



**Faculty of Engineering**

**Design Methodology and Analysis of Hash Function based-on Field Programmable Gate Array for Hash-based Message Authentication Code Application**

**Shamsiah binti Suhaili**

**Doctor of Philosophy  
2025**

Design Methodology and Analysis of Hash Function based-on Field  
Programmable Gate Array for Hash-based Message Authentication Code  
Application

Shamsiah binti Suhaili

A thesis submitted

In fulfillment of the requirements for the degree of Doctor of Philosophy

(Electronic Engineering)

Faculty of Engineering  
UNIVERSITI MALAYSIA SARAWAK  
2025

## DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Shamsiah binti Suhaili

Matric No.: 19010121

Faculty of Engineering

Universiti Malaysia Sarawak

Date: 27 August 2025

## **ACKNOWLEDGEMENT**

I am grateful to my supervisor, Assoc. Prof. Dr Norhuzaimin bin Julai, of Universiti Malaysia Sarawak (UNIMAS), for allowing me to conduct research under his guidance and for offering me guidance on research, writing, and presentation. His assistance was indispensable for the timely completion of this thesis.

I would like to extend my gratitude to all those who have made a direct or indirect contribution to this thesis. I am extremely grateful to all of my peers who have provided me with support in some capacity throughout my time at UNIMAS. I am grateful to all professors and lecturers for the knowledge they have shared with me regarding digital designs. Their expertise in this field really encourages and inspires me to continue my research.

Lastly, I would like to extend my sincere gratitude to my family for their unwavering support, encouragement, patience, and love during the entirety of my academic journey. My sincere gratitude to the Centre for Graduate Studies, for the advice and support given during my period of study in UNIMAS. I would also like to express my gratitude to the management and staff of UNIMAS for their assistance and cooperation in enabling me to complete my study in Sarawak. I am grateful to each of you.

## ABSTRACT

Security has grown in importance as a study issue in recent years. Many cryptographic techniques have been created to raise the performance of different information-protection systems. One of the categories of cryptography is a hash function. In this research, the field-programmable gate array (FPGA) device from Altera is used because of its higher performance and smaller area implementation. For this purpose, cryptography hash functions are selected as a design target. These algorithms have been studied to improve the speed, area implementation, and power consumption. However, some trade-offs exist among higher performance, area implementation, and low-power design. Cryptographic algorithms are one of the most important aspects of the hardware implementation of embedded system design. They offer methods and systems for executing secure and authenticated transactions via the network. FPGAs are chosen because they can provide high-speed design, are simple to debug, are inexpensive, and have a rapid time to market. The main objective of this research is to formulate unfolding transformation factor 2 and factor 4 methods into hash function design. Additionally, low-power design techniques using state encoding in hash function design also need to be considered. All these design methodologies aim to confirm the effect of HDL coding style on the performance of hardware algorithms implemented on FPGA. This research focuses on unkeyed algorithm hash functions, such as MD5, SHA-1, RIPEMD-160, and SHA-256, to evaluate and compare the design's performance, area implementation, maximum clock frequency, and power consumption. In this research, an FPGA device named Arria II GX from Altera is used because of its higher performance and smaller area implementation compared with others. Pipelining and unfolding methods are applied to obtain high maximum frequency and high throughput of hash functions. The proposed SHA-1 designs, implemented on the Arria II GX EP2AGX45DF29C4,

demonstrates an area of 548 combinational ALUTs and 907 registers. This is a significant reduction on the Stratix II GX. This indicates a 98.38% reduction in ALUTs and a 97.17% reduction in registers, highlighting the significant area efficiency of the proposed design. The most efficient design is the pipelined RIPEMD-160 on Arria II GX, which achieves a ratio of 3.5477 Mbps/ALUT, far greater than all other designs. In addition, the unfolding-based designs (factor 2 and factor 4) demonstrate competitive ratios of 0.5109 Mbps/ALUT and 0.5203 Mbps/ALUT, respectively, reflecting effective trade-offs between performance and resource utilisation. The proposed SHA-256 design with an unfolding factor of 4 had the greatest throughput of 4,196.30 Mbps, indicating a considerable improvement over earlier SHA-256 architecture. The proposed design demonstrates a 294.5% enhancement in throughput and the proposed architecture exhibits a 165.5% enhancement in throughput. A comparative analysis of HMAC-SHA-256 designs implemented on different FPGA platforms. With an area utilisation of 3,953 LUTs and 2,714 registers, the proposed HMAC-SHA256 design, when implemented on Altera Arria II GX with Quartus II 15.0, reaches a maximum frequency of 195.16 MHz.. The novelty and originality of this study lie in the combination of different methodologies that can improve the performance of hash function algorithms and integration of HMAC-SHA-256 designs. Besides, Gray encoding and clock gating can reduce the dynamic power dissipation caused by signal toggling. Furthermore, a combination of unfolding and pipelining can also increase the throughput of hash function algorithms.

