

**A NEW DESIGN METHODOLOGY FOR
HIGH QUALITY DESIGN**

By

NAHREEN MAHERUKH

Master of Science

University of Dhaka

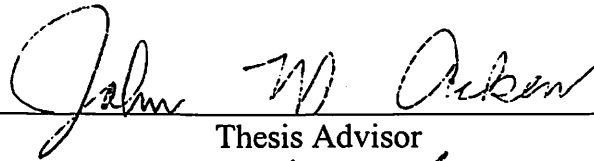
Dhaka, Bangladesh

1996

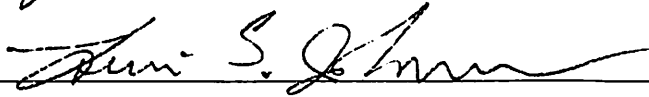
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In the Partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2004

**A NEW DESIGN METHODOLOGY FOR
HIGH QUALITY DESIGN**

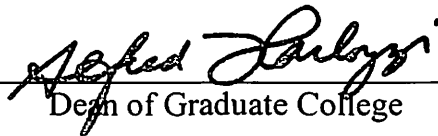
Thesis Approved:



Thesis Advisor







Dean of Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my advisor, Dr. John M. Acken, for his constructive guidance, support and encouragement. His intelligent suggestions and guidance have made this thesis feasible. His tolerance, patience, motivation and advice have been a constant source of inspiration and motivation that helped me to gain confidence. I also thank my other committee members, Dr. C. Latino and Dr. L.G. Johnson, for their assistance.

I like to sincerely thank my husband, Farhad Milki, for his unconditional love, support and endless encouragement.

I wish to thank my parents, A. Mannan Chowdhury and Fatema Chowdhury for their love, encouragements and blessings.

I dedicate this thesis to my dearest daughter, Nubaira Milki, who is the reason I have worked so feverishly to complete this project.

TABLE OF CONTENTS

Chapter	Page
I. Introduction	1
Introduction/ Objective	1
Proposal -BIST and RSP Combined	2
Design Project Description And Proposed Methodology	3
II. Background	6
Rapid System Prototyping	6
The Throw-Away Approach	8
The Evolutionary Approach	8
Digital System Testing	9
General Idea About Testing	9
Production Testing	10
Scan Based Stuck at Fault Test	11
Functional Test	11
IDDQ Test	12
Signature Analysis	12
Built in Self Test (BIST)	15
Exhaustive Testing	16
Pseudorandom Testing	16
Pseudo-Exhaustive Testing	17
Linear Feedback Shift Register (LFSR)	18
Biometrics For Identification	20
Summary	21
III. Software Prototype	22
Introduction/ Objective	22
Design Description	22
Equations Used For Calculations	24
Analysis Of Results	29
Conclusions	34

Chapter	Page
IV. Microcontroller Prototype	35
Introduction	35
Brief Description About 6811 Microcomputer	35
Design Description	37
Results	43
Conclusions.....	44
V. Final Implementation.....	46
Introduction/Objective	46
Design Description	46
Conclusions	49
VI. Conclusions.....	50
Conclusions	50
Recommendations	51
References	52
Appendices	54
Appendix A	54
Fortran Source Code	54
Appendix B	63
Three Sets Of Database	63
Calculations With Three Sets Of Data Base.....	65
Appendix C	71
6811 Access Program	71
Appendix D	83
Verilog Program	83
Appendix E	93
Graphs Showing The Hand Geometry Of Each Person	93
Appendix F	96
Summary of The Biometric Product Testing Final Report	96

LIST OF TABLES

Table	Page
I. Range of FMR (False Match Rate) for the three sets of data	33
II. Range of FNMR (False Non-Match Rate) for the three sets of data	34
III. Calculated Values With Input Dataset 1	65
IV. Calculated Values With Input Dataset 2	67
V. Calculated Values With Input Dataset 3	69

LIST OF FIGURES

Figure	Page
1. Traditional Functional Testing	11
2. Built-In Self-Test with LFSR Circuit	13
3. An 8 bit LFSR Circuit	19
4. Signature analysis with BIST	27
5. Flow chart for the Fortran Programming and Calculations	28
6. The Flow chart for the 6811 program	38
7. Block Diagram for the Simulation with Motorola MC6811	39
8. Memory map for 6811.....	40
9. Memory map for the variables in RAM	42
10. I/O ports of 6811.....	43
11. Block Diagram of the Access System	47
12. Detail Block Diagram of The Access System.....	48
13. Detection error trade-off: FMR vs. FNMR	100
14. Detection error trade-off: FAR vs. FRR	101

CHAPTER I

INTRODUCTION

Introduction/Objective

The design process utilizes prototyping to evaluate design decisions. In the final product, quality is improved by including production testing solutions during the design process. Security provided by identity verification is widely used in our daily life. Password, PIN (Personal Identification Number), mothers maiden name are just different kinds of identity verification. With Automated Teller Machines (ATM) people use their card with a PIN to get access to their money. This project aims at demonstrating improved access verification by including biometrics. The prototypes are used to statistically evaluate the improvement. Every card member's body weight, height and length of each finger will be stored in the database along with the account number and PIN. At the time of requesting access, the person's height, weight and the finger's length will be measured.

A system is developed that will crosscheck the measured values with the ones stored in database. Statistical analysis of prototype equations is used to determine the match rates and margin of error that can be tolerated. The combination of account number, PIN, height, body weight and fingers length would ensure higher security with less expense. The inclusion of a testing component ensures increased quality in the final system.

Proposal – Built-In Self-Test (BIST) and Rapid System Prototyping (RSP) Combined

Prototyping is a development approach that promotes the implementation of a pilot version of the intended product. A prototype is an executable model of a system that accurately reflects a chosen subset of its properties, such as display formats, computed results or response times. Rapid system prototyping refers to the capability of creating a prototype with significantly less time than it takes to produce an implementation for operational use. The goal of Rapid System Prototyping (RSP) is to quickly deliver a product that tests ideas, demonstrates feasibility, and refines requirements. Testing done during manufacturing is called manufacturing or production testing. These tests determine whether each product contains manufacturing defects. In order to achieve high quality, a good test strategy usually consists of a variety of test types. The capability of a circuit (chip, board, or system) to test itself is known as Built-In Self-Test (BIST). When Built-In Self-Test is incorporated in the system, at that point test is a part of the design. The ultimate in testable design is to make the design test itself. The BIST approach is the testing technique that is least dependent upon internal design details and is useful in a prototyping environment. This is valuable because in prototyping, the implementation details can always be changed. A prototype also helps to get feedback early in the development process. Adding Built-In Self-Test (BIST), a standard production test technique, to RSP is a method to add quality without slowing the product delivery.

In our access verification system we will use rapid system prototyping in order to evaluate different matching equations. Also, the prototype is used to evaluate how close the values should be matched to allow the access. We will incorporate the BIST

technique in the prototyping that will help to test the system in its early phase in less time, and the prototype will benefit from testing used to determine whether the device meets the specification.

Design Project Description And Proposed Methodology

To get access to the system each person will be asked to enter the account number, PIN, and then the height, weight and hand geometry (the length of each finger) will be measured. These are the measured data. In our system the inputs i.e. account number, PIN, height, weight and hand geometry are selected based on their commonality of use. Practically, the individual hand geometry measurements should be combined into one number but that is outside the scope of this research. Also five inputs (account number, PIN, height, weight and hand geometry) are used to increase the reliability of the system. To verify the match of the measured data some equations will be used. With these different equations the deviations of the measured data from the stored data, the ratios of these two values, the root mean square value will be determined. From these equations the one that has the lowest error rate will be chosen. The equation that best fits will be used in designing our second system prototyping. In the system there will be a database, which is a collection of data and a set of rules that will organize data by specifying certain relationship among data. In the database each person's height, weight, and hand geometry, account number and the PIN (Personal Identification Number) will be stored. Let us assume that the data that we have in the data base are the stored data which is denoted by S.D. and the input data (account number, PIN, height, weight and hand

geometry) that are measured to get access to the system are the measured data M.D. The modulus of the difference of these two values represents the absolute difference.

$$\text{Absolute Difference} = | S.D. - M.D |$$

For all the inputs (account number, PIN, height, weight and hand geometry) the above equation will be used. The account number is unique here. For the account number and PIN the absolute difference has to be zero. For other three inputs if the absolute difference are zero then it is obvious that the data matched. If the differences are not zero then based on how close the measured data are to the stored data, the percentage of likelihood to get access to the system will be determined.

Another way is to use the ratio of the stored value to the measured value.

$$\text{Ratio} = \frac{S.D}{M.D}$$

For the account number and PIN the ratio always has to be one.

The square of the stored data and the measured data will be add together and the average of the value will be taken. Then the square root of the value will give the root mean square value of that particular data.

$$R.M.S = \sqrt{\frac{((S.D)^2 + (M.D)^2)}{2}}$$

It is possible that for the same person sometimes the height or weight or hand geometry can differ from the stored value, where the difference may be small. If we design our system where all the measured data (inputs) have to match with the stored data exactly then it may not always be possible for a person to get access to his/her own account. That's why we need to define adjustable threshold for each input. This will help a person to get access even with small differences between the stored and the measured data. As

the account number is a unique number for every single person it ensures that even with the same height or weight one cannot automatically get access to the other person's account.

Variations in biometrics depend upon age. Younger people have increases in height and weight. Injuries can change hand geometry. Weight change can vary greatly with diet and exercise. All of these variations require that the database be updated regularly. Still normal variations of aof 1% for finger lengths and up to 10% for weight changes requires setting a matching threshold value for each biometrics.

The software prototype will be done using Fortran software. Absolute difference, ratios and RMS values will be calculated using three different input datasets and results will be compared. The equation that gives the best match rate results using the software prototype will be chosen. The system will be prototyped using the Motorola 68HC11A8 processor with the TExaS simulator to evaluate implementation decisions. Then the system will be design in verilog and will be tested. In all three steps signature will be calculated as a part of BIST.

CHAPTER II

BACKGROUND

This thesis describes a new methodology based upon combining established ideas. This Background chapter summarizes the concept and terminology for Rapid System Prototyping (RSP), Production Testing and Biometrics.

Rapid System Prototyping

The IEEE defined prototyping as “ A type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process”, [4].

A prototype is an executable model of a system that accurately reflects a chosen subset of its properties, such as display formats, computed results or response times. Prototypes are useful for formulating and validating requirements, resolving technical design issues and supporting Computer Aided Design (CAD) for both software and hardware components of proposed systems. Rapid prototyping refers to the capability of creating a prototype with significantly less time than it takes to produce an implementation for operational use.

A prototype may not satisfy all of the constraints on the final version of the system. For example, the prototype may provide only a subset of all the required functions, and may be

expressed in a more powerful or more flexible language than the final version. It may be less efficient in both time and space than the final version, and may have limited capacity. Full facilities for error checking and fault tolerance may not be included in prototyping. Such simplifications are often introduced to make the prototype easier and faster to build. To be effective, a partial prototype must have a clearly defined purpose that determines what aspects of the system must be faithfully reproduced and which ones can safely be neglected. Prototypes must be constructed and modified rapidly, accurately and cheaply. They do not have to be efficient, complete or robust and they do not have to use the same hardware, system software or implementation language as the delivered system. Software for rapid and inexpensive construction and modification of prototypes makes RSP feasible. The main reason for using prototypes is to get early design feedback economically. Prototype versions of most systems are much less expensive to build than the final versions. Prototypes should be used to evaluate proposed systems if acceptance by the customer or the feasibility of development is in doubt. The need for prototyping has become more urgent as systems being developed have grown more complex, more likely to have requirements errors, and more expensive to implement.

Rapid System Prototyping allows experiments on what will be hardware and what will be software. Prototypes facilitate the requirements phase for any type of software or hardware. Prototypes can demonstrate the system to the affected parties as a way to collect criticisms and feedback for update requirements, detect deviations from user expectations early, trace the evolution of the requirements, improve the communication and integration of the users and development personnel, and provide early warning of mismatches between proposed architectures and conceptual structure of requirements.

The Throw-Away Approach

The Throw-Away approach is most appropriate in the project acquisition phase where the prototype is used to demonstrate the feasibility of a new concept and to convince a potential sponsor to fund a proposed development project. The advantage of throw-away approach is that it enables the use of special purpose languages and tools, even if they introduce limitations that would not be acceptable in an operational environment or even if they are not capable of addressing the entire problem. In throw-away approach prototypes are usually built with a specific language for simulations. Refinements on the throw-away prototype mainly concern requirements [7]. The throw-away approach can be a stopgap for an inadequate level of technology and is most appropriate for rough system mock-ups used at the very early stages of a project.

The most apparent disadvantage of throw-away prototypes is spending implementation effort on code that will not contribute directly to the final product. The throw-away prototyping approach has been used in industry for about three decades. Prior to a large project, a study is performed to evaluate the feasibility and cost of the real system. Sometimes the project is canceled based on these studies.

