Hardware and algorithm study for a fingerprint verification system

Yago Mandado

Abstract

Biometrics is the area of knowledge comprising the methods for unique human recognition, based upon one or more intrinsic physical or behavioural traits. The term has its origin in the ancient Greek language (bios = "life" plus metron = "measure").

Among all the biometric techniques [1], some of them of recent development, fingerprinting has been the most widely used during the 20th Century. The maturity of this biometric technique and, mainly, the dramatic improvement of the capture devices have lead to the proposal of fingerprinting in multiple applications.

In this thesis a hardware architecture for a Fingerprint Verification System (FVS) in a mobile embedded device, one of the most challenging problems in this field, is developed. The objective of the system is to decide whether an input fingerprint belongs to the same individual than a previously stored one. This procedure is called verification, in contrast with identification, that tries to decide which fingerprint in a database shows higher resemblances with an input one.

In this thesis, new algorithms are proposed for fingerprint matching, taking as a departure point literature and research publications from last decades. The main development is the introduction of Delta and Core detection algorithms. With this points a classification process of the fingerprints is implemented.

As final step, a custom designed hardware architecture that implements the functions is designed.
## Index

1 Introduction.................................................................................................................6
   1.1 Background in FVS...............................................................................................6
   1.2 Motivation and goals.............................................................................................7
   1.3 Main results.............................................................................................................7
   1.4 Thesis overview......................................................................................................7
2 Related work..................................................................................................................9
   2.1 Basic concepts.........................................................................................................9
      2.1.1 Minutiae...........................................................................................................9
      2.1.2 Deltas and Cores...........................................................................................10
   2.2 Minutiae-based matching algorithms.....................................................................11
      2.2.1 Minutiae extractor..........................................................................................12
         2.2.1.1 Jain et al. minutiae extraction algorithm..............................................13
         2.2.1.1.1 Estimation of the Orientation Field..................................................13
         2.2.1.2 Jain et al. minutiae extraction algorithm..............................................14
      2.2.2 Minutiae matching.........................................................................................14
         2.2.2.1. Elastic comparison algorithm..............................................................14
   2.3 Core and Delta detection algorithms.....................................................................15
      2.3.1 Binarization......................................................................................................15
      2.3.2 Thinning.............................................................................................................16
      2.3.3 Calculation of the first direction matrix.........................................................16
      2.3.4 Calculation of the second direction matrix....................................................17
      2.3.5 Calculation of the singular points...................................................................17
         2.3.5.1 Calculation of Core points......................................................................17
         2.3.5.2 Calculation of Delta points......................................................................18
   2.4 Summary..................................................................................................................18
3 Proposed Hybrid algorithm..........................................................................................19
   3.1 Introduction.............................................................................................................19
   3.2 Core and Delta Extraction.....................................................................................20
      3.2.1 Estimation of the orientation field..................................................................20
      3.2.2 Delta and Core Extraction.............................................................................21
   3.3 Minutiae Extraction...............................................................................................21
      3.3.1 Binarization.....................................................................................................21
      3.3.2. Thinning.........................................................................................................21
      3.3.3 Minutiae extraction.......................................................................................22
   3.4 Minutiae matching..................................................................................................22
   3.5 Comparison of the hybrid matcher respect the state-of-the-art.............................23
   3.6 Summary..................................................................................................................24
4 Software Implementation...............................................................................................25
   4.1 Description of the implemented MATLABTM functions.......................................25
      4.1.1 Estimation of the orientation field..................................................................26
      4.1.2 Core and Delta and extraction.......................................................................27
      4.1.3 Binarization....................................................................................................28
      4.1.4 Thinning..........................................................................................................30
      4.1.5 Minutiae extraction.......................................................................................32
   4.2 Software implementation results.............................................................................32
      4.2.1 Implementation results of the classification methods....................................33
4.2.2 Implementation results of the minutiae extraction methods.............. 33
4.3 Conclusions from the test results.......................................................... 34
4.4 Summary.............................................................................................. 34
5 Hardware Architecture Implementation.................................................. 35
  5.1 Tile processing of an image.................................................................. 35
  5.2 Overview of the proposed architecture............................................... 36
  5.3 Scheduling of the hardware units....................................................... 37
    5.3.1 Minutiae Processing Unit Scheduling........................................... 37
      5.3.1.1 Binarization.......................................................................... 38
      5.3.1.2 Thinning.............................................................................. 40
    5.3.2 Delta-Core Processing Unit Scheduling........................................ 42
      5.3.2.1 Orientation Field Estimation................................................ 42
      5.3.2.1.1 Gradient Calculation Scheduling........................................ 43
      5.3.2.2 Delta and Core Extraction..................................................... 45
  5.4 Resource and timing budget of the developed methods....................... 45
  5.5 Image Memory – Processing Units interconnection............................. 45
    5.5.1 Description of the hardware memory block.................................. 45
    5.5.2 Scheduling of the memory data output........................................ 47
  5.6 Summary.............................................................................................. 47
6 Conclusions and future work.................................................................... 48
Appendix A Custom design multipliers for gradient multiplications............ 49
Bibliography............................................................................................... 54
**Figure index**

Figure 1.1 Architecture of an FVS.................................................................6
Figure 2.1. Minutiae singularities.................................................................9
Figure 2.2 Delta and Core singular points..................................................10
Figure 2.3. Fingerprint classification into six categories............................11
Figure 2.4. Flow-chart of a minutiae-based matcher.................................11
Figure 2.5. Flow-chart of a general minutiae extractor..............................12
Figure 2.6. Flow-Chart of Jain et al minutiae extraction algorithm..............13
Figure 2.7 Flow-chart of the Delta and Core detection algorithms..............15
Figure 2.8 Binarized image........................................................................16
Figure 2.9 Thinned fingerprint image........................................................16
Figure 3.1. Flow-chart of the proposed hybrid algorithm...........................19
Figure 3.2. Flow-chart of the the algorithm sharing Binarization and Thinning 20
Figure 4.1. Flow-chart of the proposed hybrid algorithm...........................25
Figure 4.2 Orientation Field Estimation Result..........................................27
Figure 4.3 Detected Core and Delta.............................................................28
Figure 4.4. Binarization Results.................................................................29
Figure 4.5 Correction of the thinning algorithm........................................31
Figure 4.6 Output of the thinning algorithm.............................................31
Figure 5.1. Redundancy needed for pixel interdependences.......................36
Figure 5.2 General scheme of the hardware architecture...........................37
Figure 5.3. C/C++ code of the binarization function..................................38
Figure 5.4. Scheduling of the sequential execution of the binarization function.39
Figure 5.5. Optimized SSG of the binarization function............................40
Figure 5.6. Optimized SSG of the calculation of condition 2 in Zheng's algorithm........41
Figure 5.7. Optimized SSG of the calculation of condition 3 in Zheng's algorithm........41
Figure 5.8. Optimized SSG of the calculation of condition 4 in Zheng's algorithm........42
Figure 5.9 List Scheduling Algorithm pseudo-code....................................43
Figure 5.10 ASAP SSG of the gradient function........................................44
Figure 5.11 List Scheduling SSG of the gradient function...........................44
Figure 5.12 ASAP SSG of the atan2 function.............................................44
Figure 5.13 Cycle budget of the implemented methods.............................45
Figure 5.14 Resource budget of the implemented methods........................45
Figure 5.15 Memory block and interconnections.......................................46
Figure 5.16 Scheduling of the memory control signals..............................47
Figure A.1 ASAP SSG of the Karatsuba multiplication function..................51
Figure A.2 List scheduling SSG with the Karatsuba multipliers...................51
Figure A.3 Example of the VLSI oriented multiplier behaviour...................53

**Table index**

Table 2.1 Quantization of the per tile fingerprint orientation field..................16
Table 3.1 Comparison of state-of-the-art and hybrid algorithm.....................23
Table 4.1 Comparison of state-of-the-art and hybrid algorithm.....................33
Table 4.2 Comparison of state-of-the-art and hybrid algorithm.....................34
Table 5.1. 8-neighbourhood of a pixel p..................................................40
1 Introduction

The study of fingerprints started in the 18th century. Although there are studies reflecting the possibility that the human being knew the uniqueness of fingerprints since the Egyptian Empire times, it was not until the end of 19th century when the use of fingerprints with the objective of personal identification started. Sir Edward Henry and Sir Francis Galton, separately, worked on the applicability of these techniques.

The main characteristics a biometric measure has to fulfil in order to be suitable for such applications are universality, uniqueness, permanence, collectability, acceptability, performance and circumvention[1]. The high agreement of fingerprinting with these characteristics has been the key of its success as an identification technique in the last century.

The most challenging problem nowadays in Fingerprint Verification Systems (FVS) is to achieve enough reliability and speed at low power, in order to implement them on mobile embedded devices.

This thesis focuses in the fingerprint matching step of FVS with the previously mentioned objective.

1.1 Background in FVS

As shown in Figure 1.1, a FVS consists basically of five steps:

- Fingerprint input.
- Enrollment.
- 2nd Fingerprint Input.
- Authentication.
- Matching Score.

![Figure 1.1 Architecture of an FVS](image)

As first step the user inputs its fingerprint in order to be enrolled in the system and the different features of the fingerprints are extracted and stored in a database by the enrolment stage. When an user wants to enter the system protected by the FVS, makes a new input of its fingerprint. The authentication module makes the same process than enrolment but subsequently uses the extracted features to compute a similarity score between both inputs.
The main specification of the system this thesis deals with, is that the database only contains fingerprint data of one user.

1.2 Motivation and goals

In last years, the maturity of fingerprint algorithms and the improvement of capture devices, have lead to the goal of a FVS implementation in mobile embedded devices. This objective implies several design constraints in the system algorithms. For example, it is essential to achieve good results without using floating-point processing units (FPU).

