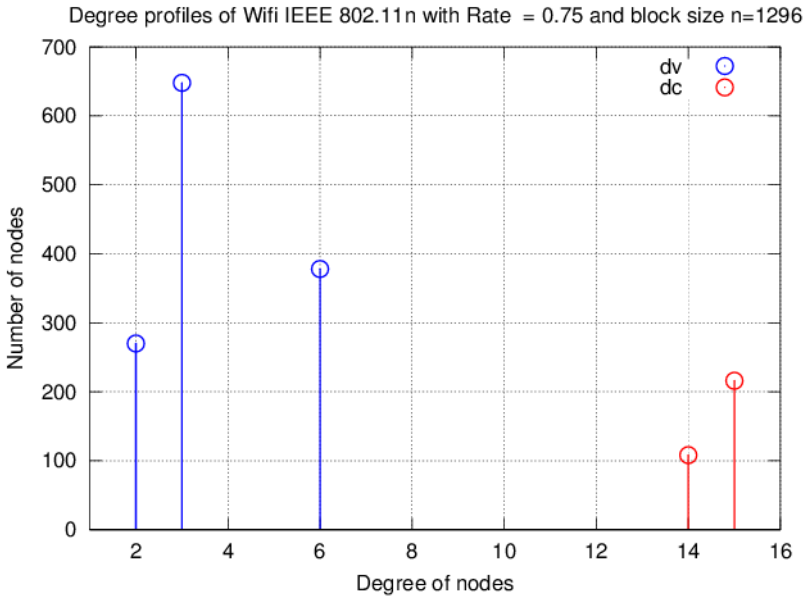


Figure 5.15: Degree profile for Rate = 0.75. VN in blue, CN in red [7]

As is possible to see, in this case the grade of the VN would be distributed between 2, 3 and 6 eliminating the problematic degree 11. This case is not studied in this thesis but the insertion of the above graphs allows a critical analysis of the function that we want to obtain and therefore of the circuit that realizes it.

The same type of analysis is carried out on the critical path. This, highlighted by all the involved structures, remains unchanged, allowing the comparison of results between different implementations.
As already seen in the previous sections, in the serial case there is no differentiation of the period as the degree changes because this do not affect the identified path that determines the working frequency. However, we note that the timing between the two configurations that are using the same type of blocks (LUT or THERMO) are almost identical for equal parameters, with the only difference of the degree 11 that in the parallel configuration introduces a greater delay because of the tree of subtractors that is present between the two nodes of the critical path.
The graphs are shown in figure 5.16 for LUT and 5.17 for THERMO.

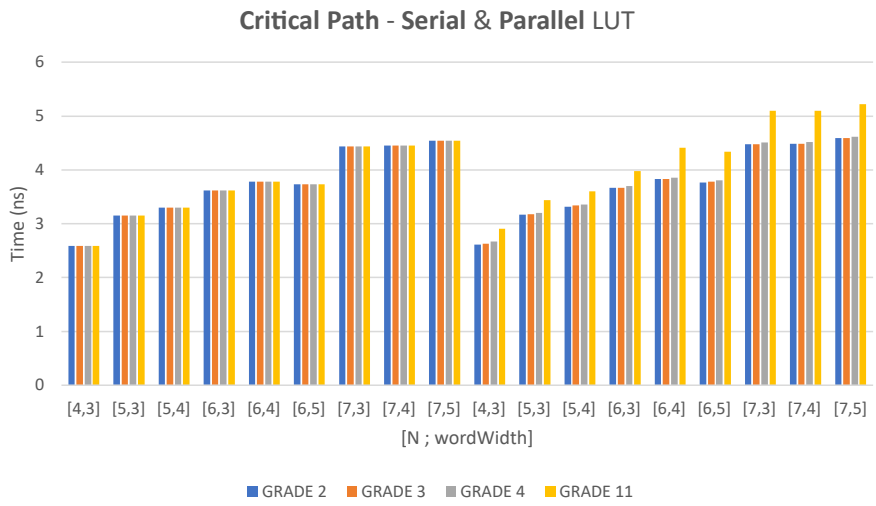


Figure 5.16: Relationship of the Critical path period ($ns$) between Serial (left) and Parallel (right) - LUT block

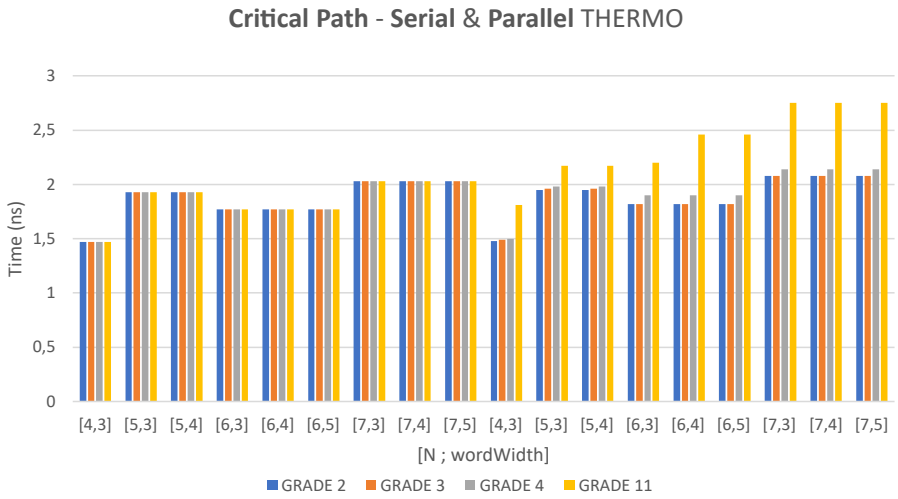


Figure 5.17: Relationship of the Critical path period ($ns$) between Serial (left) and Parallel (right) - THERMO block

The above data shows an higher period for parallel configuration, especially as the degree increases. However, to show the strong point of the architecture, graphs that highlight the latency value of the two solutions are reported, in figure 5.18 for LUT and in figure 5.19 for THERMO.

The use of parallel configuration introduces structural complexity. However, this method leads to a significant increase in the occupied area. However, this configuration allows the structure to process all incoming data simultaneously, providing an output with much lower latency compared to serial configuration. For applications that do not have area requirements, this solution can be advantageous as it enhances data processing speed.

The graphs below show how important this parameter is for fast applications, and it becomes even more significant when working with very high-grade VN.
Numerical latency data of serial architecture are visible in tables 5.29 for LUT and 5.30 for THERMO, which are to be compared with tables that report the critical path periods of the parallel configuration in 5.23 and 5.24, where the result is showed in the graph above in 5.18 and 5.19.

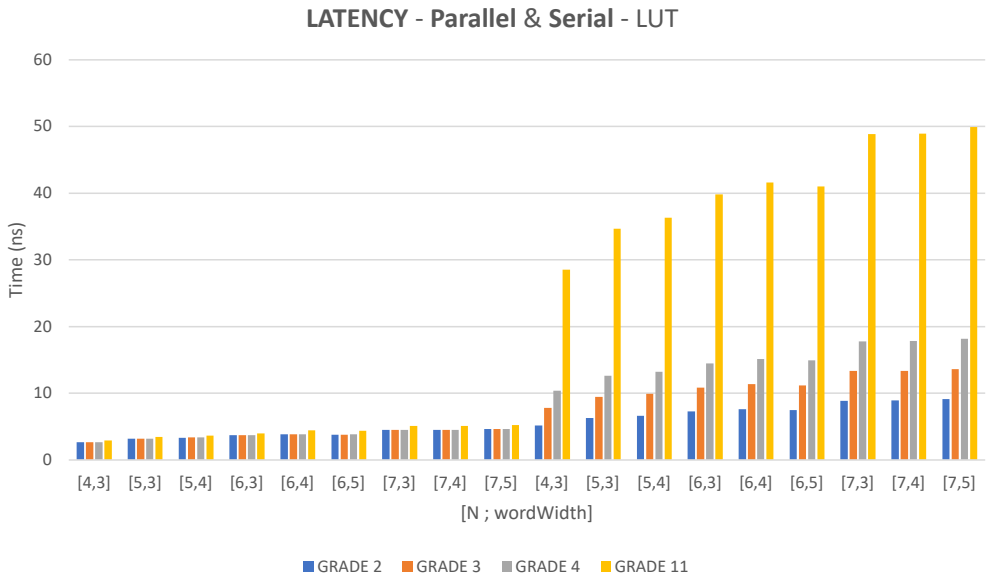


Figure 5.18: Relationship of Latency ($ns$) between Parallel (left) and Serial (right) - LUT block

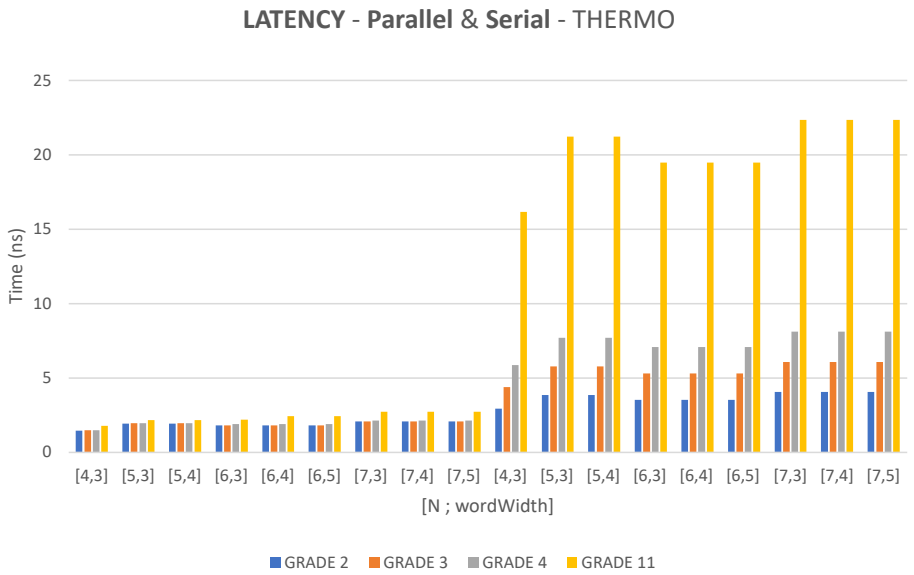


Figure 5.19: Relationship of Latency ($ns$) between Parallel (left) and Serial (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 5,18 | 7,77 | 10,36 | 28,49 |
| **[5,3]** | 6,3 | 9,45 | 12,6 | 34,65 |
| **[5,4]** | 6,6 | 9,9 | 13,2 | 36,3 |
| **[6,3]** | 7,24 | 10,86 | 14,48 | 39,82 |
| **[6,4]** | 7,56 | 11,34 | 15,12 | 41,58 |
| **[6,5]** | 7,46 | 11,19 | 14,92 | 41,03 |
| **[7,3]** | 8,88 | 13,32 | 17,76 | 48,84 |
| **[7,4]** | 8,9 | 13,35 | 17,8 | 48,95 |
| **[7,5]** | 9,08 | 13,62 | 18,16 | 49,94 |

Table 5.29: Latency ($ns$) Serial LUT

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 2,94 | 4,41 | 5,88 | 16,17 |
| **[5,3]** | 3,86 | 5,79 | 7,72 | 21,23 |
| **[5,4]** | 3,86 | 5,79 | 7,72 | 21,23 |
| **[6,3]** | 3,54 | 5,31 | 7,08 | 19,47 |
| **[6,4]** | 3,54 | 5,31 | 7,08 | 19,47 |
| **[6,5]** | 3,54 | 5,31 | 7,08 | 19,47 |
| **[7,3]** | 4,06 | 6,09 | 8,12 | 22,33 |
| **[7,4]** | 4,06 | 6,09 | 8,12 | 22,33 |
| **[7,5]** | 4,06 | 6,09 | 8,12 | 22,33 |

Table 5.30: Latency ($ns$) Serial THERMO

### Net

In this section, we proceed to show the relationships between the data obtained for serial and parallel configurations in the NET architecture. In this context, the results will demonstrate, considering the already highlighted observations, a greater area occupation by the parallel structure compared to the serial one. The results will be presented in order to highlight the level of difference that separates the two configurations.