**Keywords:** Digital design, FPGA, SHA-256, Throughput, Unfolding technique

***Kaedah Reka Bentuk dan Analisis Fungsi Hash Berasaskan Litar Bersepadu yang Boleh diprogramkan semula untuk Aplikasi Kod Pengesahan Mesej Berasaskan Hash***

**ABSTRAK**

*Keselamatan telah menjadi isu penyelidikan yang semakin penting dalam beberapa tahun kebelakangan ini. Pelbagai teknik kriptografi telah dibangunkan untuk meningkatkan prestasi sistem perlindungan maklumat. Salah satu kategori dalam bidang kriptografi ialah fungsi hash. Dalam kajian ini, peranti litar bersepadu yang boleh diprogramkan semula (FPGA) keluaran Altera telah digunakan kerana keupayaannya memberikan prestasi yang tinggi dan penggunaan kawasan perkakasan yang lebih kecil. Oleh itu, fungsi hash kriptografi telah dipilih sebagai sasaran reka bentuk. Algoritma ini dikaji dari aspek kelajuan pemprosesan, pelaksanaan kawasan, dan penggunaan kuasa. Walau bagaimanapun, terdapat kompromi antara prestasi tinggi, saiz pelaksanaan, dan reka bentuk kuasa rendah yang perlu diimbangi. Algoritma kriptografi merupakan komponen penting dalam pelaksanaan perkakasan bagi reka bentuk sistem terbenam kerana ia menyediakan kaedah untuk menjamin transaksi yang selamat dan disahkan melalui rangkaian. FPGA dipilih kerana ia mampu memberikan reka bentuk berkelajuan tinggi, mudah untuk dinyahralat, berkos rendah, dan mempercepatkan masa ke pasaran. Objektif utama kajian ini adalah untuk merumuskan kaedah transformasi unfolding dengan faktor 2 dan faktor 4 dalam reka bentuk fungsi hash. Selain itu, teknik reka bentuk kuasa rendah menggunakan kaedah pengekodan keadaan (state encoding) dalam fungsi hash juga turut diambil kira. Kesemua metodologi ini bertujuan untuk mengesahkan kesan gaya pengekodan bahasa penerangan perkakasan (HDL) terhadap prestasi algoritma perkakasan yang dilaksanakan pada FPGA. Kajian ini memfokuskan kepada fungsi hash tanpa kunci seperti MD5, SHA-1, RIPEMD-160, dan SHA-256, bagi tujuan penilaian dan perbandingan*

*prestasi reka bentuk dari segi pelaksanaan kawasan, frekuensi maksimum, dan penggunaan kuasa. FPGA Arria II GX daripada Altera telah digunakan dalam kajian ini kerana kelebihan dari segi prestasi dan kecekapan kawasan berbanding platform FPGA yang lain. Kaedah pengpaipan dan penyingkapan telah diaplikasikan untuk memperoleh frekuensi maksimum dan kadar data yang tinggi. Reka bentuk SHA-1 yang dicadangkan, yang dilaksanakan pada peranti Arria II GX EP2AGX45DF29C4, menunjukkan penggunaan kawasan sebanyak 548 ALUT gabungan dan 907 daftar (register), iaitu pengurangan ketara berbanding pelaksanaan pada Stratix II GX iaitu pengurangan sebanyak 98.38% dalam penggunaan ALUT dan 97.17% dalam daftar, sekali gus menonjolkan kecekapan kawasan yang tinggi. Reka bentuk paling efisien adalah reka bentuk RIPEMD-160 berasaskan pipelining di atas Arria II GX, dengan nisbah prestasi kepada kawasan sebanyak 3.5477 Mbps/ALUT, yang jauh lebih tinggi berbanding reka bentuk lain. Selain itu, reka bentuk berasaskan kaedah unfolding dengan faktor 2 dan faktor 4 menunjukkan nisbah yang kompetitif iaitu masing-masing 0.5109 Mbps/ALUT dan 0.5203 Mbps/ALUT, sekaligus mencerminkan keseimbangan yang efektif antara prestasi dan penggunaan sumber. Reka bentuk SHA-256 yang dicadangkan menggunakan faktor unfolding 4 menunjukkan kadar throughput tertinggi iaitu 4,196.30 Mbps, menandakan peningkatan ketara berbanding seni bina SHA-256 sebelum ini. Reka bentuk ini menunjukkan peningkatan throughput sebanyak 294.5%, manakala seni bina yang dicadangkan pula menunjukkan peningkatan sebanyak 165.5%. Satu analisis perbandingan telah dijalankan ke atas reka bentuk HMAC-SHA-256 yang dilaksanakan di atas pelbagai platform FPGA. Dengan penggunaan kawasan sebanyak 3,953 LUT dan 2,714 daftar, reka bentuk HMAC-SHA256 yang dicadangkan – apabila dilaksanakan pada Altera Arria II GX menggunakan perisian Quartus II versi 15.0 yang mampu mencapai frekuensi maksimum*