Therefore special algorithms have to be developed to achieve good results in this architecture. This thesis work has the following main goals:

- The development of such new algorithms.
- The development of a software implementation of this algorithms, to prove its correctness and suitability for fingerprint images. The trade-off between the calculation complexity of the algorithms and the quality of their results determines the final solution.
- The study of a hardware architecture design to implement the algorithms.

1.3 Main results

In this thesis, algorithms to solve the aforementioned problems have been developed. The main contribution is the application of methods to detect the Delta and Core singular points without introducing an additional complexity to the minutiae process (as will be explained throughout the thesis, Delta, Core and minutiae are the main singularities contained in a fingerprint image).

The application of the methodologies developed in this thesis, allows to detect with a high level of robustness the Delta and Core points and, at meantime, the extraction of minutiae at a very low level of operational complexity, as will be shown in Chapter 4.

Afterwards, a hardware study has been developed to test the feasibility of the implemented algorithms. As it is shown in Chapter 5, the results indicate that the methodologies are feasible and that extra redundancy could be added to raise the quality of the singularities extraction process.

1.4 Thesis overview

The subsequent chapters of this thesis are organized as follows:

On second chapter, previous research activities in fingerprint matching techniques are discussed, in order to analyze the different state-of-the-art solutions in FVS design.

On third chapter, after analyzing them, a matching technique that uses concepts from different algorithms is proposed.

On chapter four, the software implementation of these algorithms is presented, in order to validate the feasibility of the methods discussed in chapter two and make an
evaluation of the trade-off between their calculation complexity and the quality of their results.

Chapter five includes an architecture study and a proposal for the implementation of the algorithms on a custom designed hardware architecture.

Finally, chapter six, contains a summary of the thesis work, as well as the conclusions and suggestions for a latter work.
2 Related work

In this chapter, the basic concepts and an overview of the previous works in fingerprint matching are presented.

There are three main categories of fingerprint matchers:

- Minutiae-based matchers.
- Correlation-based matchers.
- Ridge feature based.

The main algorithms will be presented throughout the following paragraphs. Minutiae-based matchers are the most widely used because through the development of new algorithms[3][4], great results have been achieved for large database systems in non-real-time systems. The challenging problem is to develop suitable algorithms for real-time embedded systems. This avoids the use of non-minutiae algorithms, as these algorithms only allow the treatment of the whole image and involve more complex operations than the minutiae-based.

2.1 Basic concepts

The uniqueness of fingerprints in the human being has been the key of the success of such techniques in biometric identification. This uniqueness relies in the fact that the singularities (or singular points) of any human fingerprint have a different location in all of them.

A fingerprint can be defined as graphical flow-like ridges present in human fingers [3]. The main singularities that can be found in a human fingerprint can be calculated from the shape of the ridges, as shown in the following paragraphs.

2.1.1 Minutiae

In the ridges of a fingerprint image several singular points, named minutiae, can be found. Personal identification can be performed by means of the detected minutiae, their localization and orientation. Minutiae can be classified in a high number of types (usually 8) although all can be reduced to ridge endings and ridge bifurcations. Ridge endings occur when a ridge finishes abruptly and ridge bifurcations when two ridges intersect. Figure 2.1 [4], shows both types of minutiae.

![Minutiae singularities](image)

Figure 2.1. Minutiae singularities: a) Ridge ending b) Ridge bifurcation
Once the minutiae are extracted, relevant information about them is stored in minutiae vectors. This vectors are used to compute a similarity score between two fingerprints.

In paragraph 2.2 different algorithms for minutiae extraction are presented. In chapter 3 how these algorithms are used in the developed matcher is explained.

### 2.1.2 Deltas and Cores

The different orientation of the ridges throughout a fingerprint allows the recognition of other kind of singular points, named Cores and Deltas. The Core is defined as the point where the orientation of the ridges tends to diverge while the Deltas are the points where the orientation tends to converge. Figure 2.2 shows both types of singular points.

![Figure 2.2 Delta and Core singular points](image)

The most widespread application of Deltas and Cores is fingerprint classification, a widely used technique in large database fingerprint systems. Through the determination of the singular points [5][6][7], human fingerprints can be classified in six types, with a different appearance rate, which have the following Delta and Core singular points:

- Arch: It has no singular points (2% appearance rate).
- Tented Arch: It has a Delta point in central region (3%).
- Left Loop: It has a Core in the upper-left and a Delta in the lower-right region (30%).
- Right Loop: It has a Core in the upper-right and Delta in the lower-left region (30%).
- Whorl: It has a Core in upper-right region (30%).
- Twin or Double Loop: It has a Double Core in the central region (5%).

This classification is very important for large database systems because the identification of the type of fingerprint is performed as a first step and subsequently the input fingerprint is only matched against fingerprints belonging to the same type. This leads to a dramatic reduction of the search space. Figure 2.3 shows an image of the different types of fingerprints.

The asymmetry of the apparition pattern facilitates the use of the Delta and Core data obtained in the verification process.
In this thesis Delta and Core singular points are used with the following objectives:

- As reference points in minutiae matching: if the Delta and Core points are extracted reliably, they are used as reference point in the minutiae matching stage.

- As classification points: Through extraction of Delta and Core points, classification of input and template fingerprint is performed. If they belong to different categories, they are scored as non-match.

\[\text{a) Arch} \quad \text{b) Tented Arch} \quad \text{c) Right Loop} \]

\[\text{d) Left Loop} \quad \text{e) Whorl} \quad \text{f) Twin} \]

Figure 2.3. Fingerprint classification into six categories (a) Arch. (b) Tented arch. (c) Right loop. (d) Left loop. (e) Whorl. (f) Twin

2.2 Minutiae-based matching algorithms

In the last decades successful minutiae-based algorithms have been developed. They are the most widely used algorithms for fingerprint matching. These algorithms can be divided in different stages. Figure 2.4 shows a flow-chart of a general minutiae-based matcher.

Minutiae Extractor → Minutiae Matching → Matching Score

Figure 2.4. Flow-chart of a minutiae-based matcher
In the following paragraphs, the characteristics of the different stages of the general matcher, and different algorithms that can be used, are explained.

2.2.1 Minutiae extractor

The minutiae extractor stage can be divided, also, in different stages. Figure 2.5 shows the flow-chart of a general minutiae extractor.

![Flow-chart of a general minutiae extractor](image)

The basic concepts of each stage are:
- Preprocessing: Different operations are performed in the fingerprint image in order to prepare the image for a reliable extraction of minutiae. Examples of this techniques are noise removal and contrast enhancement algorithms.
- Minutiae extraction: In this stage a first set of minutiae candidate points is calculated from the preprocessed image.
- Post-processing: Inevitably, previous stages lead to a minutiae set that contains points that are not real minutiae (caused by bad image quality, scratches, imperfections of the algorithms...), called spurious minutiae. This points should be removed from the set, as they will be different for every acquirement of the fingerprint image and will degrade the performance of the matcher.

In the different minutiae extraction algorithms, the stages in Figure 2.5 perform different operations.

In the remaining part of the chapter, the most widely used algorithms both for minutiae extraction and minutiae matching, are explained. Special emphasis is made regardless to complexity, as this is the main constraint when targeting a mobile embedded device.

In the last years, two different algorithms have been mainly used in minutiae extraction:
- The first was devised by Anil K. Jain et al. in the University of Michigan (USA) and is explained in detail in paragraph 2.2.1.1[2].
- The second one was devised by Maio et al. in the University of Bologna (Italy) [3].
2.2.1.1 Jain et al. minutiae extraction algorithm

In 1997, Anil K. Jain et al. devised an algorithm for minutiae extraction[2], which is an improved version of the method proposed by Ratha et al.[8]. The flowchart of the algorithm is shown in Figure 2.6.

![Flow-Chart of Jain et al. minutiae extraction algorithm](image)

The Orientation Field Estimation stage, present in this algorithm, is used in the hybrid matcher developed in this thesis. This estimation is used as first step in the Delta and Core algorithm implementation. This first stage is explained in the subsequent paragraph.

2.2.1.1.1 Estimation of the Orientation Field

In this algorithm a new hierarchical implementation of Rao's algorithm[9] is used. It consists of the following basic steps:

1) Divide the input fingerprint image in sections of size $W \times W$.
2) Compute the gradients $G_x$ and $G_y$ at each pixel in each block.
3) Estimate the local orientation of each block through the following formula:

$$\theta_o = \frac{1}{2} \tan^{-1} \left( \frac{\sum_{i=1}^{W} \sum_{j=1}^{W} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{W} \sum_{j=1}^{W} (G_x^2(i,j) - G_y^2(i,j))} \right)$$

After this stage, this algorithm consists of a ridge extraction and a thinning stage. Their details are not explained here as their detailed implementation is irrelevant. Several techniques have been implemented for ridge extraction and thinning and are presented in chapters three and four.
2.2.1.2. Maio et al. minutiae extraction algorithm

Also, in 1997, Maio et al. devised an algorithm for minutiae detection directly from grey-scale images. Although rather different from each others, the other methods have in common a pre-processing step of image binarization, that is avoided in this solution. The results achieved, in spite of a greater conceptual and operational complexity, show a higher robustness and efficiency. This algorithm has been widely accepted by the research community. Among all the minutiae extraction techniques this seems to be the most efficient and should be used on pure minutiae matchers, specially if the algorithm is oriented to solve fingerprint identification applications.

In the proposed hybrid algorithm of chapter three, this direct extraction is not used, as its high operational complexity does not allow an efficient solution without the use of a large number of complex operations. This would lead to complex hardware circuits and the use of floating point units (FPU) to achieve reasonable results, drifting away from the objective of the study.

2.2.2 Minutiae matching

Once minutiae are extracted, relevant information about them is stored in vectors. For each minutiae, the following relevant information is stored:

- X-coordinate.
- Y-coordinate.
- Orientation: defined as the local ridge orientation of the associated ridge.
- Type of minutiae: Ridge ending or ridge bifurcation.