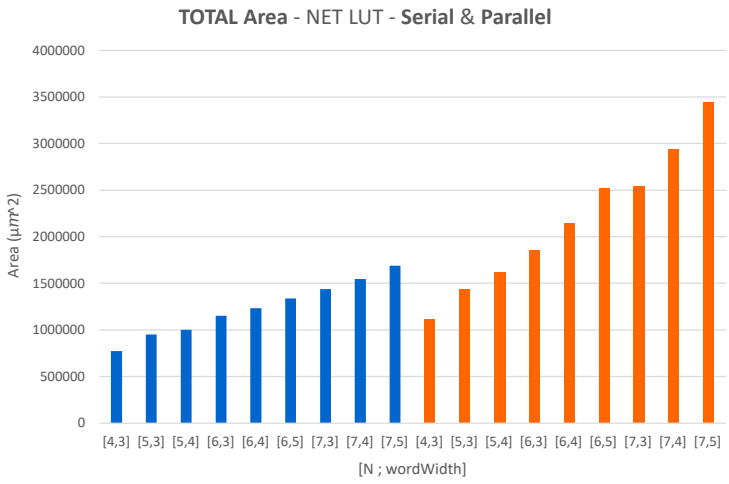


Figure 5.20: Relationship of Total NET Area between Serial (blue) and Parallel (orange) configurations - LUT block

These graphs delineate the resource consumption in terms of area in relation to the variation of parameters, allowing to evaluate the trade-off between the two structures and decide which configuration to select depending on the available space for a possible application.
From the analysis of graph in 5.20, we can observe that the maximum value of the serial configuration, corresponding to case [7,5], is reached already, in terms of area occupation, in the parallel case related to [5,4]. On the contrary, it is interesting to note how in the THERMO case in figure 5.21, the serial implementation is consistently superior to the parallel one, showing a greater resource utilization by the latter for reduced parameters.

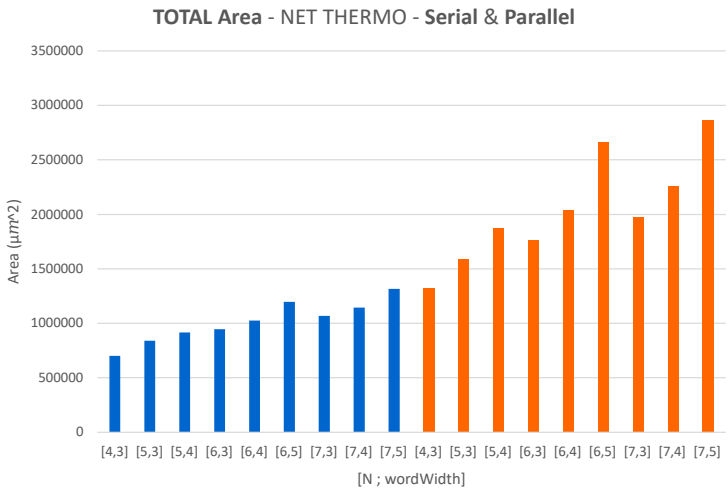


Figure 5.21: Relationship of Total NET Area between Serial (blue) and Parallel (orange) configurations - THERMO block

A similar trend is also observed in the study of the critical path, which highlights the differences in such structures. In figure 5.22, the contribution of the lookup table is shown, while in figure 5.23, the use of comparators introduced by the THERMO architecture is illustrated.

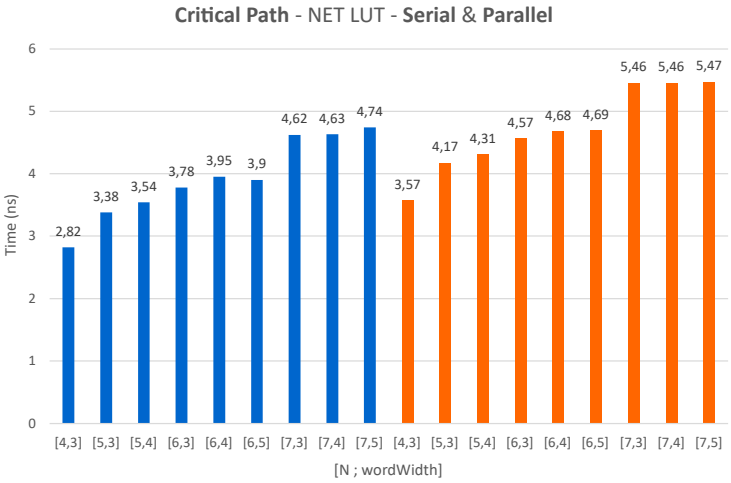


Figure 5.22: Relationship of NET Critical path period ($ns$) between Serial (blue) and Parallel (orange) - LUT block

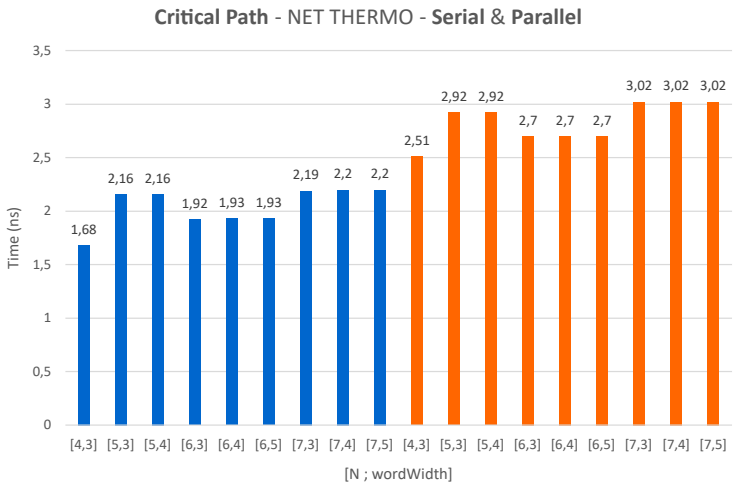


Figure 5.23: Relationship of NET Critical path period ($ns$) between Serial (blue) and Parallel (orange) - THERMO block

Similarly, we will observe a slightly longer period for the parallel configuration, due to the presence of more components along the critical path that slow down the output calculation. However, as previously mentioned, the parallel configuration offers better throughput, making it the optimal choice in terms of latency compared to the serial configuration.
For this reason, it is appropriate to consider this choice if an high-performance architecture is needed.

## 5.2 FPGA Synthesis Results

In this section, we will observe the results obtained from the synthesis of the same architectures previously studied, realized by the Intel Quartus Prime software for implementation on FPGA.
In this case, considering the loading of the circuit description in a physical board with a limited number of resources, we will observe parameters different from area occupancy, focusing instead on the number of allocated resources, in order to understand the overall usage compared to the available resources.

The FPGA contain fundamental blocks known as ALM (Adaptive Logic Module), which contain programmable logic to realize the desired circuit function. In addition to ALM, we will also examine the ALUT (Adaptive Look-Up Tables) used and the dedicated logic registers. In this specific case, the device used is the *5CGXFC7C7F23C8* from the Cyclone V board family, which has a total of 56480 ALM blocks.

Initially, we will examine the results produced by the preliminary synthesis for all three of these parameters. Then, we will analyze the results provided by the Fitter, responsible for the Place & Route operation. This process is essential during the circuit synthesis on FPGA because, given the limited number of resources on the device, the aim is to optimize the available space. In this phase, the components necessary for the desired function are placed, and in the subsequent phase, they will be connected in an optimal way to maximize resource efficiency.

The graphs presented will show the changes from the preliminary analysis to the data produced by the Fitter for all configurations previously analyzed, excluding, however, the analysis of the complete VN architecture at the decoder level with the 1296 iterations described by the H matrix. Such circuits require a high amount of resources and are impossible to use for FPGA implementation, as already highlighted by the study of total area in ASIC analysis.

Finally, we will examine the results produced by the Timing Analyzer for the *Slow 1100 mV 0 °C* model in order to evaluate the variations in working frequency in relation to the parameters [N, wordWidth], and degree. This analysis is significant because it represents the worst-case scenario, showing the maximum frequency guaranteed by the circuit under any circumstance.

After examining the optimizations performed by the Fitter, we will complete the study by observing the differences between these data for the different architectures. In particular, we will evaluate the variations in the use of LUT and THERMO blocks for quantization and reconstruction operations in Serial and Parallel structures.
Finally, we will observe the differences between Serial and Parallel architectures, considering the use of LUT or THERMO blocks.

### 5.2.1 Serial - Single

#### LUT

**ALM** modules decrease after the Place & Route operation, given the optimization that the Fitter performs. From the diagram 5.24 it is little visible since, for low degree, a variation of one or two units is observed. As the degree increases, the saving of logic becomes more marked, more visible from the tables 5.31 for Analysis and 5.32 for Fitter.

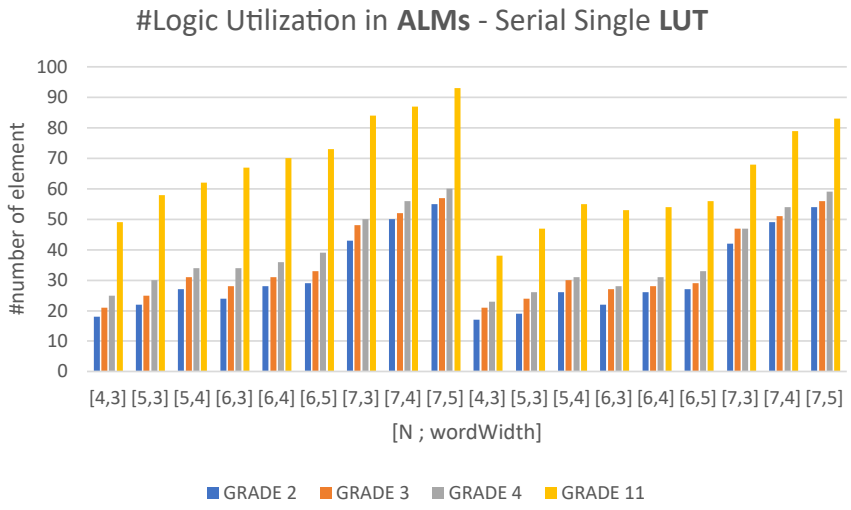


Figure 5.24: Relationship of ALMs - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 18 | 21 | 25 | 49 |
| **[5,3]** | 22 | 25 | 30 | 58 |
| **[5,4]** | 27 | 31 | 34 | 62 |
| **[6,3]** | 24 | 28 | 34 | 67 |
| **[6,4]** | 28 | 31 | 36 | 70 |
| **[6,5]** | 29 | 33 | 39 | 73 |
| **[7,3]** | 43 | 48 | 50 | 84 |
| **[7,4]** | 50 | 52 | 56 | 87 |
| **[7,5]** | 55 | 57 | 60 | 93 |

Table 5.31: ALMs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 17 | 21 | 23 | 38 |
| **[5,3]** | 19 | 24 | 26 | 47 |
| **[5,4]** | 26 | 30 | 31 | 55 |
| **[6,3]** | 22 | 27 | 28 | 53 |
| **[6,4]** | 26 | 28 | 31 | 54 |
| **[6,5]** | 27 | 29 | 33 | 56 |
| **[7,3]** | 42 | 47 | 47 | 68 |
| **[7,4]** | 49 | 51 | 54 | 79 |
| **[7,5]** | 54 | 56 | 59 | 83 |

Table 5.32: ALMs for Fitter

The same cannot be said for the **ALUT** modules used that instead increase a bit after the Fitter. The usage varies from boards used, as they depend on the physical resources that are present in the specific device family. There is therefore the possibility that, in order to achieve optimization, it is necessary to use a greater number of ALUT. The data can be found in tables 5.33 and 5.34.

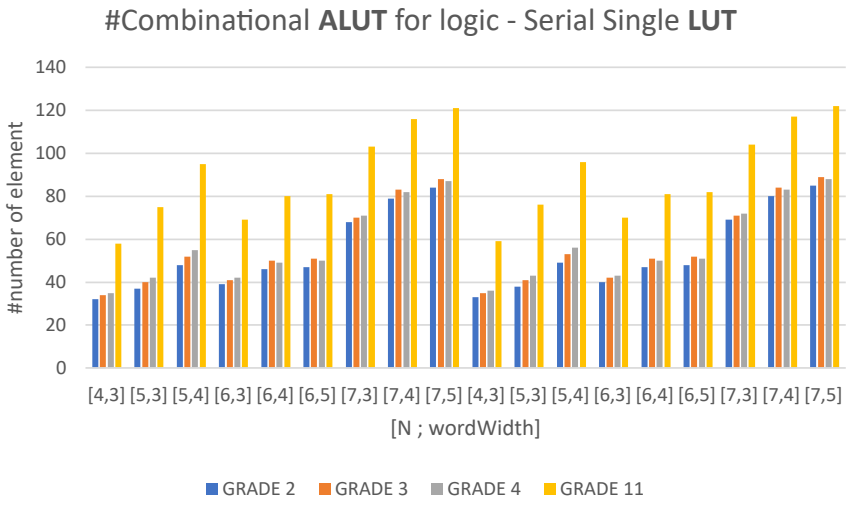


Figure 5.25: Relationship of ALUTs - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 32 | 34 | 35 | 58 |
| **[5,3]** | 37 | 40 | 42 | 75 |
| **[5,4]** | 48 | 52 | 55 | 95 |
| **[6,3]** | 39 | 41 | 42 | 69 |
| **[6,4]** | 46 | 50 | 49 | 80 |
| **[6,5]** | 47 | 51 | 50 | 81 |
| **[7,3]** | 68 | 70 | 71 | 103 |
| **[7,4]** | 79 | 83 | 82 | 116 |
| **[7,5]** | 84 | 88 | 87 | 121 |

Table 5.33: ALUTs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 33 | 35 | 36 | 59 |
| **[5,3]** | 38 | 41 | 43 | 76 |
| **[5,4]** | 49 | 53 | 56 | 96 |
| **[6,3]** | 40 | 42 | 43 | 70 |
| **[6,4]** | 47 | 51 | 50 | 81 |
| **[6,5]** | 48 | 52 | 51 | 82 |
| **[7,3]** | 69 | 71 | 72 | 104 |
| **[7,4]** | 80 | 84 | 83 | 117 |
| **[7,5]** | 85 | 89 | 88 | 122 |

Table 5.34: ALUTs for Fitter

Following the same thoughts previously made for the fitter, the number of **Dedicated Logic Register** used to perform optimization slightly increases. The tables are given in 5.35 and 5.36.