*195.16 MHz. Keaslian dan kebaharuan kajian ini terletak pada gabungan pelbagai metodologi yang mampu meningkatkan prestasi algoritma fungsi hash dan integrasi reka bentuk HMAC-SHA-256. Selain itu, penggunaan pengekodan Gray dan penyekatan jam dapat mengurangkan kehilangan kuasa dinamik akibat pertukaran isyarat. Tambahan pula, gabungan kaedah unfolding dan pipelining dapat meningkatkan throughput algoritma fungsi hash secara signifikan.*

**Kata kunci:** *Reka bentuk digital, FPGA, SHA-256, Truput, Teknik Unfolding*

## TABLE OF CONTENTS

|                                 | <b>Page</b> |
|---------------------------------|-------------|
| <b>DECLARATION</b>              | i           |
| <b>ACKNOWLEDGEMENT</b>          | ii          |
| <b>ABSTRACT</b>                 | iii         |
| <b>ABSTRAK</b>                  | v           |
| <b>TABLE OF CONTENTS</b>        | viii        |
| <b>LIST OF TABLES</b>           | xiii        |
| <b>LIST OF FIGURES</b>          | xvii        |
| <b>LIST OF ABBREVIATIONS</b>    | xxiii       |
| <b>CHAPTER 1: INTRODUCTION</b>  | 1           |
| 1.1 Research Background         | 1           |
| 1.2 Problem Statement           | 7           |
| 1.3 Motivation and Research Gap | 8           |
| 1.4 Research Objective          | 11          |
| 1.5 Hypothesis                  | 12          |
| 1.6 Research Scope              | 13          |
| 1.7 Significance of the Study   | 15          |
| 1.8 Thesis Organisation         | 17          |

|                                     |   |    |
|-------------------------------------|---|----|
| 1.9                                 | Thesis Contribution                                     | 18 |
| <b>CHAPTER 2: LITERATURE REVIEW</b> |   | 19 |
| 2.1                                 | Introduction  | 19 |
| 2.1.1                               | Overview of Hash Functions in Cryptography              | 20 |
| 2.1.2                               | Importance of Hash Function in HMAC Applications        | 21 |
| 2.2                                 | Overview of Common Hash Functions                       | 23 |
| 2.2.1                               | MD5 Design  | 24 |
| 2.2.2                               | SHA-1 Design  | 26 |
| 2.2.3                               | RIPEND-160 Design                                       | 27 |
| 2.2.4                               | SHA-256 Design  | 28 |
| 2.3                                 | Design Methodologies of Hash Function                   | 30 |
| 2.3.1                               | Iterative   | 30 |
| 2.3.2                               | Unrolling and Pipelining                                | 31 |
| 2.3.3                               | Unfolding   | 31 |
| 2.3.4                               | Low-power Design Approach                               | 32 |
| 2.4                                 | HMAC Overview and Comparative Analysis of Hash Function | 34 |
| 2.4.1                               | Properties of Cryptographic Hash Functions              | 35 |
| 2.4.2                               | Comparative Analysis of Hash Functions                  | 38 |
| 2.4.3                               | HMAC Design   | 62 |
| 2.5                                 | Summary   | 67 |

|   |           |
|---|-----------|
| <b>CHAPTER 3: METHODOLOGY</b>                           | <b>69</b> |
| 3.1 Digital Design Flow                                 | 69        |
| 3.2 Design Overview of Hash Function and HMAC Algorithm | 73        |
| 3.3 Dataflow and Behavioural Modelling                  | 78        |
| 3.3.1 Dataflow Modelling                                | 78        |
| 3.3.2 Behavioural Modelling                             | 78        |
| 3.4 Hash Function and HMAC Algorithm                    | 79        |
| 3.4.1 MD5 Algorithm                                     | 80        |
| 3.4.2 SHA-1 Algorithm                                   | 84        |
| 3.4.3 RIPEMD-160 Algorithm                              | 88        |
| 3.4.4 SHA-256 Algorithm                                 | 93        |
| 3.4.5 HMAC Algorithm                                    | 98        |
| 3.5 Proposed Design Methodology                         | 100       |
| 3.5.1 Unfolding Transformation Design Technique         | 103       |
| 3.5.2 Low Power Design Technique                        | 105       |
| 3.6 Modification to Hash Function and HMAC Design       | 107       |
| 3.6.1 MD5 Design  | 108       |
| 3.6.2 SHA-1 Design                                      | 119       |
| 3.6.3 RIPEMD-160 Design                                 | 129       |
| 3.6.4 SHA-256 Design                                    | 147       |