The Evolutionary Approach

The Evolutionary Approach produces a series of prototypes in which the final version becomes the product. This approach depends on special tools and techniques because it is usually not possible to put a prototype into production use without significant changes to its implementation to optimize the code and to complete all of the details. In the evolutionary approach, support for automated program construction of systems is needed

and such tools can be very useful in this context, even if the resulting program are not very efficient. Refinements on the evolutionary prototype concern the product itself, functions, speed, memory, consumption etc [7]. Precise specification for the components of a prototype and clear documentation of its design are critical for effective software prototyping, as are tools for transforming and completing designs and implementations.

Digital System Testing

General Idea About Testing

The process of determining whether a product is functioning correctly or is defective (i.e. broken or faulty) is called testing. When a product is manufactured or assembled it may require testing before being sold or used. Each copy of the product to be tested is called the Device Under Test (DUT). The device could be a system, a board or a chip. The goal of testing is to identify which devices contain failure. Between two test sets, the one detecting more faulty devices while passing more fault free devices is better. The procedure of identifying defective units is called testing. The design verification testing verifies the correctness of the design. A test verifies correct operation against the circuit specifications combines two objectives: Verification that the design of the circuit correctly implements its specification and Verification that the manufacture of the circuit correctly implements the design. It is usually necessary to achieve these two objectives by separate procedures, [9]. The inputs used to verify the design typically do not produce a thorough enough manufacturing test. The manufacturing process is checked by applying test vectors to the physical piece of equipment, and the design process is checked by the simulating the design. The simulation process is called design verification

and uses a set of input vectors chosen specifically to verify the design meets the specification.

An n input combinational network could be tested thoroughly by applying 2^n input combinations and verifying that the correct output is obtained for each combination. The technique is sometimes called Exhaustive testing. It provides a thorough test but can require too much test time for networks with many inputs. Another testing known as pseudo exhaustive testing completely exercise combinational circuits without applying all possible circuit input combinations, [10]. For this testing first a circuit is broken into segments and then all possible input combinations are applied to each segment while the output of each segment is propagated to a primary output. Exhaustively testing the entire circuit requires over 2 million vectors but pseudo exhaustive testing requires fewer than 400 vectors. The goal of this approach is to detection all possible combinational faults. Diagnostic test is to locate the failure site on failed part and to find what might be wrong.

Production Testing

Testing done during manufacturing is called manufacturing or production testing. These tests determine whether each product contains manufacturing defects. The purpose of this testing is to test the manufactured parts to sort out those that are faulty.

In order to achieve a high quality of components it is accepted that a good test strategy usually consists of a variety of test types. For any particular device the tests used would typically be of the following

1. Scan based stuck at fault test
2. Functional test

3. IDDQ test
4. Signature Analysis

Scan Based Stuck-At Fault Test

Structural fault models assume that the components are fault-free and only their interconnections are affected. Usually faults affecting the interconnections are shorts and opens. A short is formed by connecting points not intended to be connected, while an open results from the breaking of a connection. A short between ground or power and a signal line can force the signal to a fixed voltage level. This logical fault consists of the signal being stuck at a fixed value and it is known as a stuck-at fault. By connecting all of the memory elements into scan chains, scan testing converts a sequential stuck-at test generation problem into a combinational stuck-at test generation problem.

Functional Test

Functional Testing is a verification of the intended function of the circuit. Failures are modeled at the register transfer or functional level in terms of variation in expected function.

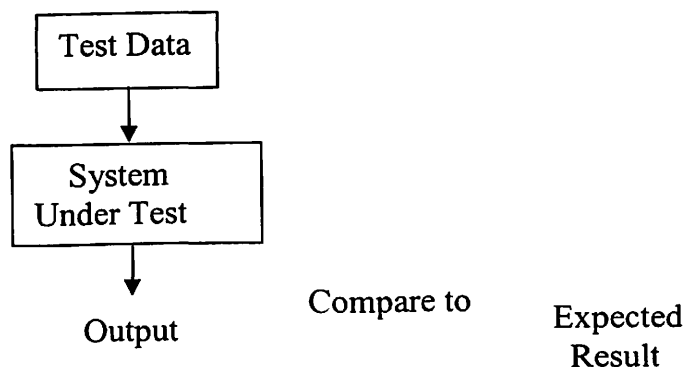


Figure 1: Traditional Functional Testing

The designer selects input vector sets based upon the intended functionality of the parts of the system. The expected output values are created by simulation. The Figure:1 shows the flow of functional testing.

IDDQ Test

I_{DDQ} testing is an effective technique for improving the quality and reliability of CMOS ICs. Local defects (which affects a few transistor in a chip) result from contamination during circuit fabrication, results in either extra or missing material at a given site. These include gate oxide shorts, unintended bridges between nodes and missing connections. Such defects particularly shorts can result in unintended connections between power and ground. As a result the circuit draws current in its static state and the defects can be detected by monitoring the quiescent power supply current (I_{DDQ}). Many defects which may be observed via I_{DDQ} are undetectable with conventional voltage testing, [6].

Signature Analysis

Polynomial division with a Linear Feedback Shift Register (LFSR) uses the remainder left in the register after completion of the test as the retained value for comparison with the good remainder. It is an extension of the well known CRC (Cyclic Redundancy Check) code and is the most popular data compression technique and it is easily modified for use with multiple output circuits. The remainder is usually called a signature and the technique is called signature analysis, [5].

An LFSR is a shift register that when clock advances the signal through the register from one bit to the next most significant bit, some of the outputs are combined in exclusive-OR

configuration to form a feedback mechanism. A Linear Feedback Shift Register can be formed by performing exclusive-OR on the outputs of the flip-flops together and feeding those outputs back into one of the flip-flops. Any data such as the test response results from a circuit can be compressed into a code word by an LFSR. This code word, the remainder from the division process, is called the signature of the input data stream and the LFSR itself is called the signature analyzer.

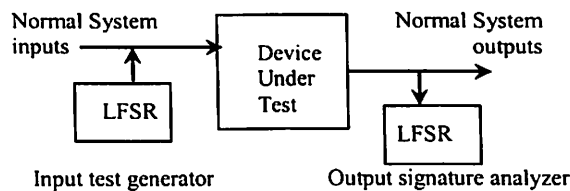


Figure 2(a)

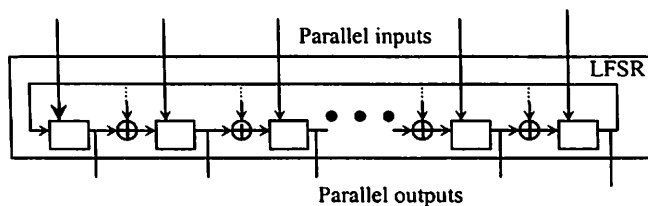


Figure 2(b)

Figure 2: Built-In Self-Test with LFSR circuits

Figure 2(b) shows a generic LFSR that can be used as an input generator or a signature analyzer. One concern with the use of CRC check words as the signature of the circuit output sequence is the possibility that an erroneous sequence from a faulty circuit will be compressed into the same signature as the fault free circuit. This phenomenon is called masking since the effect of the fault is masked by the compression process. Masking is a loss of information caused by the compression of the output sequence. The masking

probability of LFSR signature analysis can be reduced, independent of types of errors exhibit by the circuit, [3]. They are

1. Lengthen the LFSR
2. Repeating the test
3. Using variable-shift MISR (Multiple Input Signature Registers)

Signature testing on a newly built part requires knowing the correct or good machine signature for the part. The signature can be obtained in several ways. From a test engineering approach, the simplest is to take a part which has already passed functional testing in the system, run the part against the actual test patterns and save its signature as the reference for the new production. If testing is required before system installation, the simplest approach is to simulate both the circuit and the signature analyzer against the actual test patterns. An alternative to complete simulation is to simulate up to some break point say at the first 100 tests, and to obtain the expected signature at the point. Now the first 100 tests are applied to actual hardware and the hardware signature is checked against the simulated breakpoint signature. A mismatch between the two requires the hardware be reworked until it passes the checkpoint test. When the hardware passes the checkpoint test it is fault free. Now the remainder of the test patterns are applied to obtain the final hardware signature which is saved as the putative good machine signature,[3].

Signature analysis using selective feedback of various stages of a shift register fed by the data stream being created by a circuit is a powerful technique for coping with a large volume of test response data.

Built - In Self -Test (BIST)

Testing of digital circuits is a major portion of the effort in their design, production, and use. As the electronic industry evolved from discrete components through early integrated circuits to an increasingly high level of integration, the demands on testing methods and effectiveness have caused test technology to evolve to a high degree of sophistication. As the number of circuits that can be integrated onto one piece of silicon approaches twenty million it would seem that some small portion of those circuits could be devoted to testing. This concept is called Built-In Self-Test. The capability of a circuit (chip, board, or system) to test itself is known as Built-In Self-Test. There are two categories of BIST. In on-line BIST, testing is done during the normal operating conditions. On-line BIST is either concurrent or non-concurrent. Concurrent on-line BIST is a form of testing which occurs simultaneously with normal functional operation. This testing is usually done using coding techniques or duplication and comparison. In non-concurrent on-line BIST, testing is carried out while a system is in the idle state. Systems, boards, and chips can be tested in this mode. Off-line BIST is either Functional or Structural. Functional offline BIST deals with the execution of a test based on a functional description of the circuit under test and sometimes employs functional, or high level, fault model. Structural off-line BIST deals with the execution of a test based on the structure of the circuit under test. When Built-In Self-Test is incorporated into a digital network to meet a testability specification, the distinction between design and test becomes unclear. At that point test is a part of the design. The ultimate in testable design is to make the design test itself. Building test into the design, as might be expected, consumes added circuit and I/O overhead, but at the same time results in visible

reductions to the costs of testing when compared with an external test using automatic test equipment. Built-in testing achieves these savings by eliminating (or at least reducing) the costs of test pattern generation and fault simulation, decreasing the time required for tests, by running tests at circuit speed, simplifying the external test equipment, and easily adapting to engineering changes.

As there are many ways to build in the processing of test responses, so are there many ways to generate the tests. The various forms of testing are the following: Exhaustive Testing, Pseudorandom Testing, Pseudo-exhaustive Testing.

Exhaustive Testing

Exhaustive testing deals with the testing of n-input combinational circuits where all 2^n inputs are applied. Since all possible test patterns are applied, all detectable single and multiple stuck faults are detected. The tests are generated with any process that cycles exhaustively through the circuit input space, such as the binary counter, a Gray code generator or an n-stage linear feedback shift register. Exhaustive testing for high input pin count structures requires relatively long test times, but the circuit can be partitioned into sub-circuits, each of whose input pin count is low enough to permit exhaustive testing in a reasonable amount of time [10].

Pseudorandom Testing

In Pseudorandom Testing, testing a circuit is done with test patterns that have many characteristics of random patterns but where the patterns are generated deterministically and are repeatable. Pseudorandom patterns can be generated with or without replacement.

Pattern generation with replacement means that a test pattern may be generated more than once, without replacement means that each pattern is unique. Not all 2^n test patterns need to be generated. Pseudorandom Testing is applicable in both combinational and sequential circuits. Pseudorandom testing is a type of Built-In Self-Test that performs a structural test of the network involved.

Pseudo-Exhaustive Testing

Pseudo-exhaustive testing usually requires fewer test patterns but achieves many of the benefits of exhaustive testing. It relies on various forms of circuit segmentation and attempts to test each segment exhaustively. This technique completely exercises combinational circuits without applying all possible circuit input combinations. For this technique, first a circuit is broken into segments and then all possible input combinations are applied to each segment while the output of each segment is propagated to a primary output.

Let us consider a 21 input combinational circuit. Exhaustively testing the entire circuit requires over 2 million vectors (exactly $2^{21} = 2,097,152$). Now considering segmenting the circuit into three circuits with 7 inputs each. Using pseudo exhaustive testing on this segmented circuit requires fewer than 400 vectors (exactly $3 * 2^7 = 384$). The goal of this approach is to detection all possible combinational faults, [2].

Linear Feedback Shift Register (LFSR)

An LFSR is a shift register that when clock advances the signal through the register from one bit to the next most significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A Linear Feedback Shift Register can be formed by performing exclusive-OR on the outputs of two or more flip-flops together and feeding those outputs back into one of the flip-flops. LFSR make extremely good pseudorandom pattern generators. When the outputs of the flip-flops are loaded with the seed value (which is anything except all 0's, which would cause the LFSR to produce all 0 patterns) and when the LFSR is clocked it will generate a pseudorandom pattern of 0's and 1's.

A LFSR consisting of n flip-flops would go through the states 0, 1, ... 2^{n-1} , 0, 1. The maximum number of states for such a device is 2^n . An n-bit shift register cycles through at most n states. The output sequence generated by such a device is also cyclic.