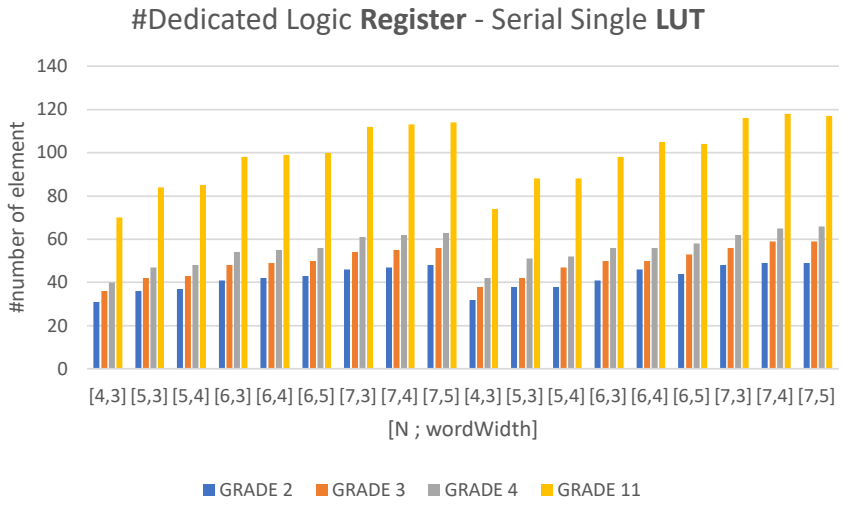


Figure 5.26: Relationship of Logic Reg - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 31 | 36 | 40 | 70 |
| **[5,3]** | 36 | 42 | 47 | 84 |
| **[5,4]** | 37 | 43 | 48 | 85 |
| **[6,3]** | 41 | 48 | 54 | 98 |
| **[6,4]** | 42 | 49 | 55 | 99 |
| **[6,5]** | 43 | 50 | 56 | 100 |
| **[7,3]** | 46 | 54 | 61 | 112 |
| **[7,4]** | 47 | 55 | 62 | 113 |
| **[7,5]** | 48 | 56 | 63 | 114 |

Table 5.35: Logic Reg for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 32 | 38 | 42 | 74 |
| **[5,3]** | 38 | 42 | 51 | 88 |
| **[5,4]** | 38 | 47 | 52 | 88 |
| **[6,3]** | 41 | 50 | 56 | 98 |
| **[6,4]** | 46 | 50 | 56 | 105 |
| **[6,5]** | 44 | 53 | 58 | 104 |
| **[7,3]** | 48 | 56 | 62 | 116 |
| **[7,4]** | 49 | 59 | 65 | 118 |
| **[7,5]** | 49 | 59 | 66 | 117 |

Table 5.36: Logic Reg for Fitter

Let us now observe the **$F_{max}$ summary** working frequencies of the various structures as the parameters taken into account change. We can observe quite variable values but that in general tend to a decrease in the frequency with the increase of the internal and external bit-widths and with a not too marked dependence on the degree. The objective of maintaining limited bit-widths is highlighted by the graph 5.27 and table 5.37 that allows to obtain higher speeds.

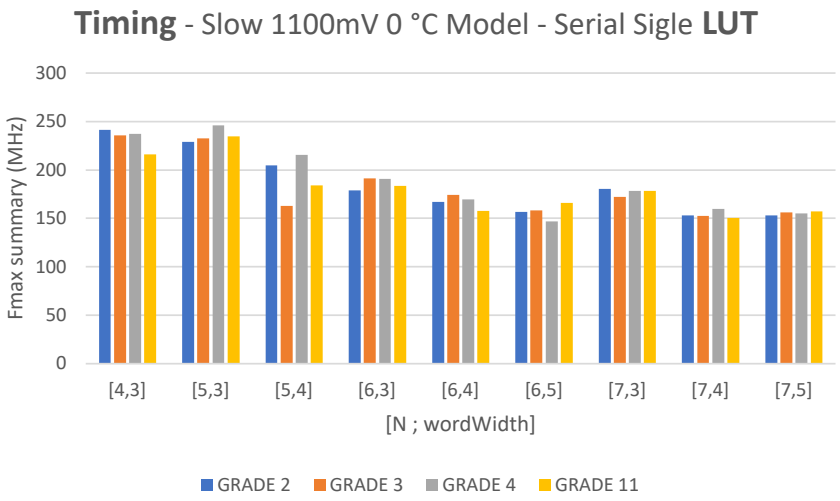


Figure 5.27: Timing Analyzer - $F_{max}$ Summary (MHz) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 241,49 | 235,68 | 237,36 | 216,08 |
| **[5,3]** | 228,68 | 232,61 | 245,7 | 234,63 |
| **[5,4]** | 204,75 | 162,76 | 215,7 | 183,89 |
| **[6,3]** | 178,95 | 191,2 | 190,48 | 183,55 |
| **[6,4]** | 167,25 | 174,19 | 169,78 | 157,88 |
| **[6,5]** | 156,49 | 158,33 | 147,06 | 165,76 |
| **[7,3]** | 180,41 | 172,38 | 178,32 | 178,19 |
| **[7,4]** | 153,33 | 152,79 | 159,97 | 150,63 |
| **[7,5]** | 153,3 | 155,98 | 155,13 | 156,99 |

Table 5.37: $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model

### THERMO

In this section we observe the same parameters for the THERMO block. We will observe that the trend of the results does not change a lot from those that use the LUT block, then it can be noted that also here, ALM modules decrease after the Place & Route operation. In this section we report the values of the **ALMs** logic for the Synthesis in table 5.38 and for the Fitter in table 5.39.

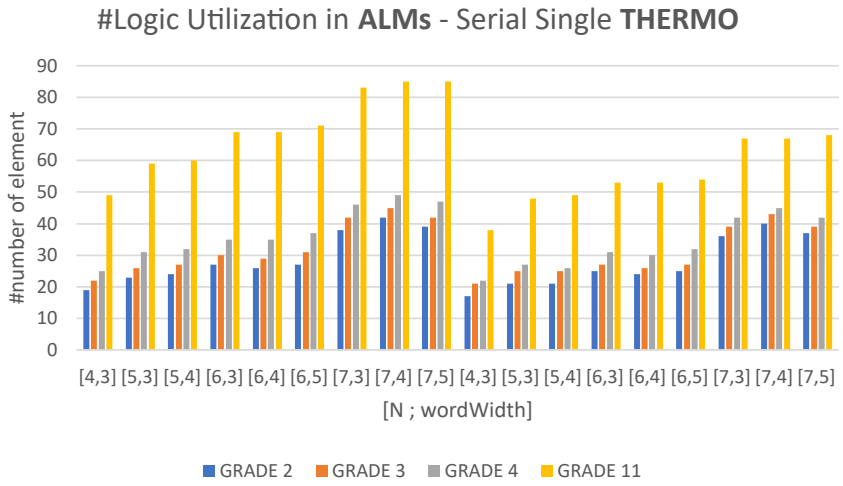


Figure 5.28: Relationship of ALMs - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 19 | 22 | 25 | 49 |
| **[5,3]** | 23 | 26 | 31 | 59 |
| **[5,4]** | 24 | 27 | 32 | 60 |
| **[6,3]** | 27 | 30 | 35 | 69 |
| **[6,4]** | 26 | 29 | 35 | 69 |
| **[6,5]** | 27 | 31 | 37 | 71 |
| **[7,3]** | 38 | 42 | 46 | 83 |
| **[7,4]** | 42 | 45 | 49 | 85 |
| **[7,5]** | 39 | 42 | 47 | 85 |

Table 5.38: ALMs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 17 | 21 | 22 | 38 |
| **[5,3]** | 21 | 25 | 27 | 48 |
| **[5,4]** | 21 | 25 | 26 | 49 |
| **[6,3]** | 25 | 27 | 31 | 53 |
| **[6,4]** | 24 | 26 | 30 | 53 |
| **[6,5]** | 25 | 27 | 32 | 54 |
| **[7,3]** | 36 | 39 | 42 | 67 |
| **[7,4]** | 40 | 43 | 45 | 67 |
| **[7,5]** | 37 | 39 | 42 | 68 |

Table 5.39: ALMs for Fitter

As for Look-Up tables, the THERMO block report a slight increase of combinational **ALUTs**, but it is reduced only to a couple of additional elements compared to the synthesis. The tables can be found below in 5.40 and in 5.41.

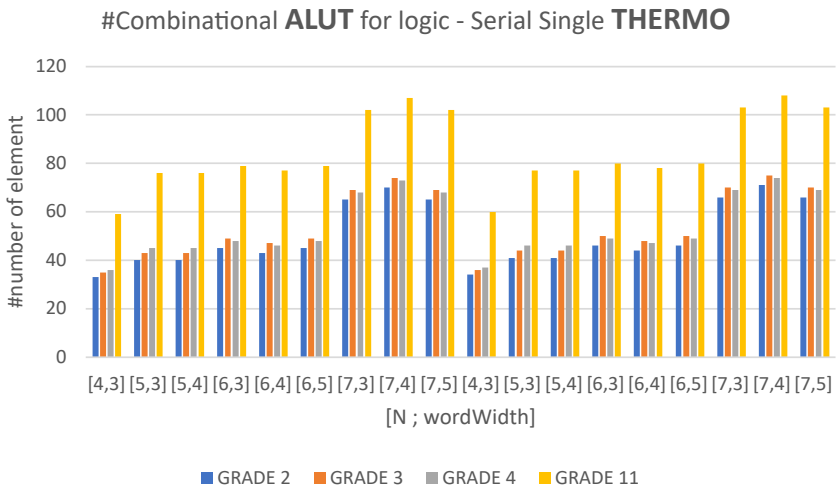


Figure 5.29: Relationship of ALUTs - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 33 | 35 | 36 | 59 |
| **[5,3]** | 40 | 43 | 45 | 76 |
| **[5,4]** | 40 | 43 | 45 | 76 |
| **[6,3]** | 45 | 49 | 48 | 79 |
| **[6,4]** | 43 | 47 | 46 | 77 |
| **[6,5]** | 45 | 49 | 48 | 79 |
| **[7,3]** | 65 | 69 | 68 | 102 |
| **[7,4]** | 70 | 74 | 73 | 107 |
| **[7,5]** | 65 | 69 | 68 | 102 |

Table 5.40: ALUTs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 34 | 36 | 37 | 60 |
| **[5,3]** | 41 | 44 | 46 | 77 |
| **[5,4]** | 41 | 44 | 46 | 77 |
| **[6,3]** | 46 | 50 | 49 | 80 |
| **[6,4]** | 44 | 48 | 47 | 78 |
| **[6,5]** | 46 | 50 | 49 | 80 |
| **[7,3]** | 66 | 70 | 69 | 103 |
| **[7,4]** | 71 | 75 | 74 | 108 |
| **[7,5]** | 66 | 70 | 69 | 103 |

Table 5.41: ALUTs for Fitter

Based on the earlier considerations done during the fitter analysis, there is a slight increase in the number of **Dedicated Logic Register** utilized for optimization. This can be visible in tables 5.42 and 5.43.