|  |  |     |
|--|--|-----|
| 3.6.5                                    | HMAC Design  | 159 |
| 3.7                                      | Summary  | 167 |
| <b>CHAPTER 4: RESULTS AND DISCUSSION</b> |  | 169 |
| 4.1                                      | Introduction   | 169 |
| 4.2                                      | Results and Discussion of MD5 Design Architecture        | 172 |
| 4.2.1                                    | MD5 Iterative Design                                     | 174 |
| 4.2.2                                    | MD5 Unfolding and Pipelining Design                      | 190 |
| 4.3                                      | Results and Discussion of SHA-1 Design Architecture      | 197 |
| 4.3.1                                    | SHA-1 Iterative Design                                   | 199 |
| 4.3.2                                    | SHA-1 Unfolding and Pipelining Design                    | 205 |
| 4.4                                      | Results and Discussion of RIPEMD-160 Design Architecture | 229 |
| 4.4.1                                    | RIPEMD-160 Iterative and Pipelining Design               | 231 |
| 4.5                                      | Results and Discussion of SHA-256 Design Architecture    | 247 |
| 4.5.1                                    | SHA-256 Design Architecture                              | 248 |
| 4.6                                      | Results and Discussion of HMAC Design Architecture       | 261 |
| 4.6.1                                    | Results of HMAC-SHA1 Design                              | 262 |
| 4.6.2                                    | Results of HMAC-SHA256 Design                            | 270 |
| 4.6.3                                    | Discussion of HMAC-SHA1 Design                           | 272 |
| 4.6.4                                    | Discussion of HMAC-SHA-256 Design                        | 275 |
| 4.7                                      | Summary  | 278 |

|                              |     |
|------------------------------|-----|
| <b>CHAPTER 5: CONCLUSION</b> | 280 |
| 5.1 Summary of the Research  | 280 |
| 5.2 Future Work              | 286 |
| <b>REFERENCES</b>            | 289 |
| <b>APPENDICES</b>            | 310 |
| <b>Appendix A</b>            | 310 |
| <b>Appendix B</b>            | 311 |
| <b>Appendix C</b>            | 313 |

## LIST OF TABLES

|            | <b>Page</b>   |    |
|------------|---|----|
| Table 2.1  | Compilation of Hash Functions in HMAC Applications                | 22 |
| Table 2.2  | Compilation of MD5 Hash Function in Different Applications        | 25 |
| Table 2.3  | Compilation of SHA-1 Hash Function in Different Applications      | 26 |
| Table 2.4  | Compilation of RIPEMD-160 Hash Function in Different Applications | 28 |
| Table 2.5  | Compilation of SHA-256 Hash Function in Different Applications    | 29 |
| Table 2.6  | State Encoding  | 33 |
| Table 2.7  | Compilation of MD5 Hash Function Design                           | 43 |
| Table 2.8  | Compilation of SHA-1 Hash Function Design                         | 48 |
| Table 2.9  | Compilation of RIPEMD-160 Hash Function Design                    | 53 |
| Table 2.10 | Compilation of SHA-256 Hash Function Design                       | 59 |
| Table 2.11 | Previous HMAC Design  | 64 |
| Table 3.1  | Behavioural Modelling   | 79 |
| Table 3.2  | Buffer Initialisations  | 81 |
| Table 3.3  | Constant $T[i]$ of MD5 Algorithm                                  | 83 |
| Table 3.4  | Shift, $S$ , and message value, $k$ of MD5 Algorithm              | 84 |
| Table 3.5  | Buffer Initialisation of SHA-1                                    | 86 |
| Table 3.6  | Constant $K_t$  | 87 |
| Table 3.7  | Round Function  | 88 |
| Table 3.8  | Initial Value   | 91 |
| Table 3.9  | Non-linear Function of RIPEMD-160                                 | 91 |
| Table 3.10 | Round Function  | 91 |
| Table 3.11 | Constant of RIPEMD-160  | 92 |
| Table 3.12 | Message Selection   | 92 |