The information of all the extracted minutiae will be recorded in a pattern of vectors. These vectors will be used by the matching algorithms to compute a similarity score. The comparison of the input vector pattern with that of the template, is one of the most critic tasks of all the matching process. The problems are related to:

- In contrast with other biometric techniques, such as in iris recognition, there is no universal reference point; i.e. a point useful for measures and present in all fingerprints.
- The finger elasticity causes, depending on the finger posture and pressure in the acquirement, a great variability between different images of the same fingerprint. This problem can become crucial if the rotation angle is large, as many minutiae can appear in the input fingerprint image that are not in the template image, or vice-versa.

2.2.2.1. Elastic comparison algorithm

Once the minutiae have been extracted and relevant information stored in minutiae vectors, the final step is to devise a method that decides whether two different vectors belong
to the same fingerprint or not. The problem is reduced to a point pattern matching problem. Several algorithms have been proposed in the literature for point pattern matching [10][11] but they have shown either to be slow or unreliable.

Among all the proposed algorithms, the most widely used has been the Elastic Comparison Algorithm developed by A. K. Jain [2]. Its steps are the following:

- The first step is the search of a reference minutiae. Analysing the template and input minutiae vectors, a minutiae whose resemblances in both vectors is high enough to be considered the same minutiae.

- Then, both vectors are translated to a polar coordinate system, using the previously calculated reference minutiae as origin.

- Subsequently, all minutiae pairs that have possibilities to be the same, are compared one-to-one. To cope with the tolerance given by the elasticity of the finger, an influence area is created around each minutiae.

- As final step, successful and unsuccessful matches are given a positive and a negative score respectively. This leads to a numeric result that states the probability of both fingerprints to belong to the same user.

As in every biometric system, a threshold of the matching score to decide whether both fingerprints belong to the same finger or not, has to be established by the designer. This allows the study of the behaviour of the system under different thresholds, based on False Acceptance Rate (FAR, a user is allowed by the system when should not) and False Rejection Rate (FRR, a user is denied entrance when should be accepted).

2.3 Core and Delta detection algorithms

In the literature, different approaches to Delta and Core detection exist[12]. In this paragraph an algorithm devised in the University of Tamkang (Taiwan) in 2000 is presented [13]. Its main interest relays in the fact that all the preprocessing steps needed in this algorithm are similar to those needed in the minutiae extraction algorithm presented in section 2.3.1.a. The flow-chart of this algorithm is presented in Figure 2.7

![Flow-chart of the Delta and Core detection algorithms](image)

**Figure 2.7 Flow-chart of the Delta and Core detection algorithms**

2.3.1 Binarization

The first step of the algorithm is a binarization process. Binarization is the process of converting a grey-level image into a binary image. A huge number of binarization methods exists due to its applicability in different fields of knowledge[14]. Figure 2.8 shows a fingerprint image after the binarization process.
2.3.2 Thinning

Subsequently a thinning process is applied to obtain the skeleton of the fingerprint image. After this process the image is prepared for the next steps for singular point extraction. Figure 2.9 shows a fingerprint image after the thinning process.

2.3.3 Calculation of the first direction matrix

In this stage the thinned image is divided in sub-areas of 16*16 pixels each one. The slope of each sub-area is calculated and the the direction of all sub-areas are quantized in four directions using the following equations:

<table>
<thead>
<tr>
<th>Area Type</th>
<th>Representation</th>
<th>Angle Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Horizontal Line</td>
<td>-22.5&lt;arctan(slope)&lt;= 22.5</td>
</tr>
<tr>
<td>2</td>
<td>Vertical Line</td>
<td>abs(arctan(slope) &gt; 67.5)</td>
</tr>
<tr>
<td>3</td>
<td>45º clockwise bended line(from vertical)</td>
<td>22.5&lt; arctan(slope) &lt;= 67.5</td>
</tr>
<tr>
<td>4</td>
<td>135º clockwise bended line(from vertical)</td>
<td>-67.5&lt;arctan(slope)&lt;= -22.5</td>
</tr>
</tbody>
</table>

Table 2.1 Quantization of the per tile fingerprint orientation field
2.3.4 Calculation of the second direction matrix

The calculation of a second direction matrix improves the performance of the algorithm for bad quality fingerprint images; damage and scratches can cause inaccuracies in the first direction matrix. These problems can be solved through a quantization of the first direction matrix. The procedure is the following:

- 2x2 pixel windows are defined as the unit to quantize the first direction matrix.
- The main direction of the unit is calculated by calculating the dominant direction in the window. If the dominant direction is not unique, the window is expanded until a result is reached.

2.3.5 Calculation of the singular points

Once we have calculated the second direction matrix, the image is prepared for the calculation of the singular point candidates. For each type of points we calculate a first set of candidate points and afterwards remove false singular points. The procedure is explained in the following sub-paragraphs.

2.3.5.1 Calculation of Core points

If a point \( (i,j) \) corresponds to a Core point it satisfies

\[
|S(i,j-1) - S(i, j+1)| < 45 \\
|S(i-1,j) - S(i+1, j)| < 45
\]

To eliminate false points we examine the 2*2 diagonal neighbourhood, resulting the following matrix:

\[
\begin{bmatrix}
H1 & X & H2 \\
X & C(i,j) & X \\
H3 & X & H4
\end{bmatrix}
\]

In this matrix, the terms H1,H2,H3,H4 are the following matrices:

\[
H1 = [C(i-2,j-2), C(i-2,j-1); C(i-1,j-2), C(i-1,j-1)] \\
H2 = [C(i-2,j+1), C(i-2,j+2); C(i-1,j+1), C(i-1,j+2)] \\
H3 = [C(i+1,j-2), C(i+1,j-1); C(i+2,j-2), C(i+2,j-1)] \\
H4 = [C(i+1,j+1), C(i+1,j+2); C(i+2,j+1), C(i+2,j+2)]
\]
Each Hn matrix has four possible directions. After its calculation, finally, the point C(i,j) is a Core point if it satisfies one of the following equations:

Lower-left Core: C=45, CH1=45, CH2=135, CH3=45, CH4= 45 or 90
Lower-right Core: C=135, CH1=45, CH2=135, CH3=90 or 135, CH4= 45
Upper-left Core: C=45, CH1=45 or 90, CH2=45, CH3=135, CH4= 45
Upper-right Core: C=135, CH1=135, CH2=90 or 135, CH3=135, CH4= 45

2.3.5.2 Calculation of Delta points

If a point (i,j) corresponds to a Delta point, it satisfies

\[ S(i,j-1) = 45, \ S(i,j+1) = 135 \]

Using the same matrix than for Core calculation, changing the notation from C(i,j) to DP(i,j) a true Delta point has to satisfy the following equations:

\[ DH1 = 45 \text{ or } 90, \ DH3 = 45 \text{ or } 90, \ DH2 = 90 \text{ or } 135, \ DH4 = 0 \text{ or } 135 \]

2.4 Summary

In this chapter, the most relevant background art has been explained to help understand the different trends in fingerprint matching:

- Minutiae Matching: Elastic Comparison algorithm.

The application of Delta and Core in fingerprint matching is not widespread but, as will be shown in later sections, it can be carried using the explained extraction algorithm with the addition of a small calculation complexity. On the other hand, this points could be of great help in the matching stage.

None of the background minutiae algorithms is directly suitable for this thesis goal, and a combination of this algorithms, some of them in a simpler (i.e. also less precise) way, have to be devised.

In the next chapter, an hybrid algorithm, developed to fulfil the constraints of a mobile embedded device application is explained. It combines different background functions, allowing the extraction of the minutiae points \((x,y,\theta,\text{type})\) and Delta and Core points \((x,y,\text{type})\) without using complex operations. These methodologies are chosen because the extraction of Delta and Core points allows to perform a fingerprint classification process; this classification allows to exclude fast fingerprints belonging to different types and, at meantime, the Delta and Core points can be used as reference in a minutiae matching process.
3 Proposed Hybrid algorithm

An application specific algorithm has been developed that tries to accomplish this thesis goals. As long as it has been possible the most simple algorithms have been included. Its software implementation, described in Chapter 4, was the final decision maker, as in the complexity vs accuracy trade-off, accuracy can not fall below an acceptable level. Several other methodologies were studied, but the final designed hardware architecture impose another restriction in the algorithms used: they have to work in a per-tile(a small squared portion of the image) basis.

3.1 Introduction

In this section an hybrid algorithm based on different steps of the previously studied algorithms is presented. Figure 3.1 shows the flow-chart of the proposed algorithm. Several possibilities have been studied for the implementation of each of the stages, and their software implementations are presented in chapter four.

![Flow-chart of the proposed hybrid algorithm](image)

The proposed hybrid algorithm has two main different paths, that converge on its final Minutiae Matching stage. The first one, whose objective is to extract the Delta and Core singular points, is explained in section 3.2 and its based in two different algorithms presented in chapter three: Estimation of the orientation field (section 2.2.1.1) and Core and Delta Extraction (section 2.3). It also extracts the third term, ridge angle, in the minutiae triplet \((x,y, \theta)\).

All this algorithms are implemented, instead of in a full-image basis, on each of the tiles the image is divided in. The image is divided in squared pixel tiles. Some of the
algorithms, operate in this tiles (e.g. the Delta and Core extraction), while others could operate in more than one tile at the same time (e.g. thinning algorithm). This led to the final solution of a basic image information unit of four tiles in the hardware architecture. The details of this implementation are shown in chapter five.

3.2 Core and Delta Extraction

In this section an algorithm devised for Delta and Core extraction is presented. It is based in the algorithm presented in section 2.3, but the first binarization and thinning stages of this algorithm are replaced by another algorithm. This allows the more efficient implementation of the flow-chart of Figure 3.1, instead of that of Figure 3.2. The tile size of the original algorithm was 16x16 pixels, while in the developed solution it has been reduced to 5x5 pixels. This size gives a better result in the images of the database that has been used in the development of this project, as it is shown in Chapter 5.