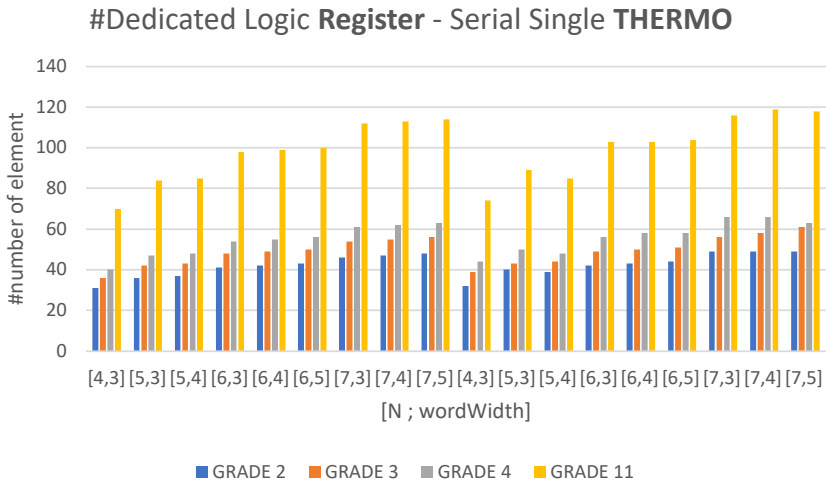


Figure 5.30: Relationship of Logic Reg - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 31 | 36 | 40 | 70 |
| **[5,3]** | 36 | 42 | 47 | 84 |
| **[5,4]** | 37 | 43 | 48 | 85 |
| **[6,3]** | 41 | 48 | 54 | 98 |
| **[6,4]** | 42 | 49 | 55 | 99 |
| **[6,5]** | 43 | 50 | 56 | 100 |
| **[7,3]** | 46 | 54 | 61 | 112 |
| **[7,4]** | 47 | 55 | 62 | 113 |
| **[7,5]** | 48 | 56 | 63 | 114 |

Table 5.42: Logic Reg for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 32 | 39 | 44 | 74 |
| **[5,3]** | 40 | 43 | 50 | 89 |
| **[5,4]** | 39 | 44 | 48 | 85 |
| **[6,3]** | 42 | 49 | 56 | 103 |
| **[6,4]** | 43 | 50 | 58 | 103 |
| **[6,5]** | 44 | 51 | 58 | 104 |
| **[7,3]** | 49 | 56 | 66 | 116 |
| **[7,4]** | 49 | 58 | 66 | 119 |
| **[7,5]** | 49 | 61 | 63 | 118 |

Table 5.43: Logic Reg for Fitter

Now we observe the **$F_{max}$ summary** working frequencies across different structures as the considered parameters change. We notice a range of values, but overall, there is a trend for working frequency to decrease with the increasing of internal and external bit-widths, with a not define dependence on the degree. The objective of maintaining limited bit-widths is highlighted also for the THERMO solutions by the graph 5.31 and table 5.44 that allows to obtain higher speeds.

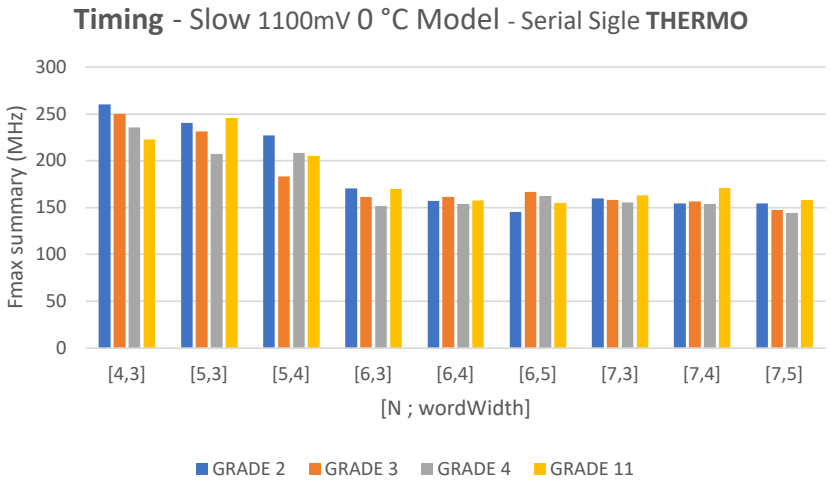


Figure 5.31: Timing Analyzer - $F_{max}$ Summary (MHz) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 260,35 | 250,25 | 235,46 | 222,87 |
| **[5,3]** | 240,62 | 231,54 | 207,3 | 245,7 |
| **[5,4]** | 227,17 | 183,49 | 208,25 | 205,38 |
| **[6,3]** | 170,59 | 161,08 | 151,63 | 169,75 |
| **[6,4]** | 157,01 | 161,5 | 153,61 | 157,68 |
| **[6,5]** | 145,52 | 166,94 | 162,39 | 154,85 |
| **[7,3]** | 159,69 | 157,93 | 155,55 | 162,76 |
| **[7,4]** | 154,13 | 156,59 | 154,01 | 170,77 |
| **[7,5]** | 154,39 | 147,62 | 144,43 | 158,33 |

Table 5.44: $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model

### 5.2.2 Parallel - Single

#### LUT

In this section we continue the analysis with the parallel configuration for both LUT and THERMO blocks. The observed logic is similar for the same change of metrics.
The parallel case reflects the observations made in the ASIC study, for this reason we observe an increase of the logic utilization **ALMs** necessary to achieve higher grade VN and higher bit-width. Figure 5.32 shows how challenging it is to manage grade 11 for both Analysis and Fitter strategies, which show a little variation between them. The results are most visible in Tables 5.45 and 5.46.

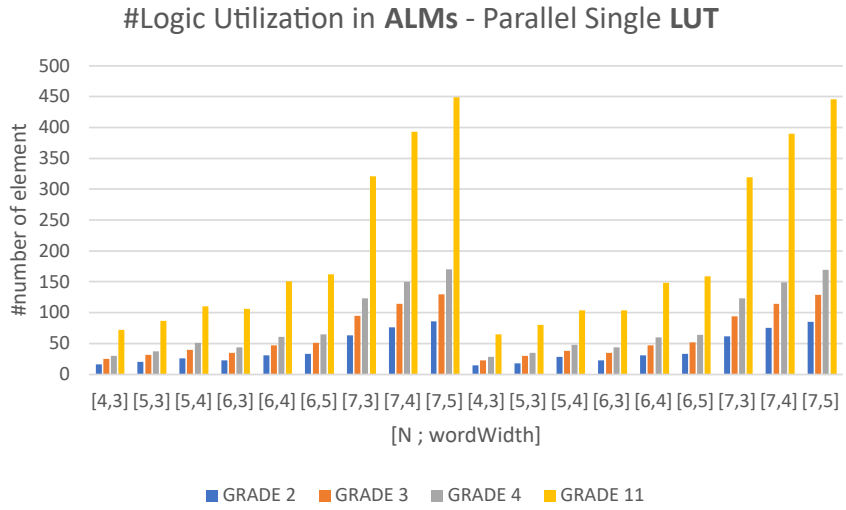


Figure 5.32: Relationship of ALMs - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 16 | 25 | 30 | 72 |
| **[5,3]** | 20 | 32 | 37 | 87 |
| **[5,4]** | 26 | 40 | 51 | 110 |
| **[6,3]** | 23 | 35 | 44 | 106 |
| **[6,4]** | 31 | 47 | 61 | 151 |
| **[6,5]** | 33 | 51 | 65 | 162 |
| **[7,3]** | 63 | 95 | 123 | 321 |
| **[7,4]** | 76 | 114 | 150 | 393 |
| **[7,5]** | 86 | 130 | 170 | 449 |

Table 5.45: ALMs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 15 | 23 | 28 | 65 |
| **[5,3]** | 18 | 30 | 35 | 80 |
| **[5,4]** | 28 | 38 | 48 | 104 |
| **[6,3]** | 23 | 35 | 44 | 104 |
| **[6,4]** | 31 | 47 | 60 | 148 |
| **[6,5]** | 33 | 52 | 64 | 159 |
| **[7,3]** | 62 | 94 | 123 | 319 |
| **[7,4]** | 75 | 114 | 149 | 390 |
| **[7,5]** | 85 | 129 | 169 | 446 |

Table 5.46: ALMs for Fitter

The argument for the parallel configuration that uses the look-up tables is similar to what is said in the serial section but with the same complexity of **ALUTs** for increasing degree introduced by the parallelization of the structure. The proof is seen in the tables below in 5.47 and 5.48 and in the graph in figure 5.33.

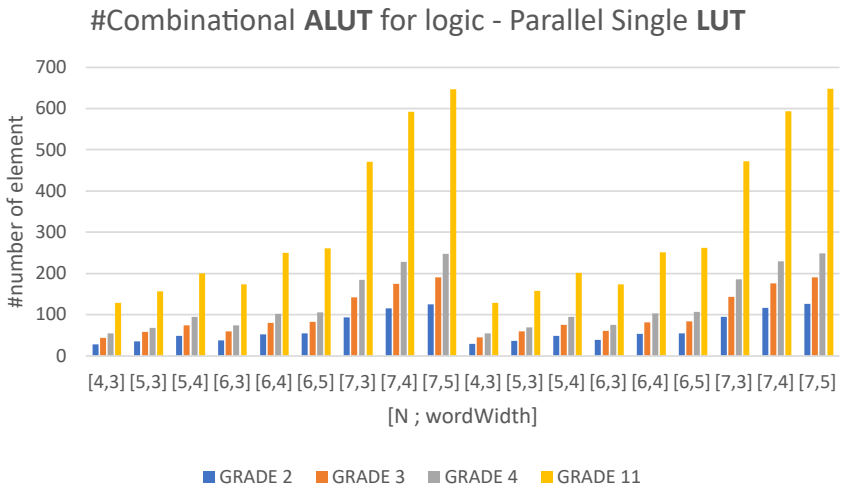


Figure 5.33: Relationship of ALUTs - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 28 | 44 | 54 | 128 |
| **[5,3]** | 35 | 58 | 68 | 157 |
| **[5,4]** | 48 | 74 | 94 | 200 |
| **[6,3]** | 38 | 59 | 74 | 173 |
| **[6,4]** | 52 | 80 | 102 | 250 |
| **[6,5]** | 54 | 83 | 106 | 261 |
| **[7,3]** | 93 | 142 | 184 | 471 |
| **[7,4]** | 115 | 175 | 228 | 592 |
| **[7,5]** | 125 | 190 | 248 | 647 |

Table 5.47: ALUTs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 29 | 45 | 55 | 129 |
| **[5,3]** | 36 | 59 | 69 | 158 |
| **[5,4]** | 49 | 75 | 95 | 201 |
| **[6,3]** | 39 | 60 | 75 | 174 |
| **[6,4]** | 53 | 81 | 103 | 251 |
| **[6,5]** | 55 | 84 | 107 | 262 |
| **[7,3]** | 94 | 143 | 185 | 472 |
| **[7,4]** | 116 | 176 | 229 | 593 |
| **[7,5]** | 126 | 191 | 249 | 648 |

Table 5.48: ALUTs for Fitter

Unlike all the other cases, the **Dedicated Logic Registers** are almost identical both for the Analysis and for the optimization of the Fitter, showing therefore that from this point of view it is not possible to optimize the already instantiated resources. These results are visible in tables 5.49 and 5.50 and in the graph in figure 5.34.