The theory behind the use of an LFSR for signature analysis is based on the concept of polynomial division where the remainder left in the register after completion of the test process corresponds to the final signature. The input sequence of a LFSR can be represented by the polynomial $G(x)$ and the output sequence by $Q(x)$. If the initial state of the LFSR is all 0's, let the final state of the LFSR be represented by the polynomial $R(x)$. Then it can be shown that these polynomials are related by the equation

$$\frac{G(x)}{P(x)} = Q(x) + \frac{R(x)}{P(x)}$$

Where $P(x)$ is the characteristic polynomial of the LFSR. Hence an LFSR carries out (polynomial) division on the input stream by the characteristic polynomial, producing an output stream corresponding to the quotient $Q(x)$ and a remainder $R(x)$, [1]. For a

primitive polynomial the LFSR will generate the maximum sequence no matter what the seed values are. There are two types of LFSR. They are 1) Type 1 LFSR and 2) Type 2 LFSR.

Type 1 LFSR is easier to understand and work with mathematics but Type 2 LFSR is easier to implement in hardware. Another name for type 1 LFSR is External XOR because the feedback is XORed to one big XOR gate external to the serial registers. Type 2 is also called Internal XOR because the feedback is XOR's between serial register stages.

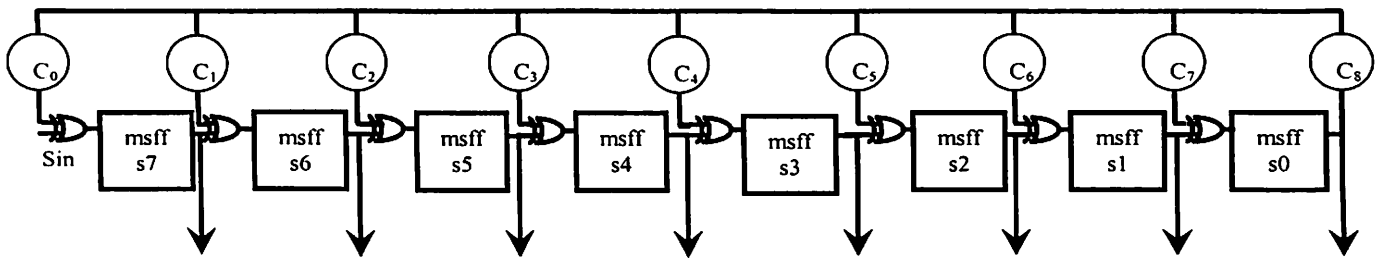


Figure 3: An 8 bit LFSR Circuit (Type 2)

We will use an 8-bit type 2 LFSR as shown in Figure: 3, in our system. An 8-bit LFSR cycles through at most 8 states. An LFSR goes through a cyclic or periodic sequence of states and the output produced is also periodic. The maximum length of the period is $2^n - 1$, where n is the number of states. The characteristic polynomial associated with a maximum length sequence is called a primitive polynomial.

Biometrics for Identifications

Biometrics are automated methods of recognizing a person based upon a physiological or behavioral characteristic. Some examples features measured in this method are face, fingerprints, hand geometry, handwriting, iris, retina, wrist vein, and voice. Biometric technologies are becoming the foundation of an extensive array of highly secure identification and personal verification solutions. In identification mode, the biometric system identifies a person from the entire enrolled population by searching a database for a match. In verification mode, the biometric system authenticates a person's claimed identity from his/her previously enrolled pattern. As the level of security breaches and transaction fraud increases, the need for highly secure identification and personal verification technologies is becoming apparent. Using biometrics for identifying and authenticating human beings offers some unique advantages.

Some terms that are used to describe the accuracy of biometric systems include false-acceptance rate (percentage of impostors accepted), false-rejection rate (percentage of authorized users rejected), and equal-error rate (when the decision threshold is adjusted so that the false- acceptance rate equals the false-rejection rate). When discussing the accuracy of a biometric system, it is often beneficial to talk about the equal-error rate or at least to consider the false-acceptance rate and false-rejection rate together, [8]. For many systems, the threshold can be adjusted to ensure that virtually no impostors will be accepted. This often means an unreasonably high number of authorized users will be rejected. Biometric-based authentication applications include workstation/ network/ domain access, single sign-on, application logon, data protection, remote access to resources, transaction security and web security. Utilizing biometrics for personal

authentication is becoming convenient and considerably more accurate than current methods (such as the utilization of passwords or PINs). This is because biometrics links the event to a particular individual on the other hand someone other than the authorized user may use a password or token. It is convenient as nothing to carry or remember and is accurate which provides for positive authentication.

Biometric technologies are ideally suited to provide highly secure identification and personal verification solutions. Biometric-based personal authentication has multiple applications in commerce, the federal, state and local governments, in the military and in commercial applications. Biometrics has a high priority in the government sectors and also in the business world, especially in fast-growing sectors such as mobile appliances and e-commerce. Open system standards increase user's confidence by preventing sole source lock-in and are vital for the growth of the global economy. The biometrics industry's maturity is demonstrated by its commitment to the development of the required biometric standards. Evidence of the growing acceptance of biometrics is the availability in the marketplace of biometric-based authentication solutions that are becoming more accurate, less expensive, faster and easy to use.

Summary

This Background chapter has described how RSP improves product development cycle. Production testing was described, which improves final product quality. Extra explanation was provided for BIST. Identity verification improvement by adding biometrics has also been explained. The rest of this thesis will describe a new design methodology that combines all of these ideas.

CHAPTER III

SOFTWARE PROTOTYPE

Introduction/Objective

For the Rapid System Prototyping (RSP) Approach, the initial prototype is usually written in a general software language. This chapter describes the Fortran prototype of the access verification system. The program runs with three different sets of data. The calculation is done using three equations and the values are compared. The prototype is used to evaluate the equations and to select the equation that gives the best result as measured by match rates.

Design Description

The following steps are planned for this prototype. First the measured data are compared with the database data, and absolute differences, ratios and root mean square values are calculated. Threshold values are applied and percentage of correct data, percentage of incorrect acceptance, percentage of incorrect rejection are calculated. Then the equation that performed best is picked from this calculation. That equation would be further used for designing the system for the next prototype version.

The Fortran prototype for my program is used to demonstrate how to get access to the system. To get access to the system each person will be asked to enter the account

number, PIN and height, weight, and hand geometry (i.e. each person's five finger-lengths from a particular point of the palm) will be measured. For hand geometry always the right hand will be used. These are the measured data. In the system, the database contains each person's account number, PIN (Personal Identification Number) height, weight, and hand geometry. The measured data is compared with the stored database data and the results are checked whether the data matched or not. To verify the accuracy of the measured data different equations are used. The three different equations: the absolute difference of the measured data from the stored data, the ratios of these two values, the root mean square value have determined the comparison value. From these equations the one that verified the system most accurately is chosen. The equation that fits best is chosen in designing the next system prototyping.

For the same person the height or weight or hand geometry can differ from the stored value, where the difference may be small. It is always possible that someone can gain weight or lose weight, the height may differ using different pairs of shoes and also the length of the fingers may vary for the nails. If we design our system where all the measured data's (inputs) have to match with the stored data exactly then sometimes it will not be possible for a person to get access to his/her own account. That is why an adjustable threshold is defined for each input. This will allow a person to access with small deviation in the measured data with the stored one. As the account number is a unique number for every single person it ensures that even a person with the same height or weight will not automatically get access to the other person's account.

In my access verification system I have used rapid system prototyping, which will evaluate whether the device meets the specification in short time. It also helps make

design decisions such as which equations and thresholds to implement. In prototyping I have incorporated the Built-in Self-Test (BIST) technique that will test the system in its early phase in less time.

Equations Used For Calculations

The Prototype used the match rates to evaluate the equations. After evaluation, the equation that gives the best result as measured by the match rates will be selected. This section describes the equations used for the calculation. As mentioned in Chapter-I the equations are as follows. The data that is in the data base are the stored data which is denoted by S.D. and the data (Account number, PIN, height, weight and hand geometry) that are measured to get access to the system are the measured data M.D. The absolute value of the difference of these two values represents the absolute difference.

$$\text{Absolute Difference} = | \text{S.D.} - \text{M.D} | \dots\dots\dots (1)$$

The second equation is the ratio of the stored data to the measured data, which is represented by

$$\text{Ratio} = \frac{\text{S.D}}{\text{M.D}} \dots\dots\dots (2)$$

the third equation find the Root Mean Square value of the measured data and the stored data. The square of the stored data and the measured data will be added together and the average of the values will be taken. Then the square root of the value will yield the root mean square value of that particular data.

$$\text{R.M.S} = \sqrt{\frac{((\text{S.D})^2 + (\text{M.D})^2)}{2}} \dots\dots\dots (3)$$

For the output data set the number of matched data is M, the number of data that are rejected is R. TT represents the number of test data. So we can write

$$TT = M + R \dots\dots\dots (4)$$

CM represents the number of data that matched correctly and CR is the number of data that are rejected correctly. So the number of correct data C can be written as

$$C = CM + CR \dots\dots\dots (5)$$

In this system every persons account number is unique and only when it exactly match then the rest of the data will be checked to give access. But for height, weight and hand geometry any two persons can have the same height or weight or hand geometry. When two or more persons have the same value (i.e. height or weight or hand geometry) then in calculating the absolute difference or the ratio for height or weight or hand geometry we will get 0 or 1. These are incorrectly matched data.

Now if IM represents the number of data that match incorrectly and IR represents the number of data that are rejected incorrectly then the total number of test TT can be written as

$$TT = CM + CR + IM + IR\dots\dots\dots (6)$$

Where CM and CR are the number of data that matched correctly and the number of data that rejected correctly. Using these equations we can find the value of C, TT, IM, IR.

The percentage of the correct data can be calculated by the ratio of the number of correct data C to the total number of test data TT

$$\text{Percentage of Correct Data} = \frac{C}{TT} \dots\dots\dots (7)$$

For example in our test dataset we have 225 data and for one data set, the number of correct data is 224 then the percentage of correct data is $224/225 = 99.56\%$.

The false match rate and false non-match rate measure the accuracy of the matching process. The percentage of the false acceptance can be calculated by the ratio of the number of data that matched incorrectly IM to the total number of test data TT. So the False Match Rate (FMR) can be written as

$$\text{False Match Rate (FMR)} = \text{Percentage of Incorrect Acceptance} = \frac{IM}{TT} \dots\dots (8)$$

For example in our test dataset, we have 1 data that matched incorrectly in a total 225 test data then the percentage of incorrect acceptance is $1/225 = 0.44\%$.

When the account number is matched but there is a slight difference in height or weight or in hand geometry then we will get the data for these (height or weight or hand geometry) fields which will not match with the stored data. These are the incorrectly rejected data. The percentage of incorrect rejection can be expressed by the ratio of the number of incorrectly rejected data IR to the total number of test data TT

$$\text{False Non Match Rate (FNMR)} = \text{Percentage of Incorrect Rejection} = \frac{\text{IR}}{\text{TT}} \dots\dots (9)$$

For the Account Number and PIN to verify access I have only used the absolute difference between the stored data and the measured data in Fortran programming. Because most access verification system already have a procedure to check Account number and PIN. For the other fields we will use the absolute difference, ratios and the root mean square values and then will select one for each field that gives the best result to that field.

BIST was implemented in the Fortran prototype as shown in Figure: 4.

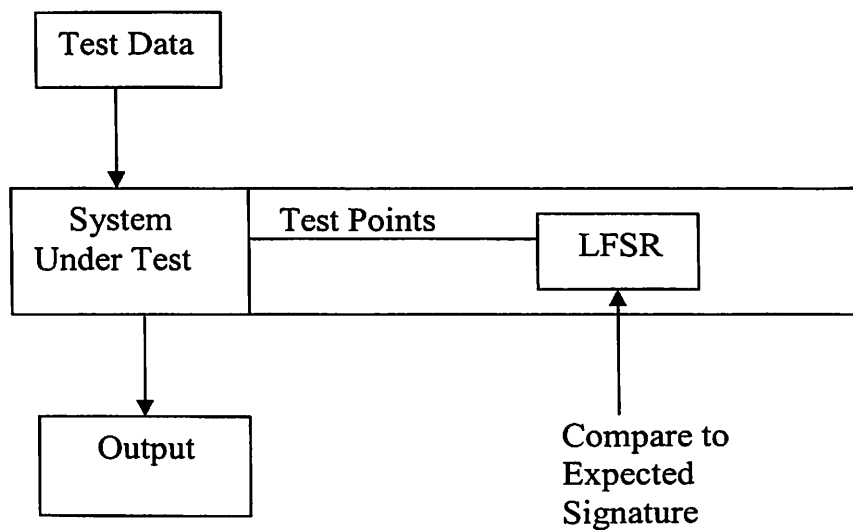


Figure 4: Signature Analysis with BIST

The flowchart for the Fortran programming and calculation is shown in Figure 5.