|            |   |     |
|------------|---|-----|
| Table 3.13 | Shift Value   | 93  |
| Table 3.14 | Characteristics of SHA-256 Hash Function                                    | 94  |
| Table 3.15 | Buffer Initialisation of SHA-256 Algorithm                                  | 96  |
| Table 3.16 | RIPEMD-160 Input for Pipelined Design                                       | 134 |
| Table 3.17 | Comparison of Binary and Gray Encoding                                      | 159 |
| Table 4.1  | ASCII Code of Message Input of MD5 Design                                   | 172 |
| Table 4.2  | 512-bit message input   | 173 |
| Table 4.3  | Message in the Little-Endian Format   | 173 |
| Table 4.4  | Round 1 of Non-linear Function F  | 175 |
| Table 4.5  | Round 4 of Non-linear Function I  | 177 |
| Table 4.6  | Final MD5 Hash Function   | 178 |
| Table 4.7  | Maximum Frequency and Area Implementation of MD5 Design                     | 179 |
| Table 4.8  | Maximum Frequency of MD5b Design  | 180 |
| Table 4.9  | Area Implementation of MD5 Design (Arria II GX: EP2AGX45DF25C4)             | 180 |
| Table 4.10 | Simulation-based Power Estimation of Four Different Types of MD5 Designs    | 186 |
| Table 4.11 | Synthesis and Implementation of MD5 Design                                  | 188 |
| Table 4.12 | Comparison of MD5 Designs and Their Implementation                          | 194 |
| Table 4.13 | ASCII Code of Message Input for SHA-1 Design                                | 198 |
| Table 4.14 | 512-bit Message Input   | 198 |
| Table 4.15 | Message in the Big-Endian Format  | 198 |
| Table 4.16 | Ain Output ( $t = 1$ to $t = 10$ )  | 200 |
| Table 4.18 | Ain Output ( $t = 71$ to $t = 80$ )   | 202 |
| Table 4.18 | Comparative Overview of Early SHA-1 Iterative Hardware Designs (2002–2009). | 203 |
| Table 4.19 | A Output ( $t = 1$ to $t = 10$ )  | 207 |

|            |  |     |
|------------|--|-----|
| Table 4.20 | A Output ( $t = 31$ to $t = 40$ )  | 208 |
| Table 4.21 | Final SHA-1 Hash Function  | 210 |
| Table 4.22 | FPGA-based SHA-1 Implementation  | 216 |
| Table 4.23 | Area Implementations of SHA-1 and SHA-1 Unfolding  | 217 |
| Table 4.24 | Maximum Frequency of SHA-1 Design  | 218 |
| Table 4.25 | Maximum Frequency of SHA-1 Unfolding   | 220 |
| Table 4.26 | Synthesis and Implementation based on Altera Arria II GX                                     | 227 |
| Table 4.27 | Synthesis and Implementation based on Xilinx Virtex5 XC5VLX50T                               | 227 |
| Table 4.28 | ASCII Code of Message Input RIPEMD-160 Design  | 229 |
| Table 4.29 | 512-bit Message Input  | 230 |
| Table 4.30 | Message in the Little-Endian Format  | 230 |
| Table 4.31 | Output $B_{in0}$ and $B_{in1}$ for $1 \leq t \leq 16$  | 232 |
| Table 4.32 | Output of $outb$ and $routb$ for $1 \leq t \leq 11$  | 236 |
| Table 4.33 | Output $outb$ and $routb$ for $78 \leq t \leq 80$  | 238 |
| Table 4.34 | Synthesis and Implementation of RIPEMD-160 Hash Function                                     | 239 |
| Table 4.35 | Power Analysis of Both RIPEMD-160 and Pipelined RIPEMD-160 Designs                           | 240 |
| Table 4.36 | Comparison of RIPEMD-160 Design with Previous Studies  | 241 |
| Table 4.37 | Synthesis and Implementation Results of RIPEMD-160   | 243 |
| Table 4.38 | Synthesis and Implementation Results of SHA-256 Design and SHA-256 Unfolding Factor 2 Design | 252 |
| Table 4.39 | Comparison of Implementation Results with Other SHA-256 Designs                              | 253 |
| Table 4.40 | Comparison of Synthesis and Implementation Results with Other SHA-256 Designs                | 255 |
| Table 4.41 | ASCII Code of Message Input “Sample #1”  | 263 |
| Table 4.42 | Message in Hexadecimal Format  | 263 |
| Table 4.43 | Key Input for HMAC-SHA1  | 263 |

|            |   |     |
|------------|---|-----|
| Table 4.44 | Key $\oplus$ Ipad for HMAC-SHA1   | 263 |
| Table 4.45 | (Ko $\oplus$ Ipad)    Text for HMAC-SHA1  | 264 |
| Table 4.46 | Output of the First SHA-1 Design  | 264 |
| Table 4.47 | Ko $\oplus$ Opad  | 264 |
| Table 4.48 | (Ko $\oplus$ Opad)    Hash((Key $\oplus$ Ipad)  Text)                               | 265 |
| Table 4.49 | Output of the Second SHA-1 Design   | 265 |
| Table 4.50 | 20-byte of HMAC-SHA-1 Output  | 265 |
| Table 4.51 | First SHA-1 Hash Function   | 266 |
| Table 4.52 | HMAC-SHA-1 Output   | 268 |
| Table 4.53 | HMAC-SHA-1 Design   | 273 |
| Table 4.54 | Power Analysis of HMAC-SHA-1 Design   | 273 |
| Table 4.55 | Comparison of FPGA-based Implementation of Previous HMAC Designs                    | 274 |
| Table 4.56 | Comparison of FPGA-based Implementation Comparison of Previous HMAC-SHA-256 Designs | 276 |