![Flow-chart of the algorithm if Binarization and Thinning stages were shared by the Core-Delta and Minutiae extraction stages](image)

3.2.1 Estimation of the orientation field

The first step of the algorithm is the computation of an estimation of the orientation field. This calculation can be done through the hierarchical implementation of Rao's algorithm presented in section 2.2.1.1.1. Another possibility for the estimation of the orientation field, developed, among others in previous works in the CAS department in Delft University of Technology [15], is to implement an algorithm developed by Hong et al. called Least Mean Square Estimation (LMSE).

In this hybrid algorithm, an implementation of the algorithm explained in 2.2.1.1.1 is used.
3.2.2 Delta and Core Extraction

In this paragraph, the algorithm presented in section 2.4 is implemented. In the implemented solution, the first two stages of binarization and thinning have been replaced by an estimation of the orientation field. In chapter four, it is shown that this change does not degrade the performance of the Delta and Core algorithm. It allows a low complexity calculation of the Delta and Core singular points. This singular points, along with the minutiae is used by the matching score calculation stage to perform a matching score calculation.

3.3 Minutiae Extraction

3.3.1 Binarization

The first step is to binarize the image in order to separate the ridges from the background.

There are a great amount of different binarization methods in the literature[14]. Two different methods that, beforehand, obey the characteristics of low computational complexity required by the objective of low power embedded fingerprint application, have been implemented:

- A simple local thresholding method (based in the mean pixel value).
- Niblack's method (based in the mean and standard deviation values of pixel values).

Those pixels above that threshold are given a value '1' (ridge pixels) while the remaining are given a value '0' (background pixels).

Although more complex binarization methods have been studied, the implemented methods give good results (as it is proved in the MATLAB implementations shown in Chapter 4) at a low operational complexity and at meantime, allow the threshold to be calculated without interdependences between tiles. This allows a simpler hardware architecture, as is shown in chapter five. It is also remarkable, that more complex algorithms, if they are not directed to fingerprint applications, do not guarantee better results, as some of them are not suitable to the specific ridge-valley pattern of fingerprints.

3.3.2. Thinning

After the binarization of the ridge map, a relatively smooth ridge map of the fingerprint is obtained. The next step is to thin the ridge map, i.e. reduce the width of the ridges to one pixel. This allows a trivial extraction of minutiae and, at meantime, the application of a Minutiae Extraction algorithm. The following algorithms have been implemented and tested:

- Hilditch algorithm: This simple and parallel (the value of a pixel in the thinning process in the nth iteration depends on the value of its neighbours in the (n-1)th iteration, and consists of simple operations[16].
– Zhang and Suen algorithm: An implementation of a more complex and accurate parallel thinning method is developed. The higher complexity of this algorithm proves to give advantages in our fingerprint directed implementation of the method[17].

These two algorithms, along with a vast majority of the thinning algorithms are not directly suited for fingerprint applications. In a fingerprint application a thinned image should be considered one such each point has no more than two neighbours, while this is not the case in most of the general purpose thinning algorithms.

The results of the implementation of these methods, including a correction to make them suitable for fingerprint applications, are shown in Chapter 4.

3.3.3 Minutiae extraction

The thinned image, result of the previous stage leads to a straightforward extraction of a first set of minutiae points (this point set is sieved in subsequent stages).

A normal ridge pixel of the thinned map has two neighbour '1' pixels. Every other pixel is a singular point. This allows to extract the singular points using the following simple equations:

– If a ridge point has only one '1' neighbour, this point is a ridge termination minutiae.
– If a ridge point has more than two '1' neighbours, this point is a ridge bifurcation minutiae.

3.4 Minutiae matching

The minutiae matching stage could be based in the Elastic Comparison algorithm presented in section 2.2.2.1. Although its implementation has not been finally tested, it shown here for completeness. The information calculated in the previous stages allows changes in this algorithm, as well as other different strategies for similarity score calculation that can be computationally simple. The following strategies for minutiae matching can be used:

– A simple classification procedure can be performed. If both fingerprints belong to different fingerprint types the matching stage outputs a 'non-match' result.
– One of the most computationally expensive operations of the Elastic Comparison Algorithm is the calculation of a reference minutiae. Delta and/or Core points can be used with this purpose when the input fingerprint has Delta and/or Core points.
– If the fingerprint belongs to a type with more than one Delta/Core point several operations can be performed:
  – Overlapping the Delta and Core points of the stores and input fingerprint, performs a good alignment of the minutiae vectors calculated in previous steps.
  – If the distance between both points differs in both images, this can be used to resize one of the images. This operation can be helpful when this difference is caused by different finger pressure in the image acquisition, although it is useless to deal with non-uniform pressures.
3.5 Comparison of the hybrid matcher respect the state-of-the-art

In the state-of-the-art in fingerprint matching there are three main approaches: minutiae-based, correlation-based and ridge-feature matchers; each of them with different characteristics and applications. It is important to recall the two different main problems in fingerprint matching:

– Verification: Comparison 1:1. The fingerprint system user provides another mean of identification different than the biometric one, i.e. identification card containing his/her fingerprint.

– Identification: Comparison 1:N. The supposed identity of the user is not known by the system beforehand. The systems outcome is if the user is one of the N registered users in a database.

The goal of the hybrid algorithm is to solve the identification problem keeping operational complexity as low as possible. In Table 3.1 a comparison between the algorithms is presented.

<table>
<thead>
<tr>
<th></th>
<th>Minutiae-based matchers</th>
<th>Correlation-based matchers</th>
<th>Ridge matchers</th>
<th>Hybrid algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational complexity</td>
<td>High</td>
<td>Medium</td>
<td>Very High</td>
<td>Medium - High</td>
</tr>
<tr>
<td>Accuracy</td>
<td>High – Very High</td>
<td>Medium</td>
<td>High – Very High</td>
<td>Medium - High</td>
</tr>
<tr>
<td>Verification SW feasibility</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Verification HW feasibility</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Identification SW feasibility</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Identification HW feasibility</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.1 Comparison of state-of-the-art and hybrid algorithm
3.6 Summary

After the presentation of the state-of-the-art, an algorithm for fingerprint matching, suitable for this thesis purposes has been presented. Due to the restrictions imposed by the hardware architecture presented in Chapter 5, all the algorithms work in a per-tile basis. In the next chapter, the software implementation of all this algorithms is disclosed. This software implementation results, for each of the presented stages, lead to the election of the different methods finally implemented.
4 Software Implementation

In this chapter, the software implementation, using MATLAB™, of the hybrid method discussed in previous chapter, is presented. In each of the sections, one of the hybrid method algorithms results is presented. Most of the implementations, include different methodologies; observing the results, the algorithms implemented in the final solution are chosen. Each sub-paragraph includes an analysis of the algorithms input and output, underlying method and its main characteristics. In the decisions there is always an underlying complexity-accuracy trade-off. The implemented hybrid method's flow-chart was presented in Figure 3.1 and it shown here again in Figure 4.1 for completeness.

![Figure 4.1. Flow-chart of the proposed hybrid algorithm](image)

4.1 Description of the implemented MATLAB™ functions

In this section, the implementation of the functions included in the flow-chart of Figure 4.1 is described in detail. Each sub-paragraph is devoted to one of the stages of the hybrid algorithm.

All the function results are presented over the same fingerprint. This fingerprint has been chosen because it presents all the fingerprint features important for the analysis of the different methods:

- Low contrast ridges: It can be observed, in the upper-left part of the image, that the ridges have a poor and discontinuous contrast. This has been caused by non-uniform pressure in the image acquisition. This could lead to holes in the processed image.
High contrast ridges: In the middle-right part of the image there are some ridges with a very small inter-ridge distance. This could lead to ridge fusion after processing the image.

As the image belongs to the left loop type, it has a Core point and a Delta point.

Obviously, as any fingerprint, it has many bifurcation and termination minutiae.

Besides the aforementioned parts of the image, the remaining parts of the fingerprint image are of medium-high quality.

The characteristics of this image, allow to test all problems that could arise during the development of a fingerprint matching method.

4.1.1 Estimation of the orientation field

In this paragraph, the implementation of the Rao's algorithm, using a new hierarchical approach is described. As this is the first step of the hybrid algorithm, its input is the original fingerprint image, as it is stored in the database. This estimation of the orientation field will be used in subsequent stages of the hybrid algorithm: singular points detection and ridge slope information. From this slope information, the third term of the minutiae triplet (x,y,θ) is extracted.

The main characteristics of this software function are the following:

- Input: Original fingerprint image.

- Output: The fingerprint image is divided in squared pixel tiles. In each of the tiles, the orientation of the fingerprint ridges is calculated using a 45° resolution, i.e. there are only four possible orientations: vertical, horizontal and 45° blended (clockwise and counter clockwise). The input and output of this algorithm are shown in Figure 4.2.

- Other important features: The orientation field estimation algorithm described in section 2.2.1.a is implemented in MATLAB. Tests have to be implemented to decide which is the tile size that gives a better result. A trade-off between complexity (smaller tiles lead to a bigger amount of calculations, and as is explained in the hardware chapter to higher data) and accuracy of the subsequent stages (smaller tiles lead to a more accurate calculation of orientation and therefore to a more accurate detection of Core and Delta points) should be bear in mind. It is important to stand out that the operation of this algorithms allows to easily calculate, at meantime, the region of interest (from here on ROI) of the fingerprint image; i.e. the region of the image that belongs to the fingerprint, separating it from the image background.
4.1.2 Core and Delta and extraction

In this paragraph, the implementation of a Core and Delta singular point detection is described. The method in which this function is based is described in section 2.4.

The main characteristics of this software function are the following:

- **Input**: In the algorithm of section 2.4 the first two steps are binarization and subsequent thinning process of the fingerprint image. This resulting image is used to calculate the orientation field and from this orientation field a 1\textsuperscript{st} direction matrix is calculated. In the hybrid method implemented in this thesis the input of the Core and Delta extraction is not the database fingerprint after a binarization and thinning process; the output image of the Orientation Field Estimation function is used instead.