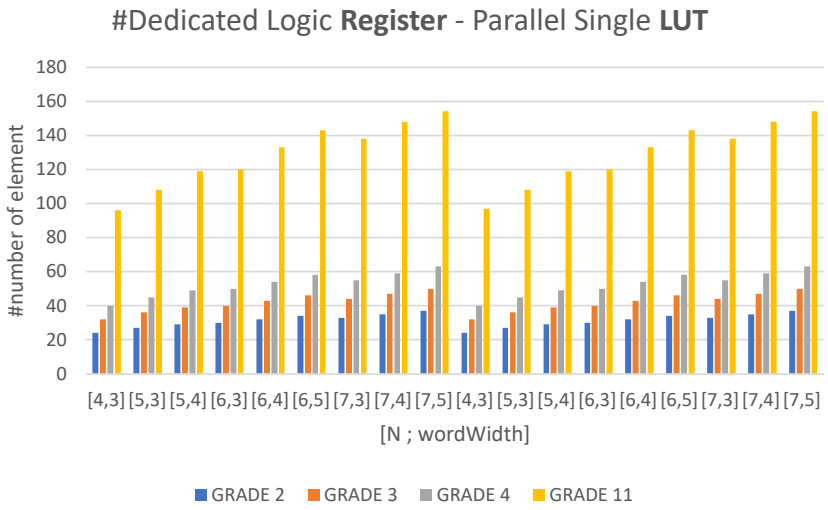


Figure 5.34: Relationship of Logic Reg - Synthesis (left) and Fitter (right) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 24 | 32 | 40 | 96 |
| **[5,3]** | 27 | 36 | 45 | 108 |
| **[5,4]** | 29 | 39 | 49 | 119 |
| **[6,3]** | 30 | 40 | 50 | 120 |
| **[6,4]** | 32 | 43 | 54 | 133 |
| **[6,5]** | 34 | 46 | 58 | 143 |
| **[7,3]** | 33 | 44 | 55 | 138 |
| **[7,4]** | 35 | 47 | 59 | 148 |
| **[7,5]** | 37 | 50 | 63 | 154 |

Table 5.49: Logic Reg for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 24 | 32 | 40 | 97 |
| **[5,3]** | 27 | 36 | 45 | 108 |
| **[5,4]** | 29 | 39 | 49 | 119 |
| **[6,3]** | 30 | 40 | 50 | 120 |
| **[6,4]** | 32 | 43 | 54 | 133 |
| **[6,5]** | 34 | 46 | 58 | 143 |
| **[7,3]** | 33 | 44 | 55 | 138 |
| **[7,4]** | 35 | 47 | 59 | 148 |
| **[7,5]** | 37 | 50 | 63 | 154 |

Table 5.50: Logic Reg for Fitter

Finally, we observe the $\mathbf{F_{max}}$ **summary** maximum frequency, as always studied in the worst case with the junction temperature at 0 °C.
Here we note an interesting detail: in the serial configuration there was a deterioration of the working frequency with the increase of the bit-widths reaching a stability around the 150 MHz. In this parallel configuration instead we note the same stability, but with worse frequencies especially for degree 3 and 4.
In general, for the application on FPGA, the parallelization of operations does not seem to offer greater advantages (as instead introduced on silicon) but on the contrary it seems to maintain the same frequencies of the serial architecture and, at the same time, deteriorating the cost in area, especially as the degree increases. The graph is reported in 5.35 while the table in 5.51.

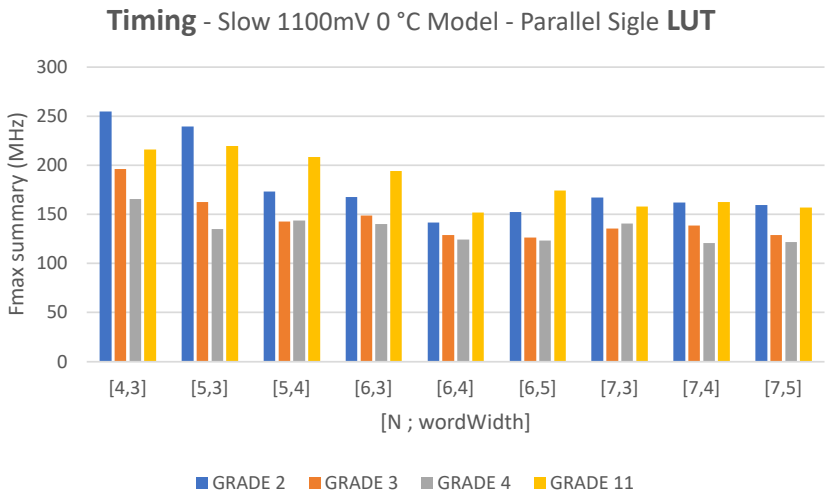


Figure 5.35: Timing Analyzer - $F_{max}$ Summary (MHz) - LUT block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 254,91 | 196,27 | 165,37 | 216,03 |
| **[5,3]** | 239,35 | 162,73 | 134,81 | 219,78 |
| **[5,4]** | 173,19 | 142,55 | 143,88 | 208,16 |
| **[6,3]** | 167,64 | 148,96 | 139,84 | 193,87 |
| **[6,4]** | 141,76 | 128,68 | 124,19 | 151,68 |
| **[6,5]** | 152,42 | 126,31 | 123,5 | 174 |
| **[7,3]** | 166,89 | 135,54 | 140,55 | 158,05 |
| **[7,4]** | 161,79 | 138,47 | 120,51 | 162,58 |
| **[7,5]** | 159,26 | 129,02 | 121,92 | 156,67 |

Table 5.51: $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model

### THERMO

In this section we will cover the results of logic utilization in **ALMs** regarding the THERMO block in the parallel configuration. The first observations show, as usual, a slight improvement by the Fitter, and an increasing number of instantiated resources with the increasing of bit-widths.
Graphs and tables concerning the number of instantiated ALMs can be displayed in 5.36 and 5.52 (for Synthesis) and 5.53 (for Fitter) respectively.

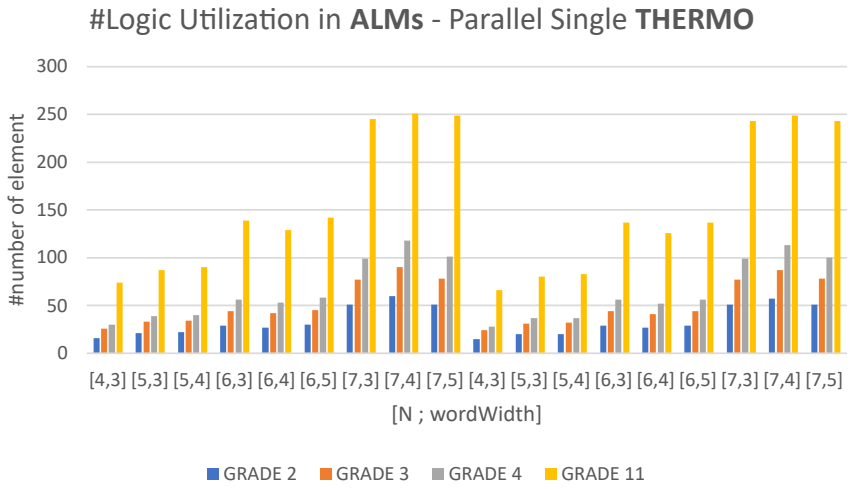


Figure 5.36: Relationship of ALMs - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 16 | 26 | 30 | 74 |
| **[5,3]** | 21 | 33 | 39 | 87 |
| **[5,4]** | 22 | 34 | 40 | 90 |
| **[6,3]** | 29 | 44 | 56 | 139 |
| **[6,4]** | 27 | 42 | 53 | 129 |
| **[6,5]** | 30 | 45 | 58 | 142 |
| **[7,3]** | 51 | 77 | 99 | 245 |
| **[7,4]** | 60 | 90 | 118 | 251 |
| **[7,5]** | 51 | 78 | 101 | 249 |

Table 5.52: ALMs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 15 | 24 | 28 | 66 |
| **[5,3]** | 20 | 31 | 37 | 80 |
| **[5,4]** | 20 | 32 | 37 | 83 |
| **[6,3]** | 29 | 44 | 56 | 137 |
| **[6,4]** | 27 | 41 | 52 | 126 |
| **[6,5]** | 29 | 44 | 56 | 137 |
| **[7,3]** | 51 | 77 | 99 | 243 |
| **[7,4]** | 57 | 87 | 113 | 249 |
| **[7,5]** | 51 | 78 | 100 | 243 |

Table 5.53: ALMs for Fitter

As the previous section, also in this case we do not notice a particular difference between the two types of synthesis in **ALUTs** but also for these elements we have a remarkable advantage over the LUT blocks.
The reference graph is visible in Figure 5.37. The tables in 5.54 and 5.55.

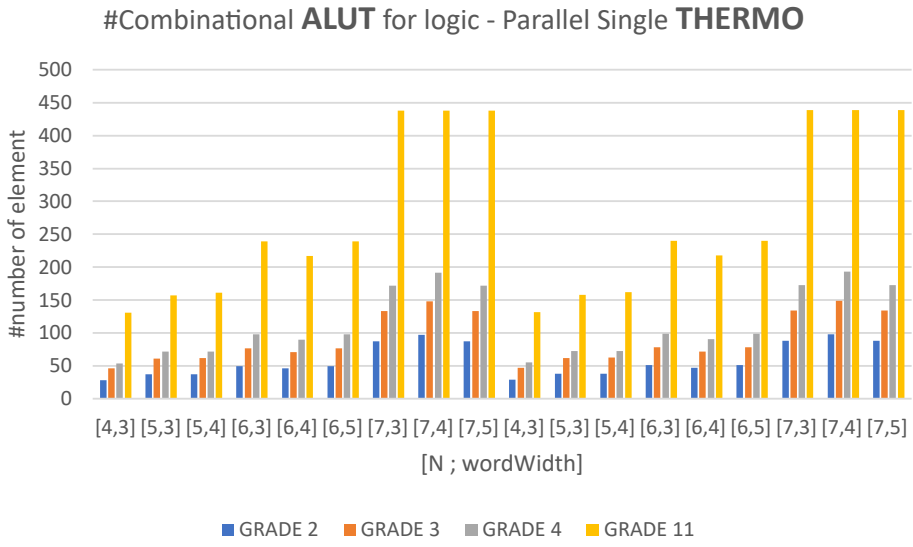


Figure 5.37: Relationship of ALUTs - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 28 | 46 | 54 | 131 |
| **[5,3]** | 37 | 61 | 72 | 157 |
| **[5,4]** | 37 | 62 | 72 | 161 |
| **[6,3]** | 50 | 77 | 98 | 239 |
| **[6,4]** | 46 | 71 | 90 | 217 |
| **[6,5]** | 50 | 77 | 98 | 239 |
| **[7,3]** | 87 | 133 | 172 | 438 |
| **[7,4]** | 97 | 148 | 192 | 438 |
| **[7,5]** | 87 | 133 | 172 | 438 |

Table 5.54: ALUTs for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 29 | 47 | 55 | 132 |
| **[5,3]** | 38 | 62 | 73 | 158 |
| **[5,4]** | 38 | 63 | 73 | 162 |
| **[6,3]** | 51 | 78 | 99 | 240 |
| **[6,4]** | 47 | 72 | 91 | 218 |
| **[6,5]** | 51 | 78 | 99 | 240 |
| **[7,3]** | 88 | 134 | 173 | 439 |
| **[7,4]** | 98 | 149 | 193 | 439 |
| **[7,5]** | 88 | 134 | 173 | 439 |

Table 5.55: ALUTs for Fitter

Again, there are no appreciable differences between the two synthesis strategies for **Dedicated Logic Register**. The values are shown in the table 5.56 and 5.57 to better observe the few oscillations that take place.

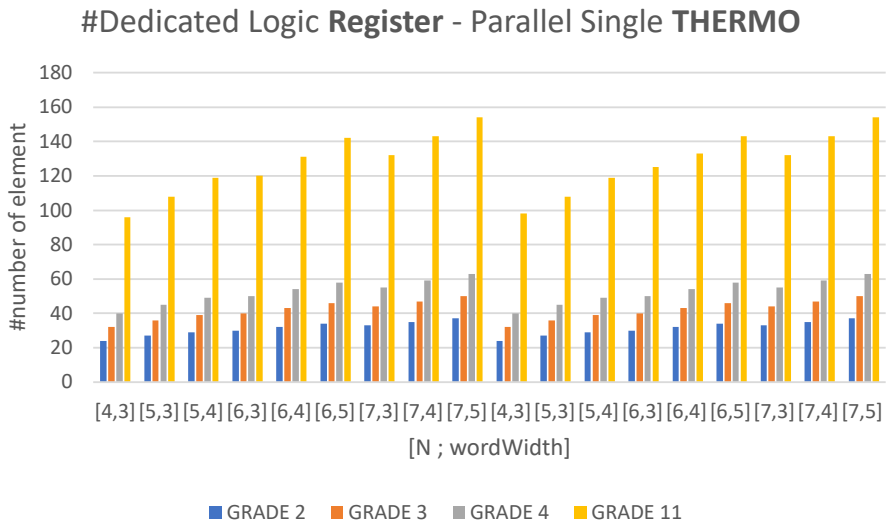


Figure 5.38: Relationship of Logic Reg - Synthesis (left) and Fitter (right) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 24 | 32 | 40 | 96 |
| **[5,3]** | 27 | 36 | 45 | 108 |
| **[5,4]** | 29 | 39 | 49 | 119 |
| **[6,3]** | 30 | 40 | 50 | 120 |
| **[6,4]** | 32 | 43 | 54 | 131 |
| **[6,5]** | 34 | 46 | 58 | 142 |
| **[7,3]** | 33 | 44 | 55 | 132 |
| **[7,4]** | 35 | 47 | 59 | 143 |
| **[7,5]** | 37 | 50 | 63 | 154 |

Table 5.56: Logic Reg for Synthesis

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 24 | 32 | 40 | 98 |
| **[5,3]** | 27 | 36 | 45 | 108 |
| **[5,4]** | 29 | 39 | 49 | 119 |
| **[6,3]** | 30 | 40 | 50 | 125 |
| **[6,4]** | 32 | 43 | 54 | 133 |
| **[6,5]** | 34 | 46 | 58 | 143 |
| **[7,3]** | 33 | 44 | 55 | 132 |
| **[7,4]** | 35 | 47 | 59 | 143 |
| **[7,5]** | 37 | 50 | 63 | 154 |

Table 5.57: Logic Reg for Fitter

As for the **$F_{max}$ summary**, we can conclude by making the same observations in the previous section for frequency, also given that similar data are reported, even if the quantization and reconstruction block has been modified. To verify this the graph and the tables are reported in 5.39 and 5.58 respectively.