## LIST OF FIGURES

|  | <b>Page</b> |
|--|-------------|
| Figure 1.1 Network encryption and authentication                           | 4           |
| Figure 2.1 Clock Gating (SemiEngineering, n.d.)                            | 34          |
| Figure 2.2 (a) and (b) Pre-image Resistance or One-Wayness                 | 36          |
| Figure 2.3 Second Pre-image Resistance or Weak Collision Resistance        | 37          |
| Figure 2.4 (a) and (b) Collision Resistance or Strong Collision Resistance | 37          |
| Figure 2.5 Hash Function   | 39          |
| Figure 3.1 Typical Design Flow   | 71          |
| Figure 3.2 FPGA Design Flow  | 72          |
| Figure 3.3 Design Overview of Hash Function and HMAC Algorithm             | 73          |
| Figure 3.4 Design Overview of Hash Function and HMAC Algorithm             | 74          |
| Figure 3.5 Flowchart for the Overall Hash Function and HMAC Design         | 76          |
| Figure 3.6 Dataflow Modelling  | 78          |
| Figure 3.7 MD5 Architecture  | 80          |
| Figure 3.8 Compression Function of MD5                                     | 82          |
| Figure 3.9 SHA-1 Compression Function                                      | 86          |
| Figure 3.10 Compression Function of RIPEMD-160 Block Diagram               | 89          |
| Figure 3.11 RIPEMD-160 Top-Level   | 90          |
| Figure 3.12 Message Preprocessing of SHA-256 Hash Function                 | 95          |
| Figure 3.13: Message Schedule of SHA-256 Algorithm                         | 96          |
| Figure 3.14 Compression Function of SHA-256 Algorithm                      | 97          |
| Figure 3.15 Message Authentication using Keyed- HMAC                       | 99          |
| Figure 3.16 Illustration of HMAC construct                                 | 100         |
| Figure 3.17 Iterative Looping (Henriguez et al., 2006)                     | 101         |

|             |  |     |
|-------------|--|-----|
| Figure 3.18 | Inner-Round Pipelining (Henriquez et al., 2006)                              | 101 |
| Figure 3.19 | Outer-Round Pipelining (Henriquez et al., 2006)                              | 102 |
| Figure 3.20 | Pipelining and Unfolding Transformation                                      | 102 |
| Figure 3.21 | An Example of an Original DSP Programme (Parhi, 1999)                        | 104 |
| Figure 3.22 | A DFG Corresponding to a DSP Programme (Parhi, 1999)                         | 104 |
| Figure 3.23 | The Unfolded DFG Corresponding to the 2-Unfolded DSP Programme (Parhi, 1999) | 105 |
| Figure 3.24 | State Encoding: (a) Binary Encoding and (b) Gray Encoding                    | 106 |
| Figure 3.25 | Gating the Clock: (a) Normal and (b) Clock-Gated Implementation              | 107 |
| Figure 3.26 | Design Methodology of Hash Function  | 107 |
| Figure 3.27 | Top-Level Architecture of MD5 Binary Design                                  | 108 |
| Figure 3.28 | Top-Level Architecture of MD5 Gray Design                                    | 109 |
| Figure 3.29 | Top-Level Architecture of MD5 Binary with Gating Design                      | 110 |
| Figure 3.30 | Top-Level Architecture of MD5 Gray with Gating Design                        | 111 |
| Figure 3.29 | Top-Level Architecture of MD5 Binary with Controller Design                  | 112 |
| Figure 3.32 | MD5 Compression Function Architecture  | 113 |
| Figure 3.33 | Counter MD5 module   | 115 |
| Figure 3.34 | Top Level of Unfolding MD5 Design  | 116 |
| Figure 3.35 | Compression Function of Unfolding MD5 Design                                 | 117 |
| Figure 3.36 | MD5 Pipelining and Unfolding Compression Function                            | 119 |
| Figure 3.37 | Top-Level Architecture of SHA-1 Design                                       | 120 |
| Figure 3.38 | SHA-1 Unfolding Compression Function   | 121 |
| Figure 3.39 | Top Level Architecture of SHA-1 Unfolding Design                             | 123 |
| Figure 3.40 | Top Level of SHA-1 Inner Pipelining Design                                   | 124 |
| Figure 3.41 | Top Level of SHA-1 Four-stage Pipelining Design                              | 125 |
| Figure 3.42 | Top Level of SHA-1 40-stage Pipelining Design                                | 127 |