- **Output**: The output of this function is the detected Core and Delta singular point vectors. The information stored in vectors for each of the singularities is:
  - Coordinates
  - Singularity type: Delta or Core point

- **Other important features**: The input and output of this function are shown in Figure 4.3, with intermediate calculation steps images: the calculation of a 2\textsuperscript{nd}
direction matrix, calculated averaging the outcome of the ridge estimation function. As it was mentioned before, this singularities will be used in later stages to improve the calculations of the minutiae matching stage keeping its complexity as low as possible.

Figure 4.3 Detected Core (bottom-left) and Detected Delta (bottom-right)

4.1.3 Binarization

In this paragraph, the implementation of the ridge detection and binarization functions is described. The methods in which this functions are based are referred in section 3.3.1

The main characteristics of this software functions are the following:

- Input: The input of this functions is the original database fingerprint.
- Output: The output of this function is a binarized image of the fingerprint, i.e. an image where all the pixels have either value '1' or value '0'.
– Other important features: It is important to recall the great importance of this process in the overall performance of the system; as this is a stage where imperfections in the algorithm can lead to an important number of spurious detected minutiae that would affect the performance of the matching stage.

As mentioned in section 3.3.1, two different non-computationally complex methods have been implemented. Another important characteristic of these methods in that they can be applied in a per-tile basis (i.e. do not depend on global image features) allowing the hardware design of chapter five. In Figure 4.4 it is shown the result of the application of both, local thresholding and Niblack's methods and the result of the application of MATLAB in-built thinning function (more powerful; i.e. more computationally complex than the implemented one). This allows an insight in the suitability of these binarization algorithms for a fingerprint verification application.

![Figure 4.4. Local threshold method (upper) and Niblack's method (lower)](image)

In Figure 4.4, the best output reached, after several different thresholds were tested for each method in different images, for a medium quality fingerprint are shown. the following characteristics can be observed:

– The local thresholding method presents holes in the regions with poor ridge-valley contrast (upper-left region of the image) after thinning has been applied. Niblack's method presents this holes, but with a smaller size.
– Niblack's binarization method presents isolated points (known as speckles in image processing terminology) mostly in the outer fingerprint region (right next to the ROI's end).

– Niblack's method presents, in the parts where ridges are closer, a fusion between ridges, showing them as connected, while they are separated in the original image.

All this features of both algorithms have to be dealt with in a post-processing step, where the spurious minutiae are deleted from the set. The complexity of this task resides in the difficulty of removing the spurious minutiae without removing real ones.

Niblack's method presents an advantage due to the smaller size of the holes in the thinned image, but this holes are rarely present in real minutiae, allowing an easier removal in subsequent stages. On the other hand, the isolated points present in the outcome can be easily removed without chance of real minutiae being erased.

Finally, the inter-ridge fusion shown in the outcome of the thinned image of Niblack's method represents a more complex problem to deal with, as it is impossible to distinguish between those artificially created by the binarization process and those belonging to the input fingerprint. The collision between two ridges is a quite common feature of fingerprints (recall, it is called bifurcation minutiae). The presence of this artificial minutiae (with its logical inherent random appearance) would degrade the performance of subsequent stages.

In the last paragraph of this chapter this methods are used combined with the thinning method in a minutiae extraction reliability study.

### 4.1.4 Thinning

In this section, several implementations of the thinning function, described in section 3.3.2 are presented. The thinning algorithm involves the higher computational complexity of all the design.

The main characteristics of this software function are the following:

– **Input:** The input of this function is the outcome of the previously explained binarization function.

– **Output:** The objective is to thin the ridge map, i.e. to turn the original ridges of the fingerprint image into 'one pixel wide' ridges that will allow a trivial extraction of the minutiae in later stages. This thinned image will be the output of the thinning function. The input and output images of this function are shown in Figure 4.5. for two different methods: Hilditch's method and Zhang's method. Subsequently, their outcome is studied in detail to reach a decision about which method to implement.

– **Other important features:** There is not an universal definition of thickness in image processing. This causes that most state-of-the-art thinning algorithms present small features that do not make them suitable for direct use in fingerprint applications. The problem of most algorithms consist in a pattern allowed in the thinned lines, that conflicts with the definition of minutiae used by subsequent stages. in Figure 4.5 the scope of this problem is shown.
This problems have to be dealt with in a post-processing step over the thinned image in order to guarantee the correctness of the result of the subsequent stages of minutiae extraction. As can be observed in Figure 4.5.a) some of the points in the thinned line have more than two neighbours, leading to an interpretation of them as bifurcation minutiae while they are no more than normal ridge points.
It is important to recall that the images shown, represent the outcome of the thinning algorithm after the application of a post-processing step to remove the small aberration previously mentioned in this section.

From the outcome of the implementation of both methods, the following facts can be observed:

- Both methods perform a correct thinning of the binarized images: there are not ridge breaks present in the binarized input image.
- Hilditch's method outcome shows, in areas with disconnected ridges, an excessive reduction of this lines.

Due to the absence of a common interpretation of thinning process, some of the state-of-the-art thinning algorithms reduce and center the patterns in the thinning process, eroding information that should not be removed in a fingerprint application.

The outcome of Zheng's algorithm seems to be better than the produced by the Hilditch's algorithm, while its computational complexity does not exceed reasonable terms (all operations are either bit-logical or additions). In the last paragraph of this chapter this fact is demonstrated when the results in fingerprint minutiae extraction reliability prove to be better using this method.

4.1.5 Minutiae extraction

In this paragraph, the implementation of a Minutiae extraction process is described. It is a simple process. As previously mentioned, all ridge non-minutiae points have exactly two neighbours in their 8-point neighbourhood. Those points with more than two neighbours are considered bifurcation minutiae, while those with only one neighbour are considered termination minutiae.

The main characteristics of this software function are the following:

- Input: The input of this function is the output of the thinning function.
- Output: The output of this function is a vector containing the set of detected minutiae points. For each point, the following information is recorded:
  - Coordinates.
  - Type of minutiae: Bifurcation or termination.

4.2 Software implementation results

In this paragraph the results extracted from the test applied to the software implementations are shown.

Two different tests have been performed:

- A classification test: In this test the Orientation Field Estimation and Delta and Core extraction functions are applied to paired fingerprints (two different images belonging to the same finger). The classification is considered successful if all the singular points present in both fingerprint images are extracted correctly. Different tile sizes have been used to determine which gives better results and will be implemented in the final solution.
A minutiae extraction test: In this test the minutiae of different fingerprint images have been extracted and compared with the minutiae actually present in the images. This test has been applied to the four different combinations given by the two different binarization and thinning methods explained in the previous paragraphs of this chapter.

### 4.2.1 Implementation results of the classification methods

In this test, the pairs of fingerprints have been divided in six different types, as mentioned in paragraph 2.1.2. The percentage of successful extraction for the different tile sizes is shown. The percentage results are presented for each fingerprint type, although the different types where not equally represented in the tests developed. The results are shown in Table 4.1.

In this process, the Delta and Core points of two different images of the same fingerprint are extracted. If the Delta and Core are extracted correctly in both fingerprints, the extraction is considered successful. An extraction will be considered correct when the existing all the points are detected and the points do not exceed an euclidean distance of two pixels from the location designated by a human expert.

As can be observed in the table below, the results are better using a 5x5 tile size, for all the different types of fingerprints. When the tile size grows or diminishes (not shown in the table) the accuracy of the extraction process is lower.

<table>
<thead>
<tr>
<th>Type</th>
<th>5x5 Tile Size</th>
<th>6x6 Tile Size</th>
<th>7x7 Tile Size</th>
<th>8x8 Tile Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>100,00%</td>
<td>100,00%</td>
<td>90,00%</td>
<td>90,00%</td>
</tr>
<tr>
<td>Tented Arch</td>
<td>93,33%</td>
<td>90,00%</td>
<td>83,33%</td>
<td>83,33%</td>
</tr>
<tr>
<td>Left Loop</td>
<td>90,00%</td>
<td>90,00%</td>
<td>83,33%</td>
<td>77,67%</td>
</tr>
<tr>
<td>Right Loop</td>
<td>86,67%</td>
<td>86,67%</td>
<td>80,00%</td>
<td>83,33%</td>
</tr>
<tr>
<td>Double Loop</td>
<td>83,33%</td>
<td>77,67%</td>
<td>73,33%</td>
<td>73,33%</td>
</tr>
<tr>
<td>Whorl</td>
<td>86,67%</td>
<td>83,33%</td>
<td>83,33%</td>
<td>80,00%</td>
</tr>
</tbody>
</table>

Table 4.1 Results of the classification test with different tile sizes

### 4.2.2 Implementation results of the minutiae extraction methods

In this second test, the different fingerprints have been classified in three different types: high-quality, medium-quality and low-quality fingerprints. The binarization, thinning and minutiae extraction methods have been applied and the results compared with the minutiae actually present in the fingerprint image. Table 4.2 shows the results obtained for the
different types of fingerprints and with the different combinations of methods used. The percentage of minutiae successfully extracted respect to the total amount of minutiae (in blue) and the percentage of spurious minutiae with respect to the total amount of minutiae (in red) are shown for each combination of methods.

As can be observed in Table 4.2, the results are better for the combination of the Mean Threshold method as Binarization algorithm and Zheng as Thinning algorithm, for the three different types of fingerprint; both in the successful extraction of the present minutiae and in the introduction of less spurious minutiae. The results also show that the results should be improved to allow the successful matching of low quality fingerprints.