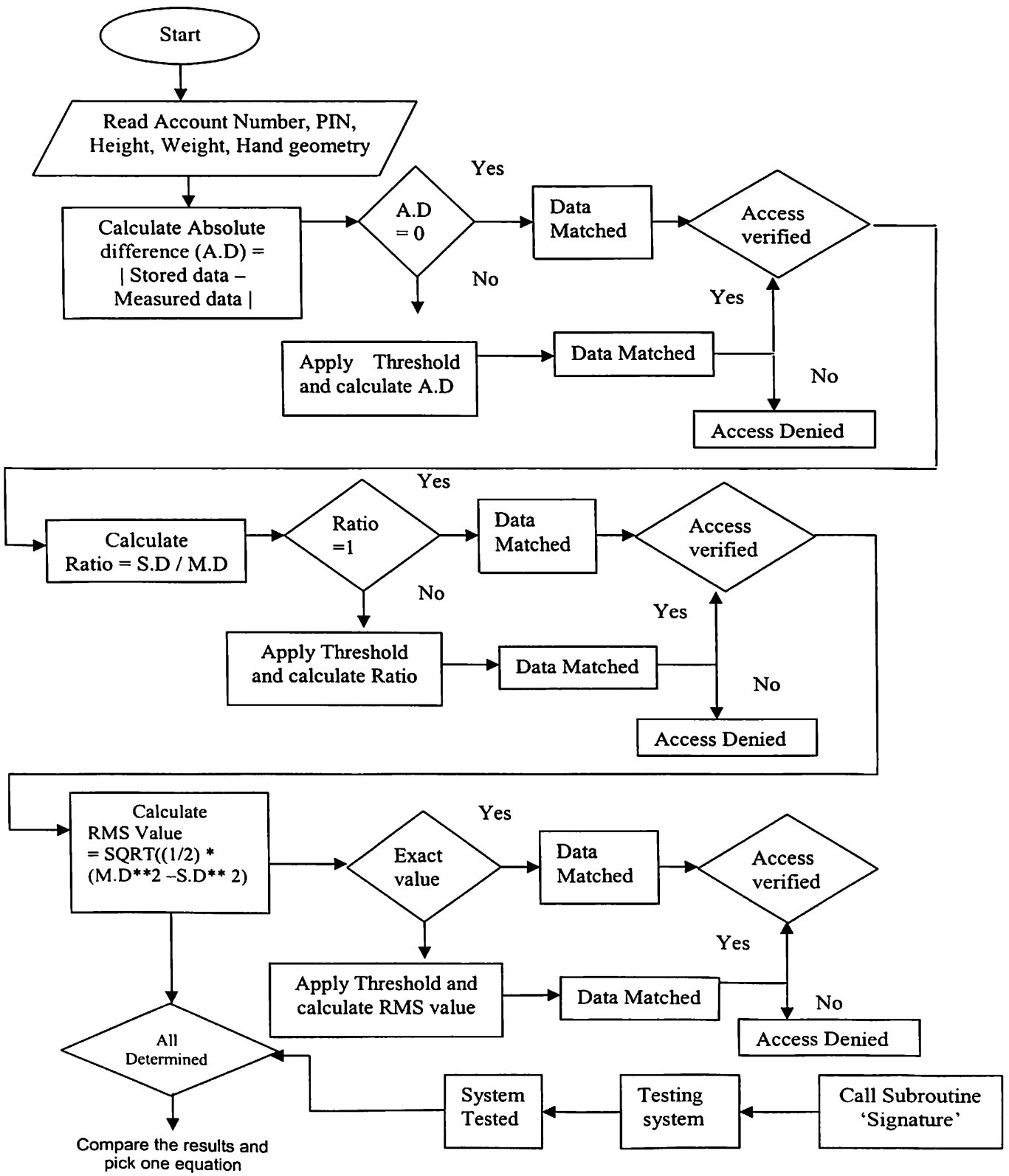


Figure 5: Flow Chart for the Fortran Programming and Calculations

As mentioned in the thesis proposal three equations have been used first in the program and from these three only one equation has been picked. Rapid system prototyping are of two types, one is throw-away approach and the other one is evolutionary approach. In this thesis I have used the throwaway approach, two equations are discarded and only one equation is picked based on the results. The percentage of correct values, the percentage of incorrect acceptance and the percentage of incorrect rejection (which are shown in Tables III, IV and V, Appendix B) have been calculated. For the equations different threshold values were applied in calculating these values, and then they are compared.

Analysis Of The Results

The equations mentioned in the above sections are used to calculate the matching rates. The software prototype is used to evaluate these equations and to select the one that gives the best result as measured by the match rates. Three sets of input database are used and a brief description of the data base are given here. Input data set 1 uses the exact data for the database. The stored data and this data set are the same. With this data set the percentage of correct data and percentage of incorrect match are calculated. In data set 2 only the last entries are changed. The account number and the PIN are the same, only the height, weight and the Finger length L1 was changed. From this data set the match rate, false match rate, false non-match rate are calculated. In input data set 3 each entry is changed by small amount in one or more data positions. For every field the percentage of correct data, false match rate, false non-match rate are calculated.

Using all three data sets the absolute difference (A.D), the ratios, and the root mean square (RMS) values are calculated for each input. The percentage of correctly matched

data, the percentage of incorrect acceptance and percentage of incorrect rejection are calculated. For all biometric inputs different threshold values are applied and all those are calculated again. The results with three sets of database will be discussed in this section and the calculations are shown in appendix B.

Using input dataset 1 to calculate the absolute difference for height the percentage of correct data is 98.23%, the incorrect acceptance is 1.77% and the incorrect rejection is 0%. Applying threshold value these values have been calculated again. Using the ratios for height the percentage of correct data is 98.23%, the percentage of incorrect acceptance 1.77% and percentage of incorrect rejection is 0%. Using the RMS values the percentage of correct data is 98.23%, the percentage of incorrect acceptance 1.77 and % of incorrect rejection is 0%. Again threshold is applied and values are calculated for both the ratios and RMS values. With all these values it has been found that applying threshold values the absolute difference gives the best result than the ratios or RMS values.

For weight calculation, absolute difference gives 100% correct data and the percentage of incorrect acceptance and the percentage of incorrect rejection is 0%. Applying threshold values in all three equations I have got 93.78% correct data, 6.22% incorrect acceptance and 0% incorrect rejection using absolute difference, 92% correct data, 8% incorrect acceptance and 0% incorrect rejection using ratios and 89.34% correct data 10.66% incorrect acceptance and 0% incorrect rejection using RMS values. The absolute difference gives better results than the other two equations used.

For finger length L1 to L5 using these three equations and applying thresholds the percentage of correct data, the percentage of incorrect acceptance, the percentage of incorrect rejection have been calculated. It has been found that for all finger lengths,

applying threshold the absolute difference gives the better results than the other two equations.

In input dataset 2, a slight change has been made in the some of the data. Height, weight and finger length 1 have been changed. Finger length2 to finger length5 are the same as in the dataset1. For height using absolute difference the percentage of correct data is 98.23%, percentage of incorrect acceptance 1.33% and percentage of incorrect rejection is 0.44%. Applying threshold the percentage of correct data is 89.33%, percentage of incorrect acceptance 10.67% and percentage of incorrect rejection is 0%. Using ratios the percentage of correct data is 98.23%, percentage of incorrect acceptance 1.33% and percentage of incorrect rejection is 0.44% and applying threshold the percentage of correct data is 73.33%, percentage of incorrect acceptance 26.67% and percentage of incorrect rejection is 0%. Using RMS values the percentage of correct data is 98.23%, percentage of incorrect acceptance 1.33% and percentage of incorrect rejection is 0.44% and applying threshold values percentage of correct data is 70.67%, percentage of incorrect acceptance 29.33% and percentage of incorrect rejection is 0%.

With all these values calculated it has been found that applying threshold values the absolute difference gives the better result than the ratios or RMS values for height with input dataset 2.

For weight using absolute difference the percentage of correct data is 99.12%, percentage of incorrect acceptance 0.44% and percentage of incorrect rejection is 0.44%. Applying threshold the percentage of correct data is 94.67%, percentage of incorrect acceptance 5.33% and percentage of incorrect rejection is 0%. Using ratios the percentage of correct data is 99.12%, percentage of incorrect acceptance 0.44% and percentage of incorrect

rejection is 0.44% and applying threshold the percentage of correct data is 91.11%, percentage of incorrect acceptance 8.89 and percentage of incorrect rejection is 0%. Using RMS values the percentage of correct data is 99.56%, percentage of incorrect acceptance 0% and percentage of incorrect rejection is 0.44% and applying threshold values percentage of correct data is 89.33%, percentage of incorrect acceptance 10.67% and percentage of incorrect rejection is 0%.

It has been found that applying threshold values the absolute difference gives the better result than the ratios or RMS values for weight with dataset2.

For finger length1 using absolute difference the percentage of correct data is 95.56%, percentage of incorrect acceptance 4% and percentage of incorrect rejection is 0.44%. Applying threshold the percentage of correct data is 96%, percentage of incorrect acceptance 4% and percentage of incorrect rejection is 0%. Using ratios the percentage of correct data is 98.23%, percentage of incorrect acceptance 1.33% and percentage of incorrect rejection is 0.44% and applying threshold the percentage of correct data is 96%, percentage of incorrect acceptance 4% and percentage of incorrect rejection is 0%. Using RMS values the percentage of correct data is 93.78%, percentage of incorrect acceptance 5.78% and percentage of incorrect rejection is 0.44% and applying threshold values percentage of correct data is 60.89, percentage of incorrect acceptance 39.11 and percentage of incorrect rejection is 0%.

It has been found that applying threshold values the absolute difference gives the better result than the ratios or RMS values for finger length1 with input dataset 2.

For input dataset 3 each entry has been changed by small amount in one or more data positions. Using three equations calculation has been done and the percentage of correct

data, percentage of incorrect acceptance and percentage of incorrect rejection have been calculated. These values are all given in Table V, appendix B. with input dataset 3 it has been found that applying threshold values the absolute difference gives the better results than the ratios or the root mean square values.

The following tables summarize this discussion. Specifically Table II shows false non matched or false rejection rate are about the same for all three equations. However the false match rate is the lowest for absolute difference.

Table I: Range of FMR (False Match Rate) For The Three Sets of Data

Types of Equations	Range of FMR (False Match Rate) for the three sets of data with three different equations		
	Input Dataset 1	Input Dataset 2	Input Dataset 3
Absolute difference	0% – 12.44%	0% – 12.44%	0% - 20.44%
Ratio	0%– 36.44%	0% - 36.44%	0% – 36.44%
Root Mean Square value	0% – 39.55%	0% – 39.11%	0% – 35.56%

Table II: Range of FNMR (False Non-Match Rate) For The Three Sets of Data

Types of Equations	Range of FNMR (False Non-Match Rate) for the three sets of data with three different equations		
	Input Dataset 1	Input Dataset 2	Input Dataset 3
Absolute difference	0%	0% – 0.44%	0% – 2.22%
Ratio	0%	0% – 0.44%	0% – 2.22%
Root Mean Square value	0%	0% – 0.44%	0% – 2.67%

Conclusions

For the next step to design and simulate the system the absolute difference will be used to compare the measured data with the stored data. To simulate the design the 6811 microprocessor will be used for the second prototype.

CHAPTER IV

MICROCONTROLLER PROTOTYPE

Introduction

The second step prototype for the access verification system uses the Motorola MC68HC11. This chapter describes briefly the 6811 microcomputer, then the design program and the results. The program has been written in assembly language. The TexaS (Test Execute and Simulate) simulator has been used to simulate the 6811 access verification system. A new idea for using BIST (Built-In Self-Test) as design verification has also been implemented in this second step prototyping.

Brief Description of 6811 Microprocessor

The Motorola MC68HC11 (referred to simply as 6811) microcomputer is optimized for low power consumption and high performance operation. The 6811 have many versions and the MC68HC11A8 is used for research. The 6811 version used for this prototype has 8 Kbytes of internal ROM, 512 bytes of EPROM and 256 bytes of RAM, [11]. The RAM contains the variables and stack. The EPROM contains the constants that are unique to each application instance. The ROM contains the program and fixed constants. The 6811 has five external I/O ports. For 6811, as with many microcomputers, the I/O ports are memory mapped. Memory mapped means that the software accesses an I/O port by

reading from or writing to the appropriate memory address rather than using specific I/O instructions.

The registers are high-speed storage devices that reside inside the processor. For the 6811, registers do not have specific memory address like RAM and ROM but have specific machine instructions that operate with the particular register. The 6811 uses either two separate 8 bit registers A and B or one combined 16bit accumulator RegD. RegA contains the most significant byte of register D. There are two index-registers X and Y. Register A and B contain data while register X and Y contain addresses. Assembly language instructions have four fields. The label field is optional and starts at the first column, and it is used to identify the position in the memory of the current instruction. The operation command (opcode) field specifies the microcomputer action to be performed. The operand field specifies where to find the data to be used by the instruction. The comment field is also optional, and it contains a description of the software, making the software easier to understand. The 6811 instructions set has five different addressing modes. The indexed addressing mode is useful when addressing data structures. The 16-bit register (RegX or RegY) is used as an index.

In this project I have used the TExaS simulator to run the Motorola 6811 microcomputer for the second prototype design of the access verification system. With the TExaS simulator, when the program is running, the microcomputer file shows the changes in different registers that help to figure out if there is any error. The I/O device file shows the port connections to the output LED's, output LCD, I/O CRT and input switches.