|              |   |     |
|--------------|---|-----|
| Figure 3.43  | Top Level of SHA-1 Unfolding 40-stage Pipelining Design     | 128 |
| Figure 3.44  | RIPEMD-160 Iterative Architecture Design                    | 131 |
| Figure 3.45  | Top-Level Architecture of RIPEMD-160 Design                 | 132 |
| Figure 3.46: | Pipelined Design  | 133 |
| Figure 3.47  | RIPEMD-160 Pipelined Architecture Design for a Single Block | 135 |
| Figure 3.48  | Top-Level Architecture of Pipelined RIPEMD160 Design        | 137 |
| Figure 3.49  | Proposed RIPEMD-160 Unfolding Factor 2 Design               | 139 |
| Figure 3.50  | Coder Module of RIPEMD-160 Unfolding Factor 2               | 140 |
| Figure 3.51  | KConst Module of RIPEMD-160 Unfolding Factor 2              | 141 |
| Figure 3.52  | Message Module of RIPEMD-160 Unfolding Factor 2             | 141 |
| Figure 3.53  | Function Parallel of RIPEMD-160 Unfolding Factor 2          | 142 |
| Figure 3.54  | Coder Module of RIPEMD-160 Unfolding Factor 4               | 144 |
| Figure 3.55  | KConst Module of RIPEMD-160 Unfolding Factor 4              | 144 |
| Figure 3.56  | Message Module of RIPEMD-160 Unfolding Factor Four          | 145 |
| Figure 3.57  | Function Parallel Module of RIPEMD-160 Unfolding Factor 4   | 145 |
| Figure 3.58  | Top Level of SHA-256 Design                                 | 149 |
| Figure 3.59  | $Temp_{1o}$ Block Diagram Architecture                      | 150 |
| Figure 3.60  | $Temp_{2o}$ Block Diagram Architecture                      | 150 |
| Figure 3.61  | Cho (next_e,e,f) Function Architecture                      | 151 |
| Figure 3.62  | Majo(next_a,a,b) Function Architecture                      | 151 |
| Figure 3.63  | $0o(next_a)$ Architecture                                   | 151 |
| Figure 3.64  | $\Sigma_{1o}(next_e)$ Architecture                          | 151 |
| Figure 3.65  | $\sigma_{0o}$ Architecture                                  | 152 |
| Figure 3.66  | $\sigma_{1o}$ Architecture                                  | 152 |
| Figure 3.67  | Architecture of Temp11 Block Diagram                        | 153 |

|             |  |     |
|-------------|--|-----|
| Figure 3.68 | Architecture of <i>Temp21Block</i> Diagram                               | 154 |
| Figure 3.69 | Architecture of <i>Temp12</i> Block Diagram                              | 156 |
| Figure 3.70 | Architecture of <i>Temp22</i> Block Diagram                              | 156 |
| Figure 3.71 | Architecture of $\sigma_{0o}$  | 157 |
| Figure 3.72 | Architecture of $\sigma_{1o}$  | 157 |
| Figure 3.73 | Message Schedule of SHA-256 Unfolding Factor 4 Design                    | 158 |
| Figure 3.74 | Message Authentication using HMAC  | 160 |
| Figure 3.75 | Implementation of Proposed HMAC-SHA-1                                    | 161 |
| Figure 3.76 | Top-Level Architecture of HMAC-SHA1 Design                               | 162 |
| Figure 3.77 | Illustration of HMAC Construction  | 164 |
| Figure 3.78 | Implementation of Proposed HMAC-SHA-256                                  | 165 |
| Figure 4.1  | CAD Tool Simulation Flow Design  | 170 |
| Figure 4.2  | Non-Linear Function F for B_t (t = 0 to t = 8)                           | 175 |
| Figure 4.3  | Non-linear Function I for B_t (t = 57 to t = 63)                         | 176 |
| Figure 4.4  | Output of MD5 Hash Function  | 178 |
| Figure 4.5  | Simulation Results of MD5 Binary Design                                  | 181 |
| Figure 4.6  | Simulation Results of MD5 Gray Design                                    | 181 |
| Figure 4.7  | Simulation Results of MD5 Binary with Gating Design                      | 182 |
| Figure 4.8  | Simulation Results of MD5 Gray with Gating Design                        | 182 |
| Figure 4.9  | Simulation Results of MD5 Design with Input “a”                          | 183 |
| Figure 4.10 | Functional Simulation of MD5 Unfolding Design                            | 191 |
| Figure 4.11 | Timing Simulation of MD5 Unfolding Design                                | 191 |
| Figure 4.12 | Functional Simulation of MD5 Unfolding with Four-stage Pipelining Design | 191 |
| Figure 4.13 | Timing Simulation of MD5 Unfolding with Four-stage Pipelining Design     | 192 |