<table>
<thead>
<tr>
<th></th>
<th>Mean - Zheng</th>
<th>Mean - Hilditch</th>
<th>Niblack - Zheng</th>
<th>Niblack - Hilditch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High Quality</strong></td>
<td>90.12%</td>
<td>86.78%</td>
<td>80.12%</td>
<td>77.55%</td>
</tr>
<tr>
<td></td>
<td>6.23%</td>
<td>7.87%</td>
<td>14.29%</td>
<td>14.10%</td>
</tr>
<tr>
<td><strong>Medium Quality</strong></td>
<td>83.56%</td>
<td>81.56%</td>
<td>75.29%</td>
<td>73.48%</td>
</tr>
<tr>
<td></td>
<td>11.94%</td>
<td>14.41%</td>
<td>17.21%</td>
<td>17.82%</td>
</tr>
<tr>
<td><strong>Low Quality</strong></td>
<td>71.46%</td>
<td>70.02%</td>
<td>68.34%</td>
<td>65.25%</td>
</tr>
<tr>
<td></td>
<td>18.49%</td>
<td>20.62%</td>
<td>21.87%</td>
<td>22.32%</td>
</tr>
</tbody>
</table>

Table 4.2 Results of the minutiae extraction test

**4.3 Conclusions from the test results**

From the results presented in the previous paragraphs, two conclusion can be extracted:

- The best results in the classification methods have been reached using a tile size of 5x5 pixel tiles. As can be observed in Table 4.1 the performance of the methods degrades when using a higher tile size.
- The combination of the Mean threshold binarization method and the Zheng thinning algorithm give the best results in the minutiae extraction process. It can be observed in Table 4.2, that both the successful extraction rate (the percentage of minutiae extracted successfully from those designated by a human observer) and the error rate (percentage of spurious minutiae with respect to the total number of minutiae).

**4.4 Summary**

In this chapter the software implementation of the developed hybrid method has been presented. Different methods have been tested for each of the stages. The best outcomes reached, after testing and adjustments have been shown, concluding in the election of different methods for the binarization and thinning stages and the parameters to adjust the Delta and Core extraction method. All the outcomes are shown over the same fingerprint. This fingerprint has been chosen because it presents all the fingerprint features important for the analysis of the different methods.
In the remaining of this report, an study pointing towards a hardware architecture to implement the selected functions is presented. A resource and cycle budget is developed to justify the suitability of the developed methods for a hardware design.
5 Hardware Architecture Implementation

In this chapter, the development of a custom designed hardware architecture to implement the software functions explained in chapter 4 is presented. The basic idea is to develop a low power architecture that develops the aforementioned functions using the minimum possible resources: hardware, power and time. Of course, an inherent trade-off between these variables is always underlying.

First, the reasoning behind the election of per-tile methods is explained.

The next part of this chapter explains the global aspects of such architecture. In subsequent paragraphs, each of the elements of the architecture is explained in detail, departing from its first general implementation and reaching its final improved version. The description of the functions in C/C++ has been made departing from the MATLAB™ implementations explained in previous chapters.

An study about the scheduling of the software function has been made, in order to optimize the trade-off between hardware resources and execution cycles.

Further optimizations are also presented. In the case of complex operations, i.e. matrix multiplication, when it is not possible to reduce its number, complex parallel algorithms are presented to reduce dramatically the number of simple operations involved.

5.1 Tile processing of an image

The first decision to be taken is to decide the unit of information the architecture is going to deal with. The image size is 365x375 pixels, as the image is grey-scale (255 possible pixel values), the total amount of information of the used fingerprint images is 136 875 bytes, approximately 0.13 MB.

The amount of information is small, and the image could be processed as a whole. This is the common trend in software implementations of fingerprint algorithms.

To accomplish this in a specific purpose hardware would mean to expend a higher amount of hardware resources compared to the other possible solution: to divide the image in tiles and design hardware units that process tile per tile, until the whole image has been processed. Once this units have been designed any number of them can be used to process tiles in parallel.

This design has a drawback. In image processing techniques, some of the algorithms calculations involve, for one pixel, those in the 8-point neighbourhood; e.g. the thinning algorithm of the developed hybrid method under study. In such case, there are two possible solutions:

- Store the information of the pixels that is going to be needed to process another tiles in inner memory registers inside the processing units.
- Add redundancy to the information stored in the memory such that the tiles become again independent.
To store the information would be a costly process. Before a decision is made, the amount of redundancy needed has to be measured. This is made using as example the thinning algorithm.

To implement the thinning algorithm over a 5x5 pixel tile, the neighbours of all the border pixels is going to be needed, converting the size of the tile in 7x7. But this is not enough; not only the pixel values of this outer neighbour layer are needed, this pixels after being processed by the function is what is really needed. To process this pixels, all their neighbours are needed again, leading to a final tile size of 9x9 pixels.

Summarizing, to process an NxN pixel tile, in the case of interdependences at pixel level, (N+4)*(N+4) pixel tiles need to be used. It can be observed that the smaller the tiles, the bigger the amount of redundancy:

- Using 3x3 pixels tiles, 7x7 pixel tiles are stored (444% extra redundancy)
- Using 5x5 pixels tiles, 9x9 pixel tiles are stored (240% extra redundancy)

As the memory requirements of the application are small, this becomes an easier and more practical solution than the storage of the needed information between the processing of different tiles.

In Figure 5.1, this process is explained for a 5x5 pixels tile. Those pixels on blue colour are the original tile. To process the border pixels of this tile, those pixels marked in red are needed. But this pixels are not enough, as they need also to be processed by the function. To this process of the red pixels, those pixels marked in grey are needed, reaching a final 9x9 pixel size.

![Figure 5.1. Redundancy needed for pixel interdependences](image)

### 5.2 Overview of the proposed architecture

In this paragraph, the general structure of the implemented architecture for fingerprint verification is explained. It is divided in the following basic elements: image memory (in the later, IM), and Processing Units: Delta and Core, Minutiae and Verification Result. In Figure 5.2 a general scheme of the architecture is shown.

36
The main characteristics of each of the basic elements of the architecture is explained in the remaining paragraphs of this chapter.

5.3 Scheduling of the hardware units

In this paragraph, the scheduling of each of the elements of the architecture is explained. One important characteristic of the hardware system implemented is that the image is going to be divided is tiles of size 5 x 5 pixels. The basic unit of information handled by the architecture is a group of four 5 x 5 pixel tiles (100 grey-level pixels, therefore 25 bytes). This allows the development of a memory efficient architecture, as with this bigger tile, the memory redundancy explained in the previous paragraphs is reduced.

After a presentation of the scheduling of each function, further improvements are shown. Through the application of different techniques, such as parallel algorithms and different scheduling heuristics, either the number of operations or the time cycles expended to implement the different stages of the hybrid method are improved.

5.3.1 Minutiae Processing Unit Scheduling

This hardware unit executes the software functions explained in sections 4.2.3, 4.2.4 and 4.2.5, that perform image binarization, image thinning and minutiae extraction respectively. Each of this functions has, as input, the output of the previous one.

An study of the operation schedule is included to develop an efficient method to implement this functions in hardware at a reasonable time and resource cost. Some C/C++ code is shown, in the functions where its simplicity allows to, in order to show the reasoning.
underlying the scheduling study. In the more complex functions, the underlying algorithm is shown.

5.3.1.1 Binarization

Finally, the implemented binarization function is a simple local thresholding method. This was decided because a per-tile binarization method is needed and the slightly more complex methods that fulfil this requirement, do not show enough improvement to be considered.

The scheduling shows that applying concepts related to the parallel execution of algorithms, great improvements in execution time can be reached, at a reasonable hardware cost.

Figure 5.3 shows the C/C++ code of the implemented function.

```c
k=0;
for (int i = 0; i < 14; i++) {
  for (int j = 0; j < 14; j++) {
    k=k+fourtiles[i][j];
  }
}

for (int i = 0; i < 14; i++) {
  for (int j = 0; j < 14; j++) {
    if (fourtiles[i][j] > (k/256)){
      Binarized[i][j]=0;
    } else{
      Binarized[i][j] = 1;
    }
  }
}
```

Figure 5.3. C/C++ code of the binarization function

The input of the algorithm is the basic unit of information, the architecture deals with, a matrix of size 14x14 grey-scale pixels. On its sequential execution, this algorithm needs one cycle for each addition, as the result of every addition is accumulated in the same variable $k$. This sequential execution is shown in Figure 5.4. through a SSG (Scheduled Sequencing Graph). This representation allows to show the data dependency between operations. As there is 196 different input data, the execution of this sequential version takes 196 cycles to execute for the pixel additions, plus an extra cycle for the last division, performed as a Shift-Right.
Applying concepts related to algorithm parallelization, great improvements can be achieved. The method used, called balanced trees method, allows to decrease the time complexity of the algorithm from $O(n)$ to $O(\log n)$, allowing the execution of the whole process in 8 cycles ($2^8 = 256$), without rising the amount of simple operations involved. This concept is applicable to several common mathematical problems, e.g. the determination of the maximum of an array.

Figure 5.5 shows an SSG representing the behaviour of the implemented method. Instead of the sequential addition, the input values are added in pairs. The result of this operations is added again in pairs until there is only one remaining value, which happens to be the wished accumulated value. If in some cycle, the number of data left is odd, there is one value that remains stored. This value is kept, until the number is odd again in a subsequent cycle (the latest the last cycle). After the scheduling of the last addition, the comparisons between the input data and the calculated threshold can be scheduled in one extra cycle, although it would imply to double the number of resources; finally this operations are scheduled in 2 cycles. The total operation time of the schedule function is 11 cycles.
5.3.1.2 Thinning

The thinning function is the most computing intensive function of the whole system. The chosen algorithm, developed by Zheng et al., is inherently parallel, i.e. its execution in the n-th iteration only depends on the values of other variables in the (n-1)th iteration. The description of the algorithm steps is shown.

Being a point p and its neighbourhood:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p9</td>
<td>p2</td>
<td>p3</td>
</tr>
<tr>
<td>p8</td>
<td>p</td>
<td>p4</td>
</tr>
<tr>
<td>p7</td>
<td>p6</td>
<td>p5</td>
</tr>
</tbody>
</table>

Table 5.1. 8-neighbourhood of a pixel p

A pixel p is deleted if four different conditions are met:

Condition 1: p = 1; If pixel = 0, then break;
If pixel = 1, then cond[1] = 1;

Condition 2: if 3 < B(p) < 7, then cond[2] =1;
where B(p) is the number of non-zero neighbours of p.