Design Description

In the access verification system, rapid system prototyping (RSP) is included in order to get early feedback so that it can be modified rapidly and accurately in the development process. Based upon experiments with the Fortran program, I have decided that for the dataset (both measured data and the stored data) the absolute difference will be used because it has the best FMR (False Match Rate) results. Inputs (which are the measured data) are verified with the stored data and when the data matched the access is verified. For the account number and the PIN the data should match exactly. That is why the threshold value for these two tests is zero. But for the other inputs the threshold is defined so that one can get access with a very small variation in his/her measured data when compared with the stored data.

The flowchart for the design program in 6811 is shown in the Figure 6.

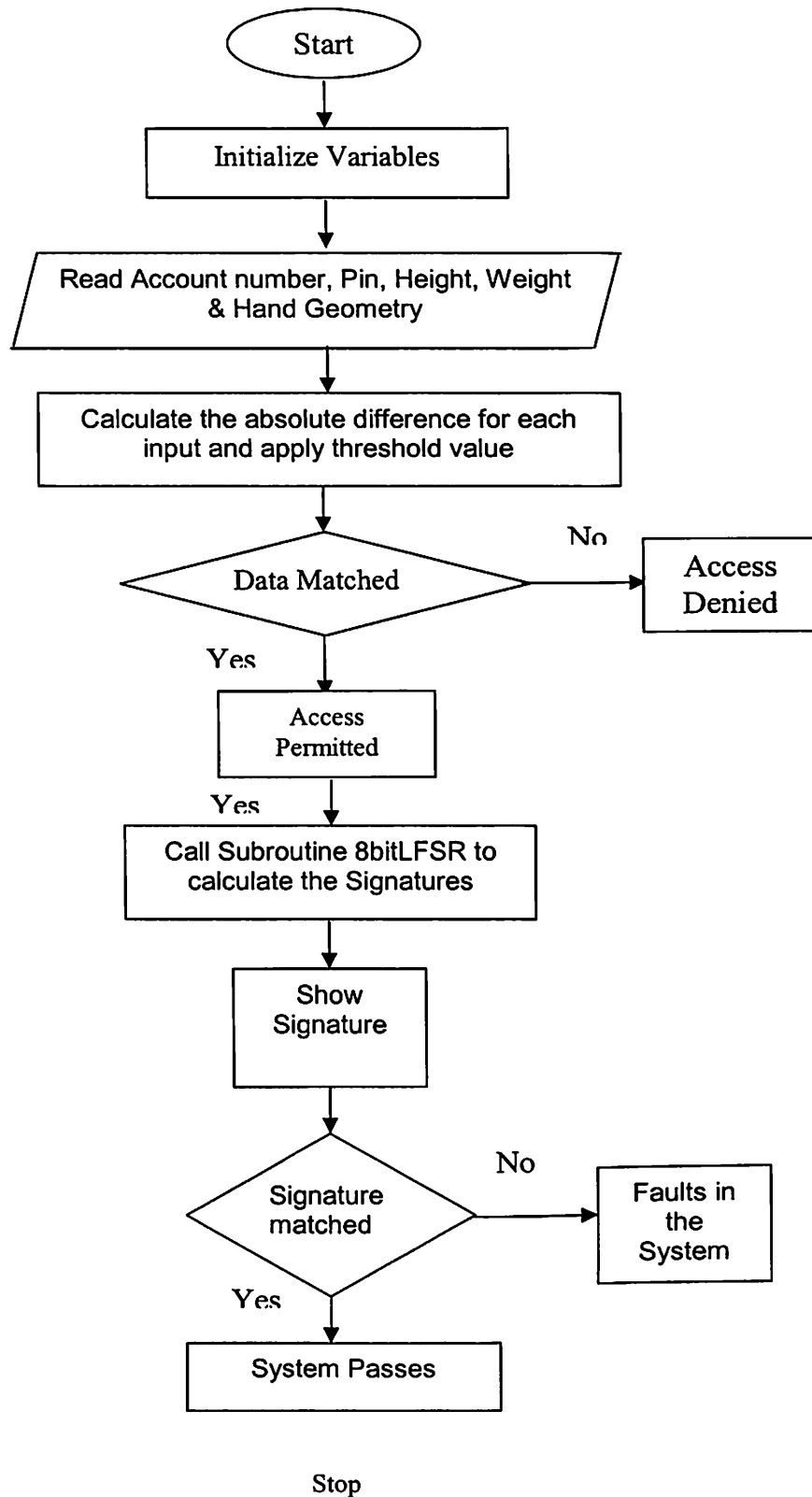


Figure 6: The Flow Chart for the 6811 Program

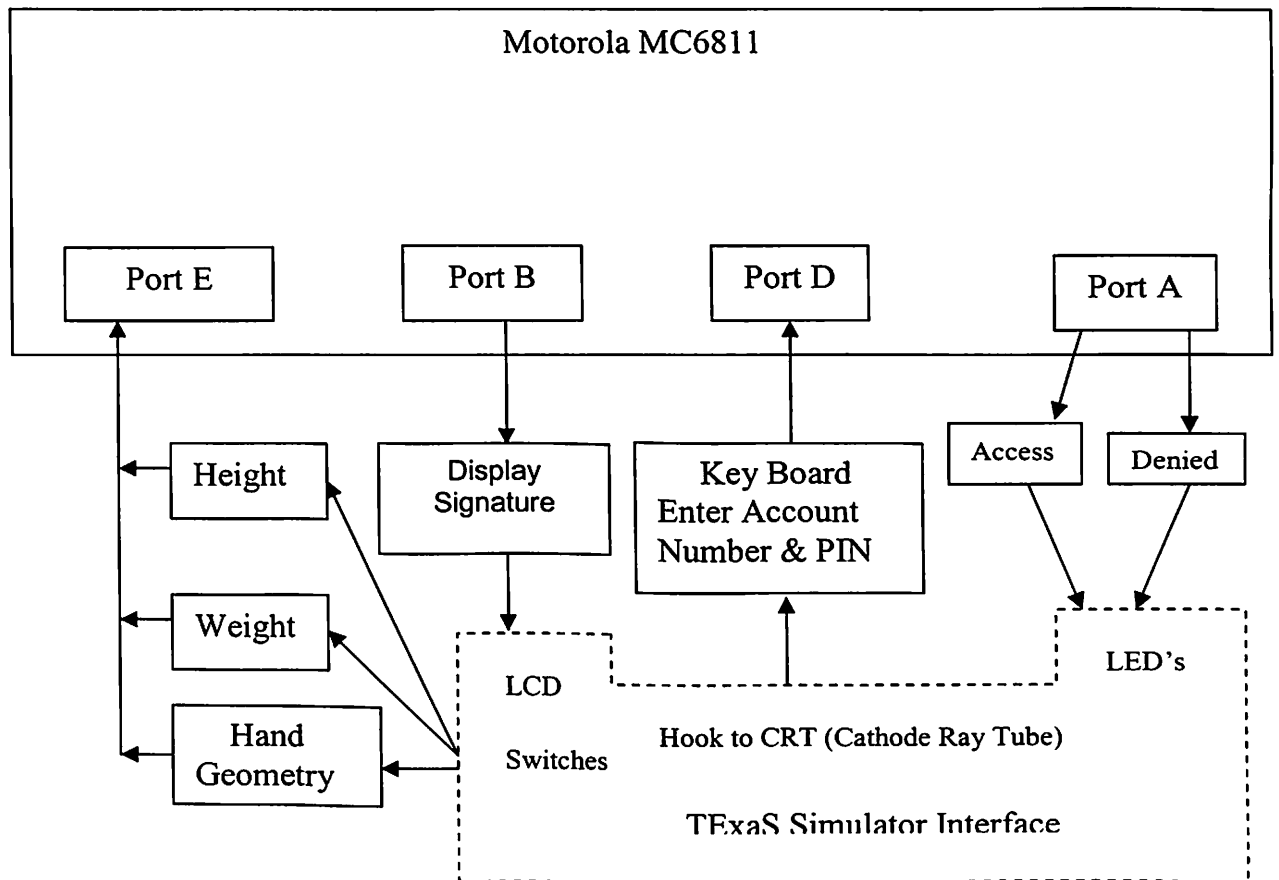


Figure 7: Block Diagram for the simulation with Motorola MC6811

The ports are used to input the values and to output the results. The ports of 6811 have input, output and bi-directional pins. The block diagram shows the input/output connections. Some ports are bi-directional but specific pins are use for input and/or for output. Port E is used for input values. Height, weight and hand geometry are entered to this port. Port D is connected to the keyboard where the account number and pin are entered. Port A shows whether the data matched or not, if matched it gives access to the

system and if not it denies access. The signature bits are calculated and are given as the output to the LCD's.

In the 6811 microprocessor system with memory mapped I/O, although the I/O devices are connected to the processor the 6811 instructions access the I/O ports similar to memory. I/O devices are assigned addresses, and the software accesses I/O using read and writes to specific I/O address. The software gets data from an input device by using the same instructions as it would if it were reading from memory. Similarly the software drives data to an output device by using the same instructions as it would if it were writing to memory. The memory map is shown in the Figure 8.

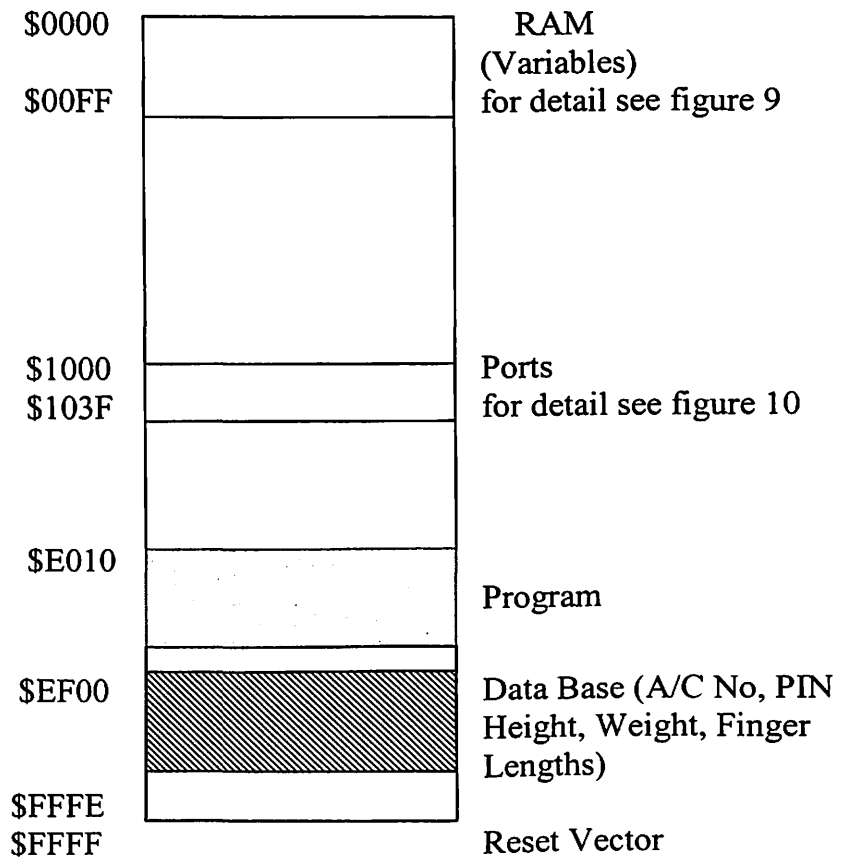
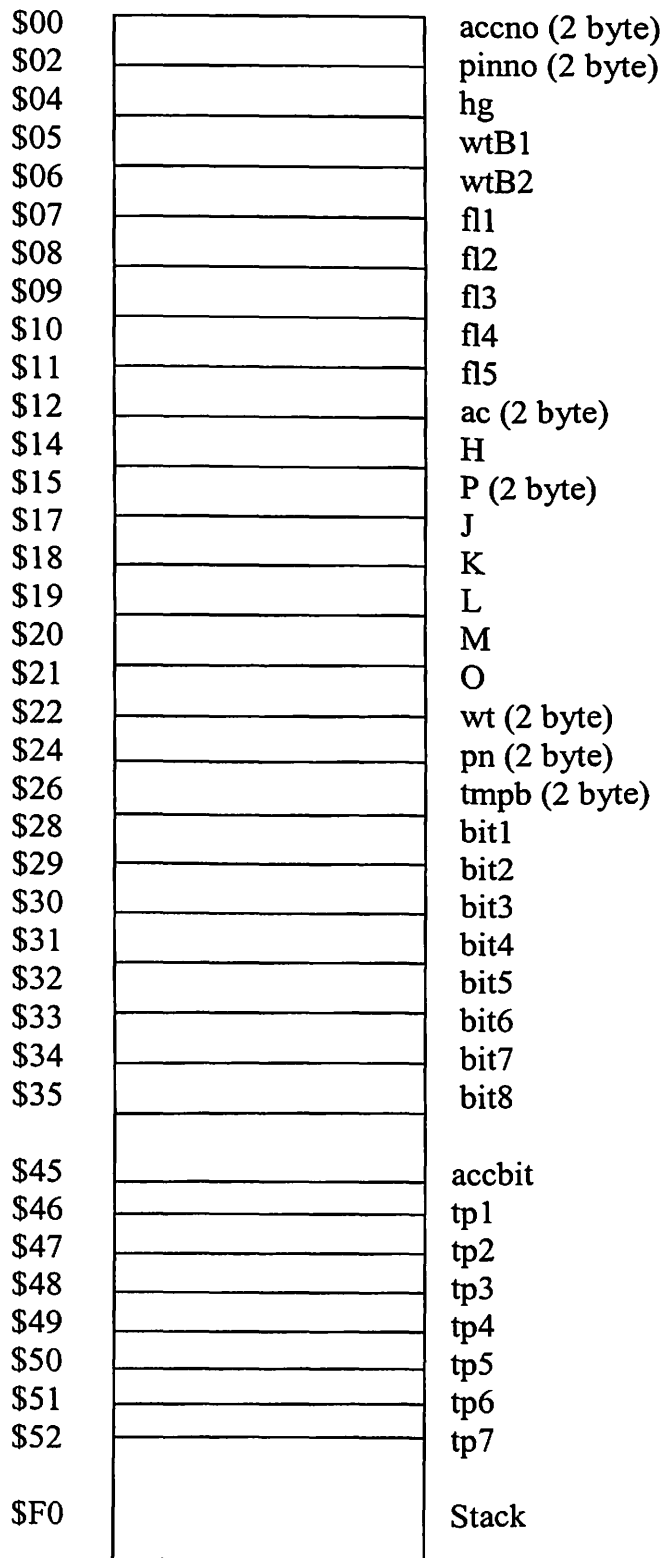