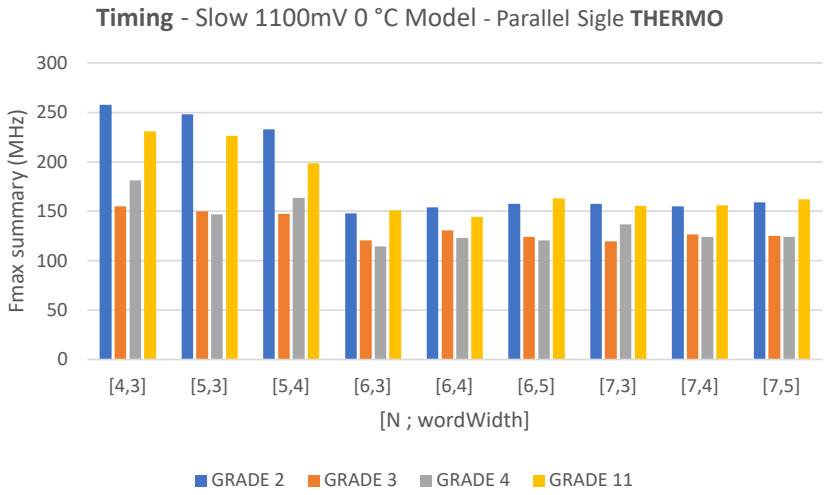


Figure 5.39: Timing Analyzer - $F_{max}$ Summary (MHz) - THERMO block

| [N, wordWidth] | GRADE | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **11** |
| **[4,3]** | 257,67 | 155,18 | 181,26 | 231,05 |
| **[5,3]** | 248,08 | 150,04 | 147,12 | 226,45 |
| **[5,4]** | 232,99 | 147,3 | 163,4 | 198,53 |
| **[6,3]** | 147,73 | 120,66 | 114,59 | 150,78 |
| **[6,4]** | 154,23 | 130,68 | 123,2 | 144,49 |
| **[6,5]** | 157,53 | 124,05 | 120,73 | 163,03 |
| **[7,3]** | 157,53 | 119,69 | 136,59 | 155,69 |
| **[7,4]** | 154,8 | 126,45 | 124,33 | 155,91 |
| **[7,5]** | 159,24 | 125,13 | 123,98 | 162,05 |

Table 5.58: $F_{max}$ Summary (MHz) - Slow 1100mV 0 °C Model

### 5.2.3 Trade-off between Serial Single - LUT and THERMO

In this section, we will focus on illustrating the differences in the number of elements used for Single Serial configuration, depending on whether the LUT or the THERMO block is used for the quantization and reconstruction blocks. The number of instantiated ALMs is shown in figure 5.40, represented in this way to facilitate a better understanding of the differences in results.

The tables containing the data presented in the graph have already been shown in table 5.32 for the LUT and in table 5.39 for the THERMO.

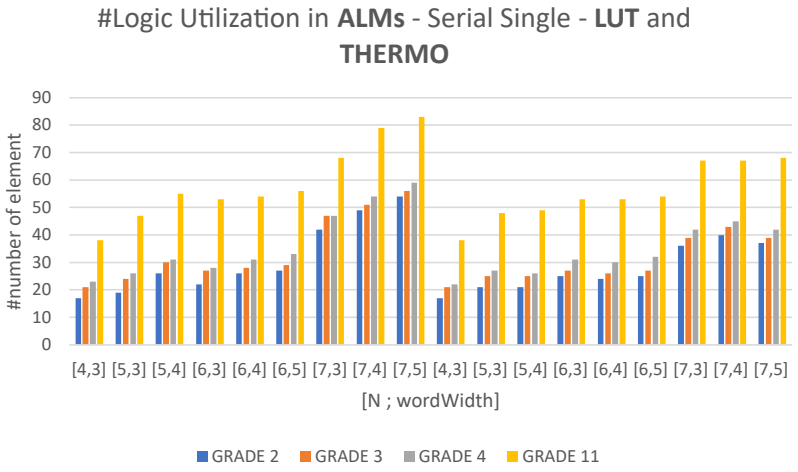


Figure 5.40: ALMs - Serial Single - LUT (left) and THERMO (right)

Similar values are observable in the overall configuration. However, it is highlighted how, especially with the increase of bit-widths of the metrics, there is a decrease in the ALMs required by the architecture adopting the THERMO block. This further confirms the observations made for ASIC synthesis, highlighting how for high parameters, THERMO proves to be more efficient in terms of resources used.

In figures 5.41 and 5.42, respectively, graphs for ALUT and Dedicated Logic Register with the same configuration are presented. We can observe very similar values for both cases, with THERMO employing some additional dedicated registers but optimizing the number of ALUTs used, especially for high parameters.

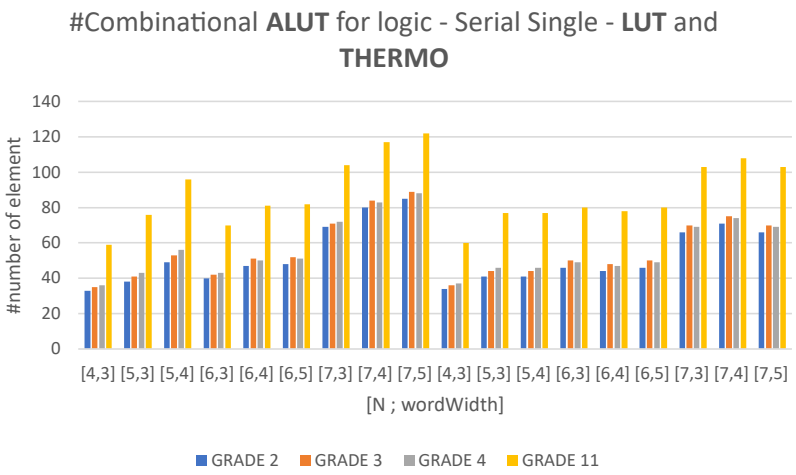


Figure 5.41: ALUTs - Serial Single - LUT (left) and THERMO (right)

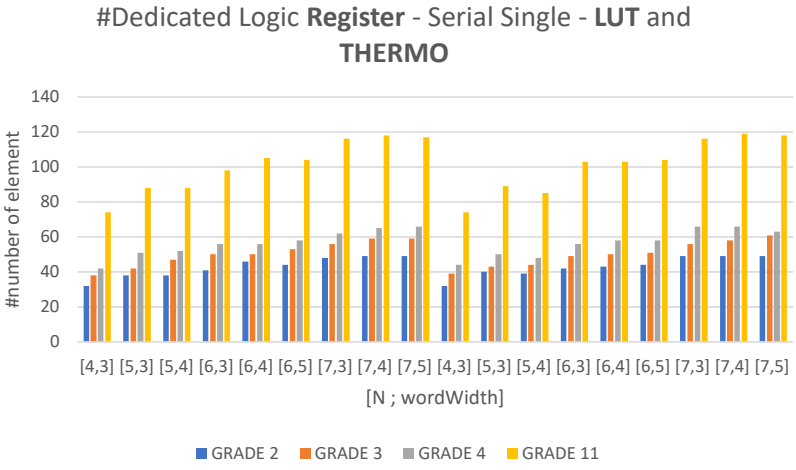


Figure 5.42: Logic Register - Serial Single - LUT (left) and THERMO (right)

The frequency graph highlights similarities in the trend of the two configurations. However, it can be observed that the THERMO architecture offers higher frequencies for reduced metric values. As these metrics increase, both configurations tend to converge towards the same performance with some little improvements from the LUT in some isolated cases. The observable working frequency is influenced by how the circuit is synthesized and can provide better optimizations than others depending on the parameters used. Therefore, this study proves that can be important do the evaluation of this results to know the best performances.
The graph of the frequencies is reported in 5.43:

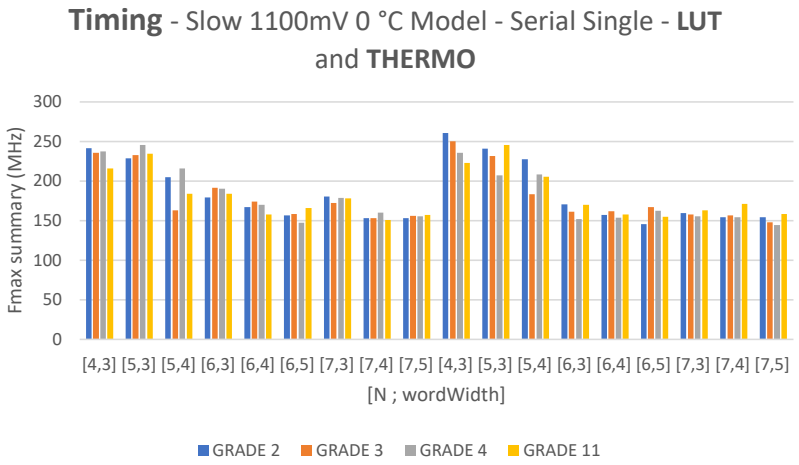


Figure 5.43: Timing Analyzer - Serial Single - LUT (left) and THERMO (right)

The tables containing the data for ALUT, Dedicated Registers and Frequencies have already been previously provided. Therefore, below are reported the references to these tables:

- Single Serial LUT - ALUT usage - 5.34
- Single Serial LUT - Logic Register usage - 5.36
- Single Serial LUT - $F_{MAX}$ Summary - 5.37
- Single Serial THERMO - ALUT usage - 5.41
- Single Serial THERMO - Logic Register usage - 5.43
- Single Serial THERMO - $F_{MAX}$ Summary - 5.44

### 5.2.4 Trade-off between Parallel Single - LUT and THERMO

In this section, we will focus on analyzing the same comparisons made previously, focusing on the differences between the two distinct R-Q blocks on the parallel architecture. The number of instantiated ALMs is shown in figure 5.44.

The tables containing the data presented in the graph have already been shown in table 5.46 for the LUT and in table 5.53 for the THERMO.

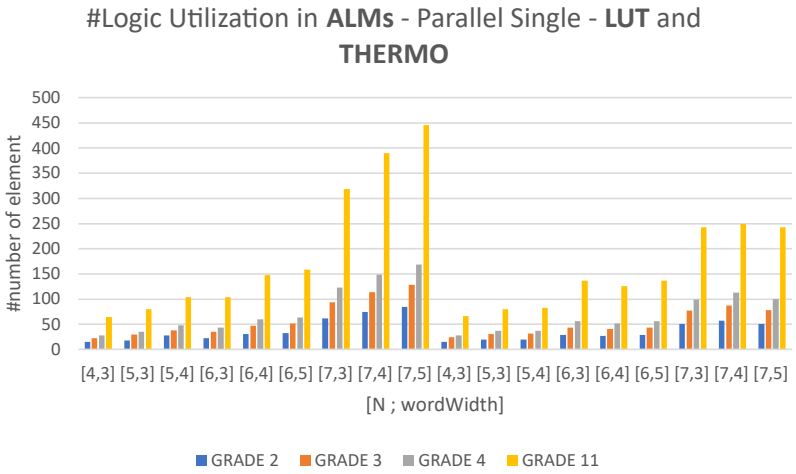


Figure 5.44: ALMs - Parallel Single - LUT (left) and THERMO (right)

Observing the large number of instantiated elements allows us to identify the use of the parallel architecture. THERMO blocks demonstrates a better space management, and then, the utilization of elements in terms of the number of ALM and ALUT. The results show a general similarity in effectiveness, but with improvements achieved by the THERMO blocks for high values of [N, wordWidth] for all degrees under consideration. However, similar results are observed in the number of dedicated registers used, which are almost identical for both configurations.
The previously data obtained for ASIC synthesis is also confirmed on FPGA, showing an advantage of using THERMO blocks within parallel architectures as the parameters increase. The contributions offered by ALUTs and Logic Registers are shown respectively in figure 5.45 and 5.46.