|              |  |     |
|--------------|--|-----|
| Figure 4.14  | Functional Simulation of MD5 Unfolding with 32-stage Pipelining Design   | 192 |
| Figure 4.15  | Timing Simulation of MD5 Unfolding with 32-stage Pipelining Design       | 192 |
| Figure 4.16  | Comparison of Performance-to-Area Ratio of MD5 Designs                   | 196 |
| Figure 4.17  | FMax of MD5 Design   | 197 |
| Figure 4.18  | Throughput of MD5 Design   | 197 |
| Figure 4.19  | $A_{in}(t = 1 \text{ to } t = 10)$                                       | 200 |
| Figure 4.20  | $A_{in}(t = 71 \text{ to } t = 80)$                                      | 202 |
| Figure 4.21: | $A(t = 1 \text{ to } t = 10)$  | 207 |
| Figure 4.22: | $A(t = 31 \text{ to } t = 40)$   | 208 |
| Figure 4.23  | Final Simulation Output of SHA-1 Hash Function                           | 209 |
| Figure 4.24  | Output of SHA-1 Hash Function  | 210 |
| Figure 4.25  | Functional Simulation of SHA-1 Inner Pipelining Design                   | 212 |
| Figure 4.26  | Timing Simulation of SHA-1 Inner Pipelining Design                       | 212 |
| Figure 4.27  | Functional Simulation of SHA-1 with Four-stage Pipelining Design         | 212 |
| Figure 4.28  | Timing Simulation of SHA-1 with Four-stage Pipelining Design             | 213 |
| Figure 4.29  | Functional Simulation of SHA-1 with 40-stage Pipelining Design           | 213 |
| Figure 4.30  | Timing Simulation of SHA-1 with 40-stage Pipelining Design               | 213 |
| Figure 4.31  | Functional Simulation of SHA-1 Unfolding with 40-stage Pipelining Design | 214 |
| Figure 4.32  | Timing Simulation of SHA-1 Unfolding with 40-stage Pipelining Design     | 214 |
| Figure 4.33  | Area Implementations for Both Proposed SHA-1 Designs                     | 221 |
| Figure 4.34  | Maximum Frequency and Throughput for Both Proposed SHA-1 Designs         | 222 |
| Figure 4.35  | Total Thermal Power Dissipation for Both Proposed SHA-1 Designs          | 222 |
| Figure 4.36  | Unfolding Transformation of SHA-1 Algorithm Compression Function         | 224 |

|              |   |     |
|--------------|---|-----|
| Figure 4.37  | Four Stages of SHA-1 Algorithm Architecture   | 225 |
| Figure 4.38  | Output B_in0 and B_in1 for $1 \leq t \leq 16$   | 232 |
| Figure 4.39  | Final Output of RIPEMD-160 Design   | 235 |
| Figure 4.40  | Output of outb and routb for $1 \leq t \leq 11$   | 236 |
| Figure 4.41  | Output outb and routb for $78 \leq t \leq 80$   | 237 |
| Figure 4.42  | Output of RIPEMD-160 Hash Function  | 238 |
| Figure 4.43  | Comparison of performance-to-area ratio   | 245 |
| Figure 4.44  | Simulation of RIPEMD-160 unfolding with factor 4 design   | 246 |
| Figure 4.45  | Functional Simulation of SHA-256 Design   | 249 |
| Figure 4.46  | Timing Simulation of SHA-256 Design   | 249 |
| Figure 4.47  | Functional Simulation of SHA-256 Unfolding Design   | 251 |
| Figure 4.48  | Timing Simulation of SHA-256 Unfolding Design   | 251 |
| Figure 4.49  | Timing Simulation Waveform of SHA-256 Design  | 257 |
| Figure 4.50  | Timing Simulation Waveform of Unfolding SHA-256 with Factor 2 Design                            | 258 |
| Figure 4.51  | Timing Simulation Waveform of Unfolding SHA-256 with Factor Four Design                         | 258 |
| Figure 4.52: | (a) FPGA Output Implementation Display and (b) Hardware Implementation of SHA-256 Hash Function | 259 |
| Figure 4.53  | HMAC-SHA-1: Hash $((K_o \oplus Ipad)    \text{Text})$   | 267 |
| Figure 4.54  | HMAC-SHA-1: Hash $((K_o \oplus Opad)    \text{Hash}((key \oplus Ipad)    \text{Text}))$         | 269 |
| Figure 4.55  | Output of HMAC- SHA-1   | 270 |
| Figure 4.56  | Simulation Waveform of HMAC-SHA-256 Design  | 271 |
| Figure 4.57  | HMAC Calculation of SHA-256 Design with a Specific Key  | 272 |
| Figure 4.58  | Maximum Clock Frequency of HMAC-SHA-256   | 272 |