Figure 8: Memory Map For 6811

As shown in Figure 8 for a 6811 the address map for the devices are, for RAM address are from \$0000 to \$00FF, for I/O \$1000 to \$103F and for ROM \$E000 to \$EFFF.

The RAM (address from \$0000 to \$00FF), which contains the variables and stack, is shown in Figure 9. The variables specified are described in the assembly code. In general the input values are stored in accno, pinno, hg, wtB1, wtB2, fl1, fl2, fl3, fl4, fl5. The differences are stored in H, P, J, K, L, M, O. The one bit samples for signatures are stored in bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8. The internal test-points for the BIST are accbit, tp1, tp2, tp3, tp4, tp5, tp6, tp7. The stack pointer is started from \$F0.



\$FF

Figure 9: Memory Map for the Variables in RAM

The I/O ports, addresses from \$1000 to \$103F are shown in the Figure10.

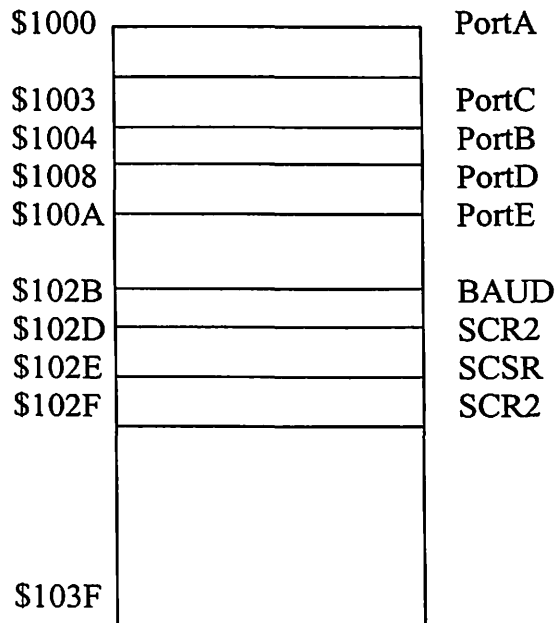


Figure 10: I/O Ports of 6811 (From \$1000 to \$103F)

Results

I have used the 6811 microprocessor to demonstrate my program accesses and verifies that it is working properly. As shown in the flowchart, in the program the variables are initialized and then the account number and PIN are entered from the keyboard, which is connected to Port D. The values of the height weight and the finger lengths are entered through port E. After entering the values they are compared with the database values and the differences are calculated. The threshold is checked for height, weight and the hand geometry and when these values are matched with the database values then the program permits access to the system otherwise it denies.

The program listing is in appendix C.

To carry test from the first step through this second step the Built-in Self-Test (BIST) technique is applied. In the 6811 the access verification program, the system can test itself and check whether it is fault free or faulty. Signature analysis technique is used in this program to test that. An 8bit LFSR is used to calculate the signature. The LFSR is implemented in an assembly language subroutine. One bit is taken from each input values (PIN, height, weight, and five finger lengths) and also from the database values to calculate the signature, when the signature matches it shows that the system is working properly and there is no fault in the system.

A surprising additional benefit of this LFSR approaches has been discovered in one of the test procedure where the signature of the entire database was calculated person by person. Error in the signature helped immediately identify error in the database. This demonstrates that the manufacturer functional test procedure of BIST with signature analysis is also useful for checking database integrity.

Values for both authorized and unauthorized cases are used as the input values in the program. It has been found that for the authorized individuals' values the system compares the values and gives access. And for the unauthorized individuals' values it shows that the values have not matched and it denies the access.

Conclusions

The assembly language program has demonstrated that the access system works properly in TExaS simulator. The signature is also calculated to test the system whether it is fault free or not. The calculated signature has showed that the system is fault free. Our Main

goal here is to prototype the design to get feedback in short time and check that it is working properly. By using 6811 and demonstrating the program it has been found that the system is working well.

A notable benefit has been discovered that the system tests with signature is also good for checking data integrity. My next step is to design the system in verilog (the hardware description language) to show how the components of the system can be connected together. Then it will be simulated to test the design module before implementing the design to real hardware.

CHAPTER V

FINAL IMPLEMENTATION

Introduction/Objective

Verilog, the hardware description language, is popular for designing digital circuitry. In this chapter, the implementation of the design for the access verification system from simple building blocks will be described. The implementation is the final product. Structural verilog is used here to describe how to put larger designs together out of smaller modules and building blocks all the way down to using the AMI standard cell library. Before implementing the design in hardware the simulator for verilog can test the functionality and timing of each design module. The simulator is also very useful for debugging designs without making the hardware prototypes.

Design Description

The access system combining the Rapid System Prototyping (RSP) with Built-In Self-Test (BIST) is the final product. It is the actual implementation of the design prototype. The access system consists of nine inputs and two outputs showing access or denial. The signatures are also shown as outputs of the final system. One of the signatures is based upon samples of the input values to test the overall system. The other signature is based

upon internal tests points and the access output, to test the chip based upon the verilog, hardware description language. The Figure: 11 the block diagram shows the inputs and outputs of the final system.

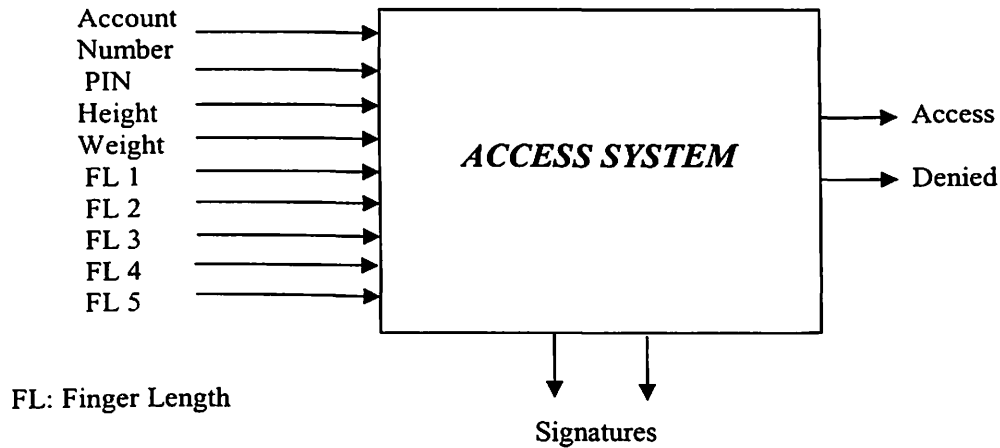
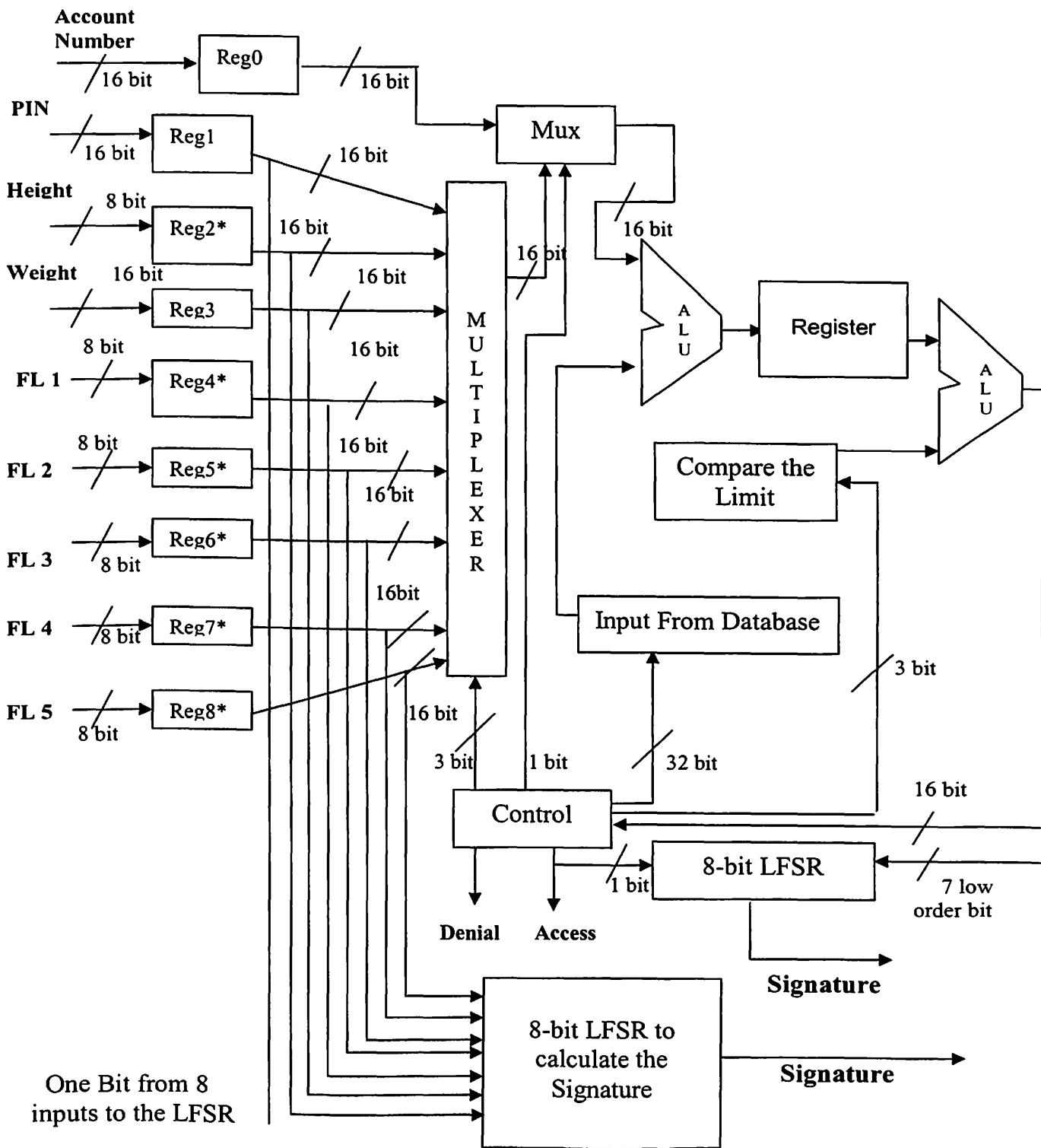


Figure 11: Block Diagram of the Access System

Figure: 12, the detail block diagram, shows the connections of the different parts of the system. The 9 inputs (account number, pin, height, weight and 5 finger lengths) are stored in registers, which are then connected to two multiplexers. The three inputs, account number, pin and the weight are 16 bits and the other six inputs are 8 bits and they are padded with 0's to get a 16 bit values from the register.

One of the multiplexer has 8 inputs and 1 output with 3 bits of control, and the other multiplexer has two inputs and one output and one bit of control. One input is used at a time. The ALU (arithmetic and logic unit) does the arithmetic operation. ALU has 2 inputs, which are 16 bits and one output of 16 bits. The measured values and the database values are the two inputs of the ALU. The output is the difference of these two values.



* 8 bit inputs are padded with "0" to get 16 bit value.