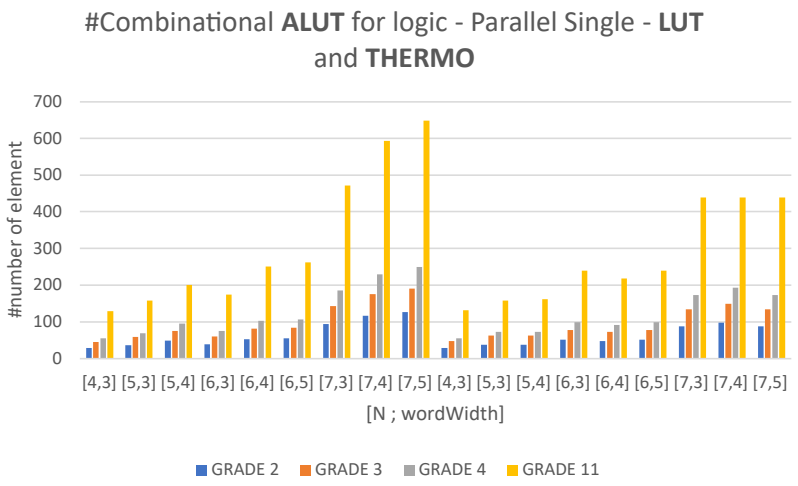


Figure 5.45: ALUTs - Parallel Single - LUT (left) and THERMO (right)

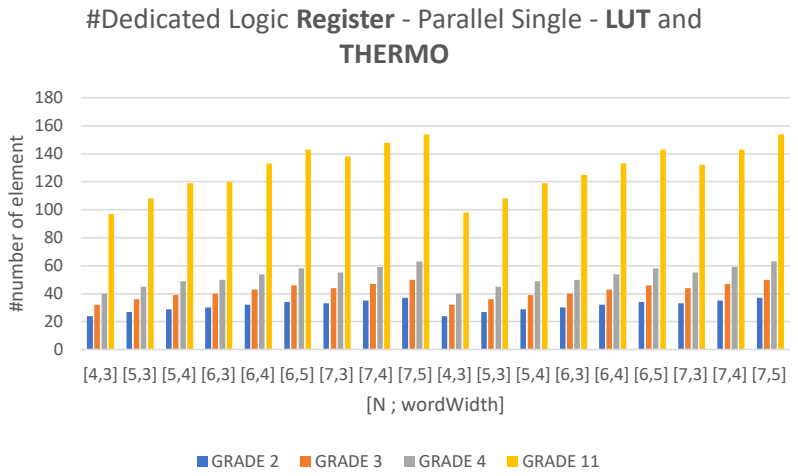


Figure 5.46: Logic Register - Parallel Single - LUT (left) and THERMO (right)

The frequency shows similar trends for both configurations, but higher frequencies are noted for degree 2 and degree 11 applications, indicating slowdowns for the other degrees, probably due to necessary synthesis required by the structures at the cost of operating frequency. The graph is reported in figure 5.47.

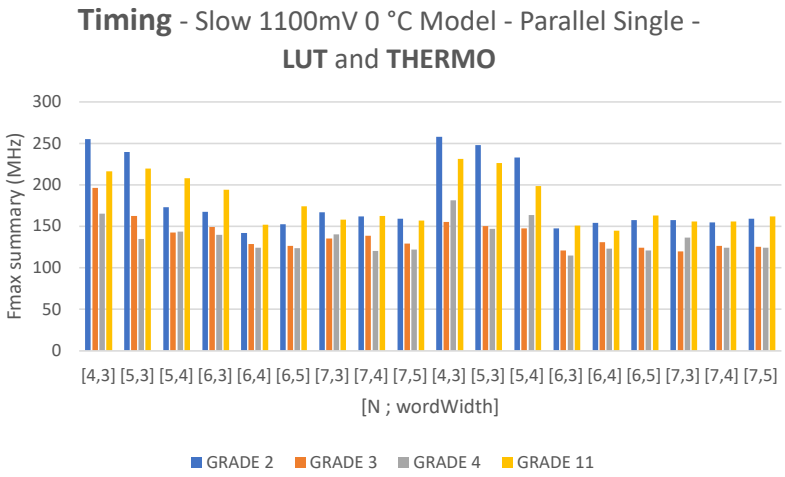


Figure 5.47: Timing Analyzer - Parallel Single - LUT (left) and THERMO (right)

The tables containing the data for ALUT, Dedicated Registers and Frequencies have already been previously provided. Therefore, below are reported the references to these tables:

- Single Parallel LUT - ALUT usage - 5.48
- Single Parallel LUT - Logic Register usage - 5.50
- Single Parallel LUT - $F_{MAX}$ Summary - 5.51
- Single Parallel THERMO - ALUT usage - 5.55
- Single Parallel THERMO - Logic Register usage - 5.57
- Single Parallel THERMO - $F_{MAX}$ Summary - 5.58

### 5.2.5 Trade-off between Single LUT - Serial and Parallel

In this section, we will present the results obtained from the synthesis of two different architectures, serial and parallel, while keeping the Look-Up tables as quantization and reconstruction blocks.

The observed results are predictable and consistent with the analyses conducted previously: the parallel architecture necessarily requires a greater number of ALM, ALUT, and Logic Register components, given the nature of the structure. Graphs are provided to simplify the analysis of the performances of these variations.

Graphs are provided for the utilization of ALMs in 5.48, ALUTs in 5.49, Logic Registers in 5.50.

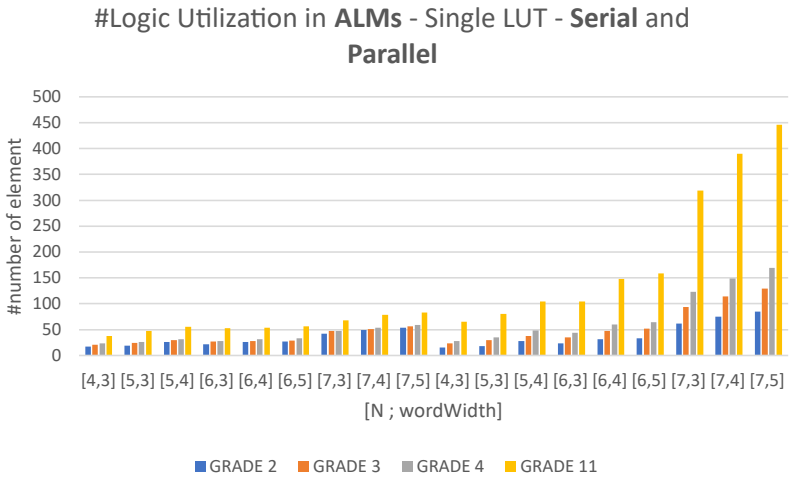


Figure 5.48: ALMs - Single LUT - Serial (left) and Parallel (right)

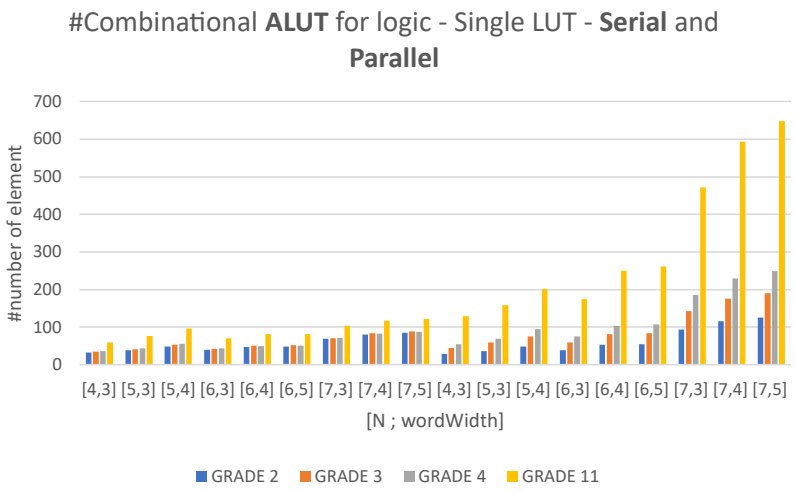


Figure 5.49: ALUTs - Single LUT - Serial (left) and Parallel (right)

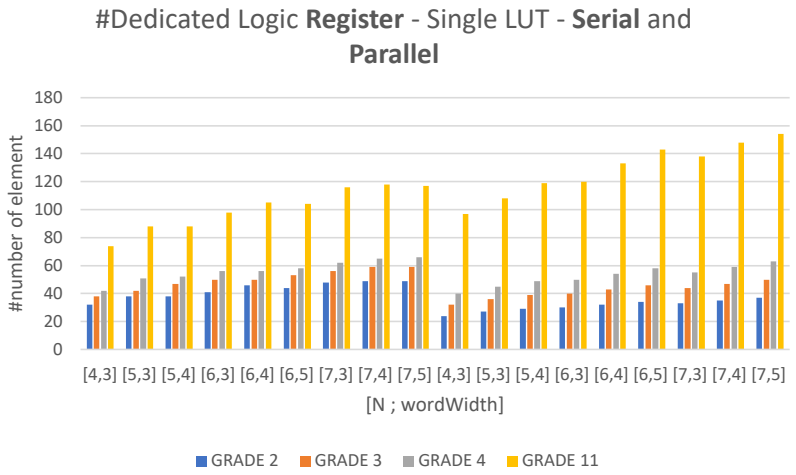


Figure 5.50: Logic Register - Single LUT - Serial (left) and Parallel (right)

Regarding performances in terms of frequency, the use of the parallel configuration decrease the values, except the applications for degree 2 and degree 11, where, in most cases, better results are observed compared to serial structures. This phenomenon highlights the difficulties encountered by the FPGA in effectively synthesizing these architectures, as already reported in previous sections. Graph for frequency is reported in 5.51.

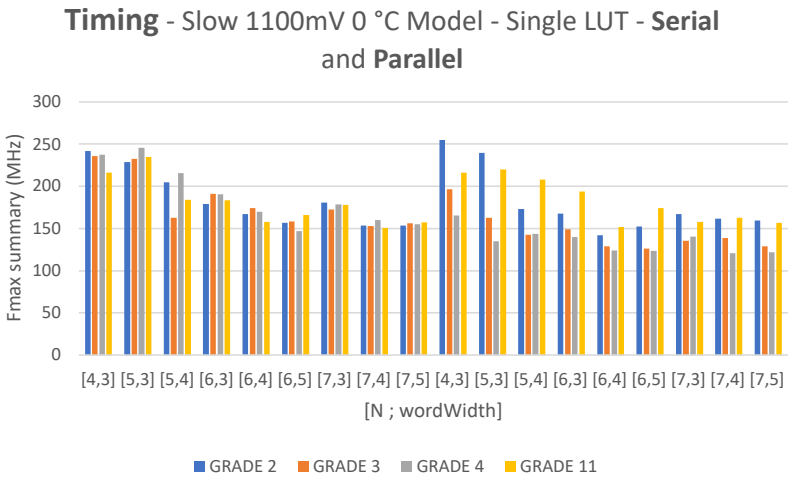


Figure 5.51: Timing Analyzer - Single LUT - Serial (left) and Parallel (right)

The tables containing the data for ALM, ALUT, Dedicated Registers and Frequencies have already been previously provided. Therefore, below are reported the references to these tables:

- Single LUT Serial - ALM usage - 5.32
- Single LUT Serial - ALUT usage - 5.34
- Single LUT Serial - Logic Register usage - 5.36
- Single LUT Serial - $F_{MAX}$ Summary - 5.37
- Single LUT Parallel - ALM usage - 5.46
- Single LUT Parallel - ALUT usage - 5.48
- Single LUT Parallel - Logic Register usage - 5.50
- Single LUT Parallel - $F_{MAX}$ Summary - 5.51

### 5.2.6 Trade-off between Single THERMO - Serial and Parallel

In this final section, the results generated by the synthesis for both Serial and Parallel architectures will be presented, while keeping THERMO as the block used for data quantization and reconstruction.

A predictable quantity of ALM and ALUT components is highlighted, although smaller than the use of LUT. Additionally, it is observed that the number of components remains almost constant with varying wordWidth parameter, maintaining the same value of N. However, significant differences are noted with variations of N or degree.

The number of Logic Registers used in the Parallel architecture is lower for all degrees, except for degree 11 which shows a more intensive use of such registers.
Graphs are provided for the utilization of ALMs in 5.52, ALUTs in 5.53, Logic Registers in 5.54.