Condition 3: if A(p) = 1, then cond[3] =1;
where A(p) is the number of 01 patterns in the ordered set p_2,p_3,...,p_9

Condition 4: if((p_2 * p_4 * p_6) = 0)&&(p_4 * p_6 * p_8) = 0), then cond[4]=0;
in the first sub-iteration and
((p_2 * p_4 * p_6) = 0)&&(p_4 * p_6 * p_8) = 0) in the second
The processing for each of the 144 pixels (although the tile has 196 pixels, the outer layer is only binarized, not thinned, leaving \(12 \times 12 = 144\) pixels to be thinned), can be done in parallel. This would imply concurrent reads of the information; but, as for each process, the only pixel to be written is the pixel \(p\), there is no impediment for the parallel execution of the thinning function in all the pixels at once, as there are not processes that have to write concurrently in the same variable.

To have an estimation of the execution time of the thinning algorithm, an scheduling tool has been used. It shows the data inter-dependencies in the execution process, for the execution of the algorithm in just one pixel. The final SSG of the whole thinning process would consist in the parallel execution of this process on the 144 concerned pixels. It is important to highlight that, the thinning algorithm, besides of its two sub-iterations execution, restarts working in the data set created in the last iteration, until no pixel value change has been performed in all the processed pixels.

The outcome of the scheduling is shown in Figures 5.6, 5.7 and 5.8. Each of them represents the scheduling of the calculation of each of the non trivial conditions of the thinning algorithm: respectively conditions two, three and four.

![Condition 2](image1.png)

Figure 5.6. Optimized SSG of the calculation of condition 2 in Zheng's algorithm

![Condition 3](image2.png)

Figure 5.7. Optimized SSG of the calculation of condition 3 in Zheng's algorithm
All this three conditions can be calculated in parallel for all the pixels in the tile. It can be observed that they involve no complex operations: only additions, logic operations and comparisons. The worst case is the calculation of condition 2 that takes 11 cycles, and can be considered the worst path and, therefore, the cycle execution time of the algorithm.

5.3.2 Delta-Core Processing Unit Scheduling

In this section, the scheduling of the Delta-Core unit is studied. Although the number of operations is smaller than in the Minutiae unit, its complexity is higher (involves several multiplications)

5.3.2.1 Orientation Field Estimation

This algorithm is the last computationally intensive process of the developed fingerprint verification application. Departing from a tile of the original image, it calculates a quantized estimation of the orientation field.

The whole fingerprint verification application has been designed avoiding the use of complex operations between operands, i.e. multiplications, divisions (by non 2^n divisors), square roots. Only simple operations, logical (And, Or), additions, subtractions, have been used. This is the only path to meet the objective of a design that allows its real implementation in a mobile embedded design while fulfilling real-time requirements.

Research has been carried to find a method computationally simple to find an estimation of the orientation field of the fingerprint grey-scale image[18][19]. The implemented one, is one of the most broadly used and computationally and conceptually simpler. Anyway, it implies the calculation of pixel gradients and, in a subsequent stage, the calculations of different calculations through the multiplication of the calculated gradients. In the last stage the atan2 function is used over the last calculated coefficients.

Although the complexity of the calculations of this algorithm cannot be avoided, it can be reduced through scheduling. The scheduling has been accomplished dividing the process in two sub-processes: gradient calculation and atan2 calculation.
5.3.2.1.1 Gradient Calculation Scheduling

It requires, for its implementation, the use of 80 multipliers and 50 byte adders, and takes only 3 cycles to execute. This outbounds the system specifications: it is not feasible.

Using other scheduling heuristics a solution closer to feasibility can be found. The list scheduling method allows to execute the gradient calculation process in a more resource efficient way, of course, at a time cost.

The List scheduling algorithm for minimum latency for a given resource bound is detailed in Figure 5.9.

```
LIST_L (G(V,E),a) {
  \text{l}=1;
  \text{repeat} {
    \text{for each resource type k = 1,2,...,r_{res} \{ }
    \text{Determine candidate operations U_{ik} ;}
    \text{Determine unfinished operations T_{ik} ;}
    \text{Select S_k that belongs to U_{ik} vertices, such that |S_k| + |T_{ik}| \leq a_k ;}
    \text{Schedule the S_k operations at step l ;}
    \text{\}}
  \text{l = l + 1;}
  \text{\}}
  \text{until (v_a is scheduled);}
  \text{return(t);}
}
```

Figure 5.9 List Scheduling Algorithm pseudo-code

The best possibilities found are the following:

- Using 8 multipliers and 5 adders, in an execution time of 15 cycles:
  
  This solution leads to a reduction of 90% in multipliers and adders, at a time cost of 400%.

- Using 15 multipliers and 13 adders, in an execution time of 6 cycles:
  
  This solution leads to a reduction of 81.25% in multipliers and 74% in adders, with a time cost of 100%.

In Figure 5.10 a reduced scheme of the scheduling of the gradient calculation part using the As-Soon-As-Possible heuristic is shown, while in Figure 5.11 shows a reduced SSG of the gradient process using the List Scheduling heuristic.

Although there is a time cost, any of this seems to be a solution closer to feasibility. The second one, at a small resource cost compared to the first one, allows the implementation of the function with a reasonable time cost.
Using the multipliers explained in Appendix A, based in the Karatsuba algorithm, as the multiplications will involve four cycles each one, the total amount of cycles to calculate the gradient will be 24. This multipliers are chosen due to their reduced complexity.

Subsequently, an atan2 calculation is performed over the coefficients calculated by the gradient process. Units to accomplish the atan calculation have been used in the SSG. The atan2 execution has been decomposed in simpler operations to calculate its SSG. It shows a high but feasible calculation complexity. The detailed ASAP SSG is shown in Figure 5.12.
5.3.2.2 Delta and Core Extraction

As it was disclosed in previous chapters, the Delta and Core Extraction is a simple process: it takes the quantized estimated orientation of four 5x5 tiles, calculates the predominant one and changes the value of those that are different to this predominant value. It takes 10 cycles to execute and does not involve complex operations.

5.4 Resource and timing budget of the developed methods

Finally, after the scheduling study, the expected cycle budget can be determined. At the same time, a resource budget at bit level has been performed. In Figure 5.13 the cycle study is presented, while in Figure 5.14, the resource budget study is shown (at bit level adders and byte level multipliers). This figures clearly show, also, that the previously made statement that the algorithm configuration of Figure 3.1. gives better results than the other option, shown in Figure 3.2.

The upper part of the algorithm can be implemented in two different ways. As the Binarization and Thinning methods work in 10x10 tiles, while the Orientation Calculation and Core and Delta Extraction work in 5x5 tiles, the designer can choose to implement the latter methods with four units running in parallel, or only one. The results in red, for the Orientation Field Estimation and Core/Delta Extraction, show the results if only one unit performs this tasks.

The Orientation Field Estimation unit takes 37 cycles to execute, resulting from the addition of the gradient calculation and atan calculation, described in paragraph 5.4. The Delta and Core extraction takes 10 cycles to execute. It can be seen that this part of the whole system takes less time to execute, while, as is shown in Figure 5.14, has a higher resource cost.

The Binarization algorithm, as shown in Figure 5.5. takes 10 cycles to execute. As discussed in paragraph 5.3.1.2, one iteration of the thinning algorithm, takes 11 cycles to execute. In computer simulations, an average of 20 iterations was found for this algorithm to finish the thinning of the fingerprint binarized pattern. This results in a total of 220 cycles.

![Figure 5.13 Cycle budget of the implemented methods](image-url)
Observing the figures in paragraphs 5.3.1 and 5.3.2, the number of basic hardware resources can be counted. The results are shown in Figure 5.14. The results are given in bit adders and byte multipliers used.

**Figure 5.14 Resource budget of the implemented methods**

5.5 Image Memory – Processing Units interconnection

Once, in future work, exact execution times are calculated for each of the units, the connection between the memory and the processing units, has to be designed to deliver the data in a synchronized with the aforementioned execution times.

It is important to recall, that the data unit of this architecture is 100 pixels (i.e. bytes), and to avoid tile interdependences, some redundancy has been added, reaching 196 pixels. The details are explained in section 5.1.

As the IM is not going to be written, the adequate design for the fingerprint image memory is a Read-Only-Memory (ROM).

5.5.1 Description of the hardware memory block

The IM is connected to the Controller. The Controller feeds the input control signals that control the read memory operations. In Figure 5.13 the interconnections are shown.

The fingerprint IM has the following ports:

- **Inputs:**
  - Address input (18 bits): Next position to be read address.
  - Read Enable signal (1 bit). There will be one signal per unit.
  - Clock

- **Outputs:**
  - Data Outputs: There is a port connecting the memory and each processing unit, to send the 196 bytes data packet to the units.
5.5.2 Scheduling of the memory data output

Once the functions had been designed in a hardware description language, probably one of them (the Delta and Core unit) will have an execution time shorter than the other. One trivial solution would be to feed the same data to both units at the slower unit consumption rate.

Through an example, in the following paragraphs, a design of a multi-port memory to feed timely multiple units at the same time is explained.

Suppose that the Delta-Core and Minutiae units have a different execution time: $t_\Delta$ is the execution time of the Delta-Core unit and $t_M$ is the execution time of the Minutiae unit. They have a relation $t_M \approx N \times t_\Delta$; where $N$ is a natural number. In this case, to improve the performance, the Minutiae unit can be duplicated $N$ times. This way, the $N$ minutiae units and the Delta-Core unit would have the same data consumption rate. To accomplish this, the memory has to have $N+1$ ports, $N$ connected to $N$ different Minutiae units, and one connected to the Delta-Core unit. The memory has $N+1$ different Read Enable signals. In Figure 5.18 the signals for the case $N=2$ are shown.
Once this units had been designed in a hardware description language, if the power and chip area restrictions of the application allow to this design can be also replicated, i.e. include M Delta-Core units, and M x N Minutiae Units.