Figure 12: Detail Block Diagram of Access System

Module control is in the testbench file. Another multiplexer is used to compare the limit. 8 threshold values (for PIN, height, weigh and hand geometry) are used for the differences that are obtained from the first ALU. The second ALU is used to apply the threshold values to get the result as output.

Two LFSR's are used to test the system. The system external to the access chip is tested with one LFSR that monitors the input. The other LFSR used to test the access chip is tested by monitoring outputs.

From 8 inputs (PIN, height, weigh and five finger lengths) stored in the register, one bit is taken from each input and are used as input for an 8bit LFSR to calculate the signature. This 8bit LFSR is used here as a part of Built- In Self-Test (BIST). In designing the access verification system the BIST is also a part of the system, which will test the system itself by calculating the signatures and showing that it is working fault free. Another 8bit LFSR uses 7 low order bits from the results of the second ALU and 1 bit from the access control output. This LFSR tests the access chip by monitoring the outputs.

Conclusions

The module has been tested with a testbench file. With this testbench file all the connections of the design module are tested and the signatures are calculated. It has been demonstrated that this system implementation successfully solves the problem of improved access verification.

CHAPTER VI

CONCLUSIONS

Conclusions

In this thesis a new design methodology was proposed and demonstrated. Rapid System Prototyping (RSP) in the development process permitted early feedback for update requirements. Prototyping is fast and incorporating BIST in the design prototype ensures the quality of the final design without slowing down the design process. In this project nine inputs (account number, PIN, height, weight and five finger lengths) were used to provide secure identity verification of the system.

One of the advantages of using BIST is it reduces the cost of testing when compared with an external test using automatic test equipment. In this project BIST with signature analysis, helped check the database integrity. The surprise side benefit during prototyping was that an error in signature helped identify the error in the database.

For my access verification system, software prototyping the design (in Fortran) with the throw-away approach was used to evaluate and compare access design decisions. Based on those results, one equation was picked and used in second prototyping stage with 6811 microprocessor.

Then the system was designed in verilog and it was successfully simulated with a test bench file. In all these steps, the signatures (both whole system and chip) were calculated as a part of the BIST.

This thesis demonstrates a process to design a high quality new product by using BIST in prototyping phase. It links testing with security as well as testing with rapid system prototyping.

Recommendations

Future research is recommended for comparing other biometric measures. Some physical measurements variation should be solved over a longer time period. For example the weight would vary more over some years. So an update of the biometric database would be required. Also the system could be submitted to the MSIS (Management Science and Information Systems) department of OSU, as a project to evaluate its security.

REFERENCES

- [1]. Abramovici, Breuer and Friedman, "Digital Systems Testing and Testable Design", Computer Science Press, 1990.

- [2]. Acken J. M, "Deriving Accurate Fault Models", September 1988.

- [3]. Bardell P.H., McAnney W.H., Savir J, "Built-In Test for VLSI: Pseudorandom techniques", John Wiley& Sons, New York, 1987.

- [4]. Booth C.J. and Kurpis G.P., The New IEEE Standard Dictionary of Electrical and Electronics Terms [Including Abstracts of All Current IEEE Standards]. Fifth Edition, New York: IEEE, 1993.

- [5]. Frohwerk, "Signature Analysis: A New Digital Field Service Method", Hewlett-Packard J. May 1977.

- [6]. Gulati R.K and Hawkins C.F, "IDDQ Testing of VLSI Circuits", Kluwer Academic Publishers, 1993.

- 7]. Kordon, F and Luqi, “An introduction to Rapid System Prototyping”, IEEE Transaction on software Engineering, Vol 28, September 2002.
- [8]. Masfield T, Kelly G, Chandler D, Kane J, “Biometric Product Testing Final Report”, Issue 1.0, 19th March 2001.
- [9]. McCluskey, E.J and Buelow, F “ IC Quality and test transparency”, International test conference, Proc. IEEE Sept 1988, pp. 295- 301
- [10]. McCluskey, E.J and Bozorgui-Nesbat, S. “ Design for Autonomous Test”, Proceeding of International Test Conference, 1980
- [11]. Valvano Jonathon W, “ Embedded Microcomputer System”, Brooks/Cole Thomson Learning, 2000.

Appendices

Appendix A

Fortran Source Code

PROGRAM Access

```
INTEGER accn(20)
INTEGER PIN(15), pn(15)
INTEGER ACCOUNTNUMBER, ac(15)
REAL HEIGHT(15), WEIGHT(15), hg(15), wg(15), L1(15)
REAL L2(15), L3(15), L4(15), L5(15)
REAL fl1(15), fl2(15), fl3(15), fl4(15), fl5(15)
INTEGER ADPIN, ADAC, VAC1, VAC2, VAC3
REAL VAC, RMSAC
REAL ADHG, ADWG, ADL1, ADL2, ADL3, ADL4, ADL5
REAL RAC, RPIN, RHG, RWG, RFL1, RFL2, RFL3, RFL4, RFL5
INTEGER VP1, VP2, VP3
REAL RMSPN, VP
REAL VHG, VHG1, VHG2, VHG3, RMSHG
REAL VWG, VWG1, VWG2, VWG3, RMSWG
REAL VL, VL1, VL2, VL3, RMSL1
REAL V2L1, V2L2, V2L3, V2L, RMSL2
REAL V3L1, V3L2, V3L3, V3L, RMSL3
REAL V4L1, V4L2, V4L3, V4L, RMSL4
REAL V5L1, V5L2, V5L3, V5L, RMSL5
```

C OPEN DATA FILE

```
OPEN(UNIT=10, FILE='DATA1.TXT', STATUS='OLD')
OPEN(UNIT=20, FILE='SAMPLE1.TXT', STATUS='OLD')
```

C WRITTING THE OUTPUT FILES

```
OPEN(UNIT=30, FILE='OP.TXT', STATUS='UNKNOWN')
OPEN(UNIT=31, FILE='OP1.TXT', STATUS='UNKNOWN')
OPEN(UNIT=32, FILE='OP2.TXT', STATUS='UNKNOWN')
OPEN(UNIT=33, FILE='OP3.TXT', STATUS='UNKNOWN')
OPEN(UNIT=34, FILE='OP4.TXT', STATUS='UNKNOWN')
OPEN(UNIT=35, FILE='OP5.TXT', STATUS='UNKNOWN')
OPEN(UNIT=36, FILE='OP6.TXT', STATUS='UNKNOWN')
OPEN(UNIT=37, FILE='OP7.TXT', STATUS='UNKNOWN')
OPEN(UNIT=38, FILE='OP8.TXT', STATUS='UNKNOWN')
```

c WRITTING THE RATIO'S IN FILE

```
OPEN(UNIT=40, FILE='RATIO.TXT', STATUS='UNKNOWN')
OPEN(UNIT=41, FILE='RATIO1.TXT', STATUS='UNKNOWN')
OPEN(UNIT=42, FILE='RATIO2.TXT', STATUS='UNKNOWN')
```

```

OPEN(UNIT=43, FILE='RATIO3.TXT', STATUS='UNKNOWN')
OPEN(UNIT=44, FILE='RATIO4.TXT', STATUS='UNKNOWN')
OPEN(UNIT=45, FILE='RATIO5.TXT', STATUS='UNKNOWN')
OPEN(UNIT=46, FILE='RATIO6.TXT', STATUS='UNKNOWN')
OPEN(UNIT=47, FILE='RATIO7.TXT', STATUS='UNKNOWN')
OPEN(UNIT=48, FILE='RATIO8.TXT', STATUS='UNKNOWN')
OPEN(UNIT=49, FILE='Output.TXT', STATUS='UNKNOWN')
OPEN(UNIT=50, FILE='Output1.TXT', STATUS='UNKNOWN')
OPEN(UNIT=60, FILE='RMS-AC.TXT', STATUS='UNKNOWN')
OPEN(UNIT=61, FILE='RMS-HG.TXT', STATUS='UNKNOWN')
OPEN(UNIT=62, FILE='RMS-WG.TXT', STATUS='UNKNOWN')
OPEN(UNIT=63, FILE='RMS-L1.TXT', STATUS='UNKNOWN')
OPEN(UNIT=64, FILE='RMS-L2.TXT', STATUS='UNKNOWN')
OPEN(UNIT=65, FILE='RMS-L3.TXT', STATUS='UNKNOWN')
OPEN(UNIT=66, FILE='RMS-L4.TXT', STATUS='UNKNOWN')
OPEN(UNIT=67, FILE='RMS-L5.TXT', STATUS='UNKNOWN')
OPEN(UNIT=68, FILE='RMS-PN.TXT', STATUS='UNKNOWN')

```

```

OPEN(UNIT=70, FILE='Output2.TXT', STATUS='UNKNOWN')

```

```

DO 77 i = 1, 15

```

```

    read (10,350) accn(i),PIN(i), HEIGHT(i), WEIGHT(i),
    . L1(i), L2(i), L3(i), L4(i), L5(i)
350   format (i5, 3x,i4, 4x, f3.0, 3x, f4.0, 4X, f4.2,
    . 5x, f4.2, 4x, f4.2, 4x, f4.2, 4x, f4.2)
    c   write (6,550) accn(i),PIN(i), HEIGHT(i), WEIGHT(i),
    c   . L1(i), L2(i), L3(i), L4(i), L5(i)
c550   format (i5, 3x,i4, 4x, f3.0, 3x, f4.0, 4X, f4.2,
    c   . 5x, f4.2, 4x, f4.2, 4x, f4.2, 4x, f4.2)

```

```

    REWIND 20

```

```

DO 55 j = 1, 15

```

```

    read (20,450) ac(j),pn(j), hg(j), wg(j),
    . fl1(j), fl2(j), fl3(j), fl4(j), fl5(j)
450   format (i5, 3x,i4, 4x, f3.0, 3x, f4.0, 4X, f4.2,
    . 5x, f4.2, 4x, f4.2, 4x, f4.2, 4x, f4.2)
    c   write(6,650) ac(j),pn(j), hg(j), wg(j),
    c   . fl1(j), fl2(j), fl3(j), fl4(j), fl5(j)
c650   format (i5, 3x,i4, 4x, f3.0, 3x, f4.0, 4X, f4.2,
    c   . 5x, f4.2, 4x, f4.2, 4x, f4.2, 4x, f4.2)

```

```

    if (accn(i) .eq. ac(j)) write (6,987) accn(i), ac(j),
    . PIN(i), HEIGHT(i), WEIGHT(i),L1(i), L2(i), L3(i),
    . L4(i), L5(i)
987   format(i5, 2x, i5, 2x,i4, 2x, f3.0,2x,f4.0,2x,f4.2,2x,
    . f4.2, 2x,f4.2, 2x,f4.2, 2x,f4.2 )

```

```

C   Calculating the absolute difference

```

```

    ADAC = ABS(accn(i)-ac(j))
    write(30,12) " Absolute Difference in A/C = ",
    . ADAC

```

```

12  FORMAT(A31, 2x, i5)

      ADPIN = ABS(PIN(i)-pn(j))
      write(31,13) " Absolute Difference in PIN = ",
      ADPIN
13  FORMAT(A31, 1x, I4)

      ADHG = ABS(HEIGHT(i) - hg(j))
      write(32,14) " Absolute Difference in HEIGHT = ",
      ADHG
14  FORMAT(A34, 1x, F3.0)

      ADWG = ABS(WEIGHT(i) - wg(j))
      write(33,15) " Absolute Difference in WEIGHT = ",
      ADWG
15  FORMAT(A34, 1x, F4.0)

C    Finger 1
      ADL1 = ABS(L1(i)- fl1(j))
      write(34,16) "Absolute Difference in FingerLength1 = ",
      ADL1

C    Finger 2
      ADL2 = ABS(L2(i)- fl2(j))
      write(35,16) "Absolute Difference in FingerLength2 = ",
      ADL2

C    Finger 3
      ADL3 = ABS(L3(i)- fl3(j))
      write(36,16) "Absolute Difference in FingerLength3 = ",
      ADL3

C    Finger 4
      ADL4 = ABS(L4(i)- fl4(j))
      write(37,16) "Absolute Difference in FingerLength4 = ",
      ADL4

C    Finger 5
      ADL5 = ABS(L5(i)- fl5(j))
      write(38,16) "Absolute Difference in FingerLength5 = ",
      ADL5

16  FORMAT(A40, 1x, f4.2)

      write(49,90) "ADAC =", ADAC, "ADPIN =", ADPIN,
      "ADHG =", ADHG, "ADWG =", ADWG,"ADL1 =", ADL1,
      "ADL2 =", ADL2,"ADL3 =", ADL3,"ADL4 =", ADL4,
      "ADL5 =", ADL5
90  format(A7, 1x, i5, 2x, A8, 1x, i4, 2x, A7, 1x, f3.0,
      2x, A7, 1x, f4.0, 2x, A7, 1x,f4.2, 2x, A7, 1x, f4.2,
      2x, A7, 1x,f4.2, 2x, A7, 1x, f4.2, 2x, A7, 1x, f4.2)

C    CALCULATING THE RATIO'S

      RAC = accn(i)/ac(j)
      write(40,17) " Ratio in ACCOUNTNUMBER = ",