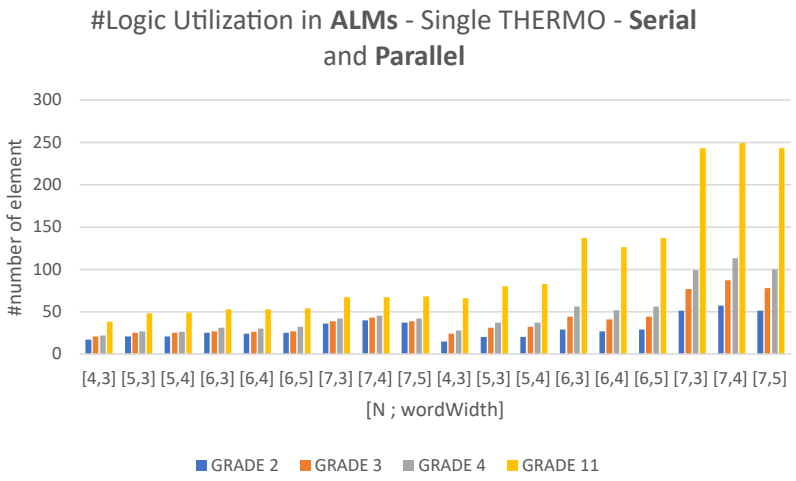


Figure 5.52: ALMs - Single THERMO - Serial (left) and Parallel (right)

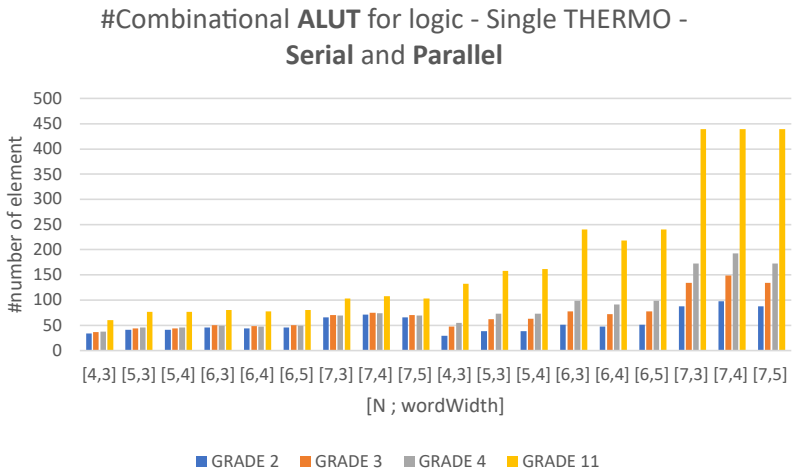


Figure 5.53: ALUTs - Single THERMO - Serial (left) and Parallel (right)

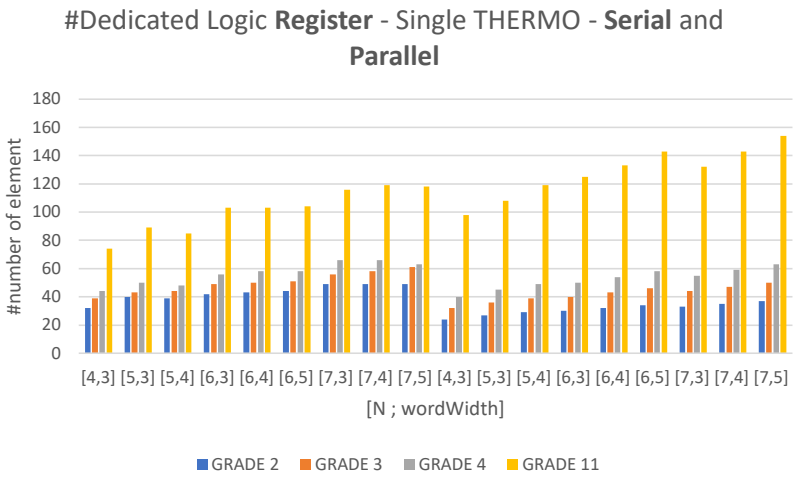


Figure 5.54: Logic Register - Single THERMO - Serial (left) and Parallel (right)

The working frequency perform a decrease with the increasing of parameters, for both Serial and Parallel architectures. This trend confirms the observations made in the previous section about the advantages offered by degrees 2 and 11, maintaining values similar to Serial implementations, except for applications with reduced parameters like [4,3], [5,3], and [5,4].
In this case, it is noted how the Parallel architecture influences the intensive use of resources. However, it is also observed that the use of THERMO blocks allows, in some cases, achieving better performance in terms of resource utilization or working frequency. Graph for frequency is reported in 5.55.

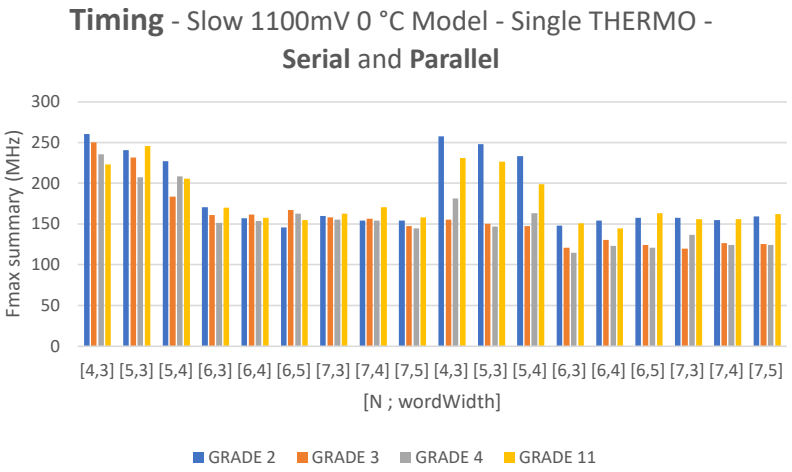


Figure 5.55: Timing Analyzer - Single THERMO - Serial (left) and Parallel (right)

The tables containing the data for ALM, ALUT, Dedicated Registers and Frequencies have already been previously provided. Therefore, below are reported the references to these tables:

- Single THERMO Serial - ALM usage - 5.39
- Single THERMO Serial - ALUT usage - 5.41
- Single THERMO Serial - Logic Register usage - 5.43
- Single THERMO Serial - $F_{MAX}$ Summary - 5.44
- Single THERMO Parallel - ALM usage - 5.53
- Single THERMO Parallel - ALUT usage - 5.55
- Single THERMO Parallel - Logic Register usage - 5.57
- Single THERMO Parallel - $F_{MAX}$ Summary - 5.58

# Chapter 6

# Conclusion

The presented thesis aimed to explore the complexity of the Variable Node (VN) in an LDPC decoder, with the objective of investigating the Reconstruction-Computation-Quantization (RCQ) paradigm for the decoding. This approach involves the quantization of the data inside the VN structure from an internal parallelism to an external one with a lower number of bits, in order to reduce the computational load on the Check Nodes (CN), another component of the decoder. Subsequently, the quantized data are reconstructed inside the VN to return to their original form, at the original bit-width.

The main focus of the study was directed towards analyzing the components involved in quantization and reconstruction within the VN, considering different configurations as the key metrics vary. The observed metrics include the number of bits internal to the VN (N), the number of external bits (wordWidth) for the quantized data, and the degree of the structure, which identifies the number of inputs to a single VN.

Various architectures of the VN were studied and described in VHDL language, both serial and parallel, considering both the single VN and its complete configuration with the distribution of degrees described by the H interconnection matrix, in order to find the optimal parameters to optimize the occupied area, frequency, and critical path of the structure.
Additionally, two different approaches for quantization and reconstruction were examined, one based on Look-up tables (LUT) and the other using a block called THERMO, which employs comparators for data quantization.

The circuits were simulated to verify the correctness of the data flow and then synthesized to collect all the necessary data to outline a complete overview of the performances.
Synthesis campaigns were conducted both on silicon for ASIC using 65nm technology through Synopsys software, and on FPGA using Intel Quartus Prime. The results were collected through scripts, which automated the synthesis processes of all configurations with all combinations of the observed metrics.
The data showed how the use of comparators in the THERMO block for quantization and reconstruction led to significant improvements in the area occupied by the blocks,

especially for high values of the parameters N and wordWidth. However, for VN operating on reduced metrics, the use of LUT is still convenient. Furthermore, the use of THERMO blocks reduces critical path periods, allowing for a higher working frequency compared to that provided by the LUT.

Several analyses were performed by mixing different architectures and block implementations to provide a comprehensive study of VN complexity, highlighting the strengths of the structure depending on the chosen parameters.

This study can provide useful information on the parameters to use for a design of a circuit that must meet specific requirements in terms of area or timing.
Furthermore, this work provide the basis for future work that may expand the study by exploring additional structures for quantization and reconstruction branches, different architectures for the VN, or the use of other parameter values, in order to identify the optimal structure to use in the design of a circuit with specific requirements.

# Bibliography


[1] Glossary intel. https://www.intel.com/content/www/us/en/programmable /quartushelp/current/index.htmreference/glossary/glosslist.htm. [Online; Accessed: March 28, 2024].

[2] Timing closure methodology for advanced fpga designs. https://www.intel.com/content/www/us/en/docs/programmable/683145/21-3/change-fitter-placement-seeds.html. [Online; Accessed: March 28, 2024].

[3] M.P. Ajanya and George Tom Varghese. Thermometer code to binary code converter for flash adc - a review. In *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)*, pages 502–505, 2018. doi: 10.1109/ICCPCCT.2018.8574244.

[4] Omur Aydogmus and Gullu Boztas. Implementation of an fpga-based motor control with low resource utilization. In *2019 4th International Conference on Power Electronics and their Applications (ICPEA)*, pages 1–5, 2019. doi: 10.1109/ICPEA1.2019.8911149.

[5] Robert G. Gallager. Low-density parity-check codes. *Monograph MIT Press*, 1963.

[6] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x.

[7] Mahmudul Hasan. LDPC Codes from Communication Standards. Mar 2024. URL `http://webdemo.inue.uni-stuttgart.de`. Webdemo.

[8] A. Hasani, L. Lopacinski, G. Panic, and E. Grass. 550 gbps fully parallel fully unrolled ldpc decoder in 28 nm cmos technology. In *2022 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, pages 429–433, 2022. doi: 10.1109/EuCNC/6GSummit54941.2022.9815814.

[9] Xuan He, Kui Cai, and Zhen Mei. On mutual information-maximizing quantized belief propagation decoding of ldpc codes. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019. doi: 10.1109/GLOBECOM38437.2019.9013335.

[10] Michael Meidlinger, Alexios Balatsoukas-Stimming, Andreas Burg, and Gerald Matz. Quantized message passing for ldpc codes. In *2015 49th Asilomar Conference on*

*Signals, Systems and Computers*, pages 1606–1610, 2015. doi: 10.1109/ACSSC.2015.7421419.

[11] Joo-Yul Park and Ki Chung. Parallel ldpc decoding using cuda and openmp. *EURASIP Journal on Wireless Communications and Networking*, 2011, 12 2011. doi: 10.1186/1687-1499-2011-172.

[12] Shiva Kumar Planjery, David Declercq, Ludovic Danjean, and Bane Vasic. Finite alphabet iterative decoders—part i: Decoding beyond belief propagation on the binary symmetric channel. *IEEE Transactions on Communications*, 61(10):4033–4045, 2013. doi: 10.1109/TCOMM.2013.090513.120443.

[13] Dewan Siam Shafiullah, Mohammad Rakibul Islam, Mohammad Mostafa Amir Faisal, and Imran Rahman. Optimized min-sum decoding algorithm for low density pc codes. In *2012 14th International Conference on Advanced Communication Technology (ICACT)*, pages 475–480, 2012.

[14] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. URL `http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`.

[15] J Shrinidhi, P. Sri Krishna, Yamuna B, and Pargunarajan K. Modified min sum decoding algorithm for low density parity check codes. *Procedia Computer Science*, 171:2128–2136, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.04.230. URL `https://www.sciencedirect.com/science/article/pii/S1877050920312151`. Third International Conference on Computing and Network Communications (CoCoNet'19).

[16] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981. doi: 10.1109/TIT.1981.1056404.

[17] Caleb Terrill, Linfang Wang, Sean Chen, Chester Hulse, Calvin Kuo, Richard Wesel, and Dariush Divsalar. Fpga implementations of layered minsum ldpc decoders using rcq message passing. 04 2021.

[18] Linfang Wang, Richard D. Wesel, Maximilian Stark, and Gerhard Bauch. A reconstruction-computation-quantization (rcq) approach to node operations in ldpc decoding. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020. doi: 10.1109/GLOBECOM42002.2020.9348139.

[19] Linfang Wang, Caleb Terrill, Maximilian Stark, Zongwang Li, Sean Chen, Chester Hulse, Calvin Kuo, Richard D. Wesel, Gerhard Bauch, and Rekha Pitchumani. Reconstruction-computation-quantization (rcq): A paradigm for low bit width ldpc decoding. *IEEE Transactions on Communications*, 70(4):2213–2226, 2022. doi: 10.1109/TCOMM.2022.3149913.