5.6. Summary

In this chapter, the reasoning behind the election of a per-tile processing architecture has been explained. After giving an overview of the proposed architecture, an scheduling algorithm has been applied to the developed C/C++ functions in order to get an estimation of their execution time and, at meantime, to make an analysis of the amount of resources needed to implement them. In the functions where the expended hardware resources could be beyond the feasibility, an study has been made to develop custom designed hardware to reduce their complexity. Finally, an study of the interconnection between the architecture memory and the processing units has been carried to avoid a bottleneck in this stage.
6 Conclusions and future work

In this last chapter, the connection between the thesis objectives, stated in Chapter 1, and the developed work is shown. Future work to develop a final hardware in a hardware description language is discussed.

As first step, it is important to recall the thesis objectives, stated in chapter 1:

- The development of new algorithms suitable for a custom designed hardware architecture.
- The development of a software implementation of this algorithms, to prove its correctness and suitability for fingerprint images.
- The study of a hardware architecture design to implement the algorithms.

The fist two objectives have been met. New fingerprint algorithms, that use features from different algorithms from the state-of-the art, have been developed and a software implementation has been carried. This algorithms have shown enough correctness to consider this a good path. The main results can be observed in Table 4.1 and 4.2 and in Figures 5.13 and 5.14. This results show that the extraction of the Delta and Core points and Minutiae has reached promising results at a very low operational complexity.

In future works, several improvements could be carried:

- The development of a rejection unit, that would run in parallel with the Delta-Core and Minutiae. This unit would reject, in a short time, fingerprints with low quality, asking the user for a new acquisition.
- Although the results obtained in fingerprint classification and minutiae extraction are not good enough for a reliable matching, they have been reached at an extremely low computational complexity and reached an accuracy close to feasibility. This allows to add extra complexity to the algorithms to improve the results.
- The development of a new binarization algorithm. To improve the results in minutiae extraction this should be the main algorithm where the extra complexity should be added. This algorithm should be able to adapt to poor contrast regions without degrading its performance in the medium-high quality tiles. At meantime it should not introduce too much extra operational complexity.

Related to the study of the hardware architecture design, its development in a hardware description language remains undone. In this implementation, one key step, is the design of the custom design multipliers. Once both explained trends are implemented, one should be selected, and several different design should be carried in the selected trend to choose the optimal design.
Appendix A Custom design multipliers for gradient multiplications

Although, through scheduling, the number of multipliers involved in the execution of the hybrid method has been thoroughly reduced, further improvements are needed to reduce the calculation complexity of the implementation.

Instead of the multiplication of two integers with decimal part, as have been accomplished in the software implementations, software tests has been carried to know if less precise operations would lead to the same result. The Orientation Estimation does not need to be carried with a high level of accuracy. In the calculation of the first direction matrix, the directions are quantized in just four different values and in the second direction matrix, this values are checked, four by four and averages, as has been explained in section 4.1.1. This gives the designer the freedom to choose less precise data types; in the improbable case of an error it would be corrected in most of the cases by the quantization performed to calculate the second direction matrix.

Following this argument the data type of the gradients calculated in this function, that are the data involved in all the multiplications, are restricted to integers (no decimal part), allowing its representation with 8 bits. This encourages the implementation of custom design multipliers instead of the use of general purpose multipliers.

Research has been carried to find multiplication algorithms suitable for this purpose and develop the custom designed units.

Using general purpose hardware multipliers, multiplications can be performed using $O(1)$ time. But, as can be seen from the scheduling description on this chapter, there is one of the arms of the algorithm, the Minutiae Processing Unit, is much more computationally intensive than the one involving the gradient multiplications. This allows to explore slower multiplication procedures. In section 5.4, the scheduling of the memory-processing units communication was studied. In this case there is a significant difference in the processing times of both main units; the different options are studied.

Different solutions have been studied. In this chapter, two of them, radically different are presented:

- The first one is a direct, base 10 multiplication[20], that could also be applied binary.
- The second one is a high-speed algorithm[21]. Most of the steps of this algorithm, are performed in parallel.

A.1 Karatsuba algorithm

A.1.1 Introduction

The Karatsuba algorithm is an efficient procedure for multiplication[1]. It reduces the multiplication of two $n$-digit numbers to at most $3n^{1.585}$ single-digit multiplications. It is therefore faster than the classical algorithm, which requires $n^2$ single-digit products. If $n = 2^{10} = 1024$, in particular, the exact counts are $3^{10} = 59049$ and $(2^{10})^2 = 1,048,576$, respectively.
For sufficiently large \( n \), Karatsuba's algorithm is beaten by the Schönhage-Strassen algorithm. As in our case, \( n \) is small, this last method would not bring an improvement.

The Karatsuba algorithm is a notable example of the divide and conquer paradigm (recursively breaking down a problem into two or more sub-problems of the same type, until these become simple enough to be solved directly).

The standard procedure for multiplication of two \( n \)-digit numbers requires a number of elementary operations proportional to \( n^2 \), or \( O(n^2) \).

A.1.2 Algorithm description

The basic step of Karatsuba's algorithm is a formula that allows us to compute the product of two large numbers \( x \) and \( y \) using three multiplications of smaller numbers, each with about half as many digits as \( x \) or \( y \), plus some additions and digit shifts.

Let \( x \) and \( y \) be represented as \( n \)-digit strings in some base \( B \). For any positive integer \( m \) less than \( n \), one can split the two given numbers as follows

\[
x = x_1 B^m + x_0 \\
y = y_1 B^m + y_0
\]

where \( x_0 \) and \( y_0 \) are less than \( B^m \). The product is then

\[
xy = (x_1 B^m + x_0)(y_1 B^m + y_0) \\
= z_2 B^{2m} + z_1 B^m + z_0
\]

where

\[
z_2 = x_1 y_1 \\
z_1 = x_1 y_0 + x_0 y_1 \\
z_0 = x_0 y_0
\]

These formulas require four multiplications. Karatsuba observed that we can compute \( xy \) in only three multiplications, at the cost of a few extra additions:

Let \( z_2 = x_1 y_1 \)
Let \( z_0 = x_0 y_0 \)
Let \( z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0 \)

since

\[
z_1 = (x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0) - x_1 y_1 - x_0 y_0 = x_1 y_0 + x_0 y_1
\]

A.1.3 Algorithm example

Karatsuba's basic step works for any base \( B \) and any \( m \), but is most efficient when \( m \) is equal to \( n/2 \), rounded up.

Say we want to compute the product of 23 and 45. We choose \( B = 10 \) and \( m = 1 \):

\[
23 = 2 \times 10^1 + 3 \\
45 = 4 \times 10^1 + 5 \\
z_2 = 2 \times 4 = 8 \\
z_0 = 3 \times 5 = 15 \\
z_1 = (2+3) \times (4+5) - 8 - 15 = 22 \\
result = z_2 \times 10^2 + z_1 \times 10^1 + z_0 = 800 + 220 + 15 = 1035
\]
A.1.4 Application to the gradient calculations

This algorithm can be applied to the gradient calculation, either using binary numbers (vector length would be restricted to 7) or decimal numbers (restricted to 2). The latter case is interesting because, as the numbers are very short, the three multiplications involved in the calculation would be 1-digit multiplications. This would allow to include, in the multiplication units, look-up tables, with the pre-calculated outcome of the possible multiplications stored, allowing a faster implementation. Through the following schedule, shown in Figure A.1, it can be observed that this multiplications could be performed in 4 clock cycles. The involved multiplications can be performed as shift-left operations.

![Figure A.1. ASAP SSG of the Karatsuba multiplication function](image1)

If we followed this trend, and apply this multiplication algorithm to the final implementation, the scheduling finally decided (recall, using the list scheduling), shown in Figure 5.11, becomes the one shown in Figure A.2, where the multipliers have a delay of four cycles.

![Figure A.2. List scheduling SSG of the gradient multiplication function with the Karatsuba multipliers](image2)
The final cycle time of the gradient multiplications, following this approach, would be 32 cycles.

A.2 High-speed VLSI directed multiplier

A.2.1 Introduction

The other option is to use a VLSI implementation directed multiplier: several options have been studied[22]. The basic type of adders are those based on Shift/Add operations. To speed up the multiplication, there are two main options:

- Reducing the number of operands
- Performing faster additions

The main options in multiplier design are:

- High-radix multipliers
- Tree and Array multipliers.
- Divide and Conquer algorithms.

The one finally disclosed here is one of the conceptually simpler. The purpose of its disclosing is to show the complexity of a VLSI directed multiplier. This would allow, in the future work, to implement this and the previously explained Karatsuba algorithm (in binary and decimal versions) to make a decision about which trend to follow in the final implementation.

The main characteristics of this algorithm are the following:

- A time complexity of $O(\log_2 n)$.
- An operational complexity of $O(n^2)$
- An area complexity of $O(n^2 \log_2 n)$

One of the past challenging problems this algorithm came to solve was the lack of an algorithm with a computation time proportional to the logarithm of the word length and that, at meantime, could handle 2’s complement binary integer multiplication easily. This algorithm uses a redundant binary representation, where each digit can be 0 or 1 or -1.

A.2.2 Algorithm description

The algorithm has the following input and output:

- Input: A and B. Multiplicand and multiplier. Both n-bit binary integers
- Output: P, the product of A and B
The algorithm involves the following steps:

- Step 1: Convert A and B into equivalent redundant binary integers
- Step 2: Generate n n-digit partial product $P_{o,j}$ ($j = 0,1,\ldots,n-1$) for n multiplier digits. A partial product is zero, or the multiplicand itself or the negation of the multiplication.
- Step 3: Add up the partial products by means of a binary tree of redundant binary adders and obtain the product $P_{o,k}$ where $k$ is $\log_2 n$.
- Step 4: Convert the product $P_{o,k}$ into the equivalent binary number P.

Figure A.3 Example of the VLSI oriented multiplier behaviour
Bibliography