```

```

. RAC
17  FORMAT(A28, 1x, f9.7)

    RPIN = PIN(i)/pn(j)
    write(41,18) " Ratio in PIN = ",
. RPIN
18  FORMAT(A18, 1x, f7.5)

    RHG = HEIGHT(i)/hg(j)
    write(42,19) " Ratio in Height = ",
. RHG
19  FORMAT(A21, 1x, F4.2)

    RWG = WEIGHT(i)/wg(j)
    write(43,20) " Ratio in Weight = ",
. RWG
20  FORMAT(A21, 1x, F4.2)

    RFL1 = L1(i)/fl1(j)
    write(44,21) " Ratio in FL1 = ",
. RFL1

    RFL2 = L2(i)/fl2(j)
    write(45,21) " Ratio in FL2 = ",
. RFL2

    RFL3 = L3(i)/fl3(j)
    write(46,21) " Ratio in FL3 = ",
. RFL3

    RFL4 = L4(i)/fl4(j)
    write(47,21) " Ratio in FL4 = ",
. RFL4

    RFL5 = L5(i)/fl5(j)
    write(48,21) " Ratio in FL5 = ",
. RFL5

21  FORMAT(A18, 1x, f4.2)

    write (50,91)"RAC =", RAC, "RPIN =", RPIN, "RHG =",
. RHG, "RWG =", RWG, "RFL1 =", RFL1, "RFL2 =", RFL2,
. "RFL3 =", RFL3, "RFL4 =", RFL4, "RFL5 =", RFL5
91  Format(A7,1x,f9.7, 2x,A8,1x,f7.5, 2x,A7,1x,f4.2,
. 2x,A7,1x,f4.2, 2x,A7,1x,f4.2, 2x,A7,1x,f4.2,
. 2x,A7,1x,f4.2, 2x,A7,1x,f4.2, 2x,A7,1x,f4.2)

C   CALCULATING THE RMS VALUES FOR EACH FIELD
C   RMS value for Account number

    VAC1 = accn(i)**2
    VAC2 = accn(j)**2
    VAC3 = (VAC1 + VAC2)
    VAC = VAC3/2
    RMSAC = SQRT(VAC)
    WRITE(60,24)RMSAC

```



```

24  FORMAT(f8.2)

C  RMS value for PIN

VP1 = PIN(i)**2
VP2 = pn(j)**2
VP3 = VP1 + VP2
VP = VP3/2
RMSPN = SQRT(VP)
WRITE (68,28) PIN(i), pn(j), RMSPN
28  FORMAT(i4, 2x, i4, 2x, f7.2)

C  RMS value for HEIGHT

VHG1= HEIGHT(i)**2
VHG2= hg(j)**2
VHG3 = (VHG1+ VHG2)
VHG = VHG3/2
RMSHG = SQRT(VHG)
WRITE(61,25)HEIGHT(i), hg(j), RMSHG
25  FORMAT(f3.0, 2x, f3.0, 2x, f5.2)

C  RMS value for WEIGHT

VWG1= WEIGHT(i)**2
VWG2= wg(j)**2
VWG3 = (VWG1+ VWG2)
VWG = VWG3/2
RMSWG = SQRT(vWG)
WRITE(62,26)WEIGHT(i), wg(j), RMSWG
26  FORMAT(f4.0, 2x, f4.0, 2x, f6.2)

C  RMS values for FINGERS

VL1= L1(i)**2
VL2= fl1(j)**2
VL3 = (VL1+ VL2)
VL = VL3/2
RMSL1 = SQRT(VL)
WRITE(63,27)"L1(i) =", L1(i),"L1(j) =",fl1(j),
"RMSL1 =", RMSL1

V2L1= L2(i)**2
V2L2= fl2(j)**2
V2L3 = (V2L1+ V2L2)
V2L = V2L3/2
RMSL2 = SQRT(V2L)
WRITE(64,27)"L2(i) =", L2(i),"L2(j) =",fl2(j),
"RMSL2 =", RMSL2

V3L1= L3(i)**2
V3L2= fl3(j)**2
V3L3 = (V3L1+ V3L2)
V3L = V3L3/2
RMSL3 = SQRT(V3L)
WRITE(65,27)"L3(i) =", L3(i),"L3(j) =",fl3(j),

```

```

. "RMSL3 =", RMSL3

V4L1= L4(i)**2
V4L2= fl4(j)**2
V4L3 = (V4L1+ V4L2)
V4L = V4L3/2
RMSL4 = SQRT(V4L)
WRITE(66,27)"L4(i) =", L4(i),"L4(j) =",fl4(j),
. "RMSL4 =", RMSL4

V5L1= L5(i)**2
V5L2= fl5(j)**2
V5L3 = (V5L1+ V5L2)
V5L = V5L3/2
RMSL5 = SQRT(V5L)
WRITE(67,27)"L5(i) =", L5(i),"L5(j) =",fl5(j),
. "RMSL5 =", RMSL5

27  FORMAT(A7,1X,f4.2,2x,A7,1X,f4.2,2X,A7,1X,f4.2)

write (70,92)"RMSHG =", RMSHG, "RMSWG =", RMSWG,
. "RMSL1 =", RMSL1,"RMSL2 =", RMSL2,"RMSL3 =", RMSL3,
. "RMSL4 =", RMSL4, "RMSL5 =", RMSL5
92  Format(A7,1x,f5.2, 2x,A7,1x,f6.2, 2x,A7,1x,f4.2,
. 2x,A7,1x,f4.2, 2x,A7,1x,f4.2, 2x,A7,1x,f4.2,
. 2x,A7,1x,f4.2,f4.2)

55  continue
77  continue

STOP
END

```

Program LFSR To calculate the Signature

```

PROGRAM LFSR
  INTEGER accn(15), PIN(15)
  REAL HEIGHT(15), WEIGHT(15)
  REAL L1(15), L2(15), L3(15), L4(15), L5(15)
  REAL fl1, fl2, fl3, fl4, fl5
  INTEGER BP, BH, BW, BFL1, BFL2, BFL3, BFL4, BFL5
  INTEGER OP8, Stg1, Stg2, stg3, Stg4, Stg5
  INTEGER Stg6, Stg7, Stg8, Stage3, Stage4, stage8

  OPEN(UNIT=10, FILE='NEWDATA1.TXT', STATUS='OLD')
  OPEN(UNIT=20, FILE='STORE7.TXT', STATUS='UNKNOWN')

  REWIND 10

  DO 11 i = 1, 15
    read (10,110) accn(i),PIN(i), HEIGHT(i), WEIGHT(i),

```

```
. L1(i), L2(i), L3(i), L4(i), L5(i)
110  format (i5, 3x, i4, 4x, f3.0, 3x, f4.0, 4X, f4.2,
. 5x, f4.2, 4x, f4.2, 4x, f4.2, 4x, f4.2)
```

```
IP = PIN(i)
IP2 = IP/2
IP2 = IP2 * 2
BP = IP - IP2
WRiTE(20,120) "BP =", BP
```

```
IH = IFIX(Height(i))
IH2 = IH/2
IH2 = IH2 * 2
BH = IH- IH2
WRiTE(20,120) "BH =", BH
```

```
IW = IFIX(Weight(i))
IW2 = IW/2
IW2 = IW2 * 2
BW = IW- IW2
WRiTE(20,120) "BW =", BW
```

```
IFL1 = IFIX(L1(i))
IFL12 = IFL1/2
IFL12 = IFL12 * 2
BFL1 = IFL1- IFL12
WRiTE(20,120) "BL1 =", BFL1
```

```
IFL2 = IFIX(L2(i))
IFL22 = IFL2/2
IFL22 = IFL22 * 2
BFL2 = IFL2- IFL22
WRiTE(20,120) "BL2 =", BFL2
```

```
IFL3 = IFIX(L3(i))
IFL32 = IFL3/2
IFL32 = IFL32 * 2
BFL3 = IFL3- IFL32
WRiTE(20,120) "BL3 =", BFL3
```

```
IFL4 = IFIX(L4(i))
IFL42 = IFL4/2
IFL42 = IFL42 * 2
BFL4 = IFL4- IFL42
WRiTE(20,120) "BL4 =", BFL4
```

```
IFL5 = IFIX(L5(i))
IFL52 = IFL5/2
IFL52 = IFL52 * 2
BFL5 = IFL5- IFL52
WRiTE(20,120) "BL5 =", BFL5
```

```
120  FORMAT(A5, 1X, I3)
```

```

C    Using 8-bit LFSR for the Signature

c    Stage1 = InStage1 XOR oldStage8
c    Stage1 = BP XOR oldStage8

    OldStage8 = Stg8
    OP8 = OldStage8

    If (BP .EQ. OP8) Stg1 = 0
    If (BP .NE. OP8) Stg1 = 1
    write(20,130) "Stage 1=", Stg1

c    Stage2 = InStage2 XOR Stage1
c    Instage 2 = BH

    If (BH .EQ. Stg1) Stg2 = 0
    If (BH .NE. Stg1) Stg2 = 1
    write(20,130) "Stage 2=", Stg2

c    Stage3 = InStage3 XOR Stage2 XOR OldStage8
c    Instage 3 = BW

    If (BW .EQ. Stg2) Stage3 = 0
    If (BW .NE. Stg2) Stage3 = 1
c    write (6,12)stage3
c12  format(i2)
    If (Stage3 .EQ. OP8) Stg3 = 0
    If (Stage3 .NE. OP8) Stg3 = 1

    write(20,130) "Stage 3=", Stg3

c    Stage4 = InStage4 XOR Stage3 XOR OldStage8
c    Instage 4 = BFL1

    If (BFL1 .EQ. Stage3) Stage4 = 0
    If (BFL1 .NE. Stage3) Stage4 = 1
c    write (6,13)stage4
c13  format(i2)
    If (Stage4 .EQ. OP8) Stg4 = 0
    If (Stage4 .NE. OP8) Stg4 = 1
    write(20,130) "Stage 4=", Stg4

c    Stage5 = InStage5 XOR Stage4
C    Instage 5 = BFL2

    If (BFL2 .EQ. Stage4) Stg5 = 0
    If (BFL2 .NE. Stage4) Stg5 = 1
    write(20,130) "Stage 5=", Stg5

c    Stage6 = InStage6 XOR oldStage5
C    Instage 6 = BFL3

```

```

If (BFL3 .EQ. Stg5) Stg6 = 0
If (BFL3 .NE. Stg5) Stg6 = 1
write(20,130) "Stage 6=", Stg6

c   Stage7 = InStage7 XOR oldStage6
C   Instage 7 = BFL4

If (BFL4 .EQ. Stg6) Stg7 = 0
If (BFL4 .NE. Stg6) Stg7 = 1
write(20,130) "Stage 7=", Stg7

c   Stage8 = InStage8 XOR Stage7 XOR OldStage8
c   Instage 8 = BFL5

If (BFL5 .EQ. Stg7) Stage8 = 0
If (BFL5 .NE. Stg7) Stage8 = 1
c   write (6,14)stage8
c14  format(i2)
If (Stage8 .EQ. OP8) Stg8 = 0
If (Stage8 .NE. OP8) Stg8 = 1
write(20,130) "Stage 8=", Stg8

130  format(A8, 1x,i2)

11   CONTINUE

STOP
END

```

Appendix B

The Three Input Data Set

Input Data Set 1

This uses the exact data for the database. The stored database and this data set are the same. With this data set the percentage of correct data and percentage of incorrect match are calculated.

10212	1902	75	202	0.95	3.2	3.6	3.2	2.4
10213	1038	64	100	0.84	3.0	3.3	3.0	2.3
10214	8934	67	125	0.86	3.1	3.5	3.2	2.2
10215	4823	75	169	1.0	3.3	3.7	3.4	2.6
10216	4324	77	210	1.1	3.4	3.9	3.5	2.8
10217	9786	69	181	0.94	3.1	3.6	3.2	2.4
10218	4253	58	105	0.7	2.8	3.1	2.9	2.1
10219	1223	62	135	0.8	2.9	3.1	2.9	2.1
10220	4395	76	195	0.95	3.0	3.5	3.1	2.4
10221	9596	69	145	0.95	3.0	3.5	3.1	2.4
10222	2434	66	120	0.86	3.2	3.5	3.2	2.3
10223	1255	60	152	0.82	3.0	3.4	3.1	2.3
10224	3844	65	147	0.85	3.1	3.5	3.2	2.4
10225	1234	68	193	1.3	3.5	4.0	3.6	2.9
10226	6564	73	134	1.0	3.2	3.7	3.3	2.7

Input Data Set 2

In this data set only the last entry was changed. The account number and the PIN are the same, only the height, weight and the Finger length L1 was changed. From this data set the False non-match rate is calculated.

10212	1902	75	202	0.95	3.2	3.6	3.2	2.4
10213	1038	64	100	0.84	3.0	3.3	3.0	2.3
10214	8934	67	125	0.86	3.1	3.5	3.2	2.2
10215	4823	75	169	1.0	3.3	3.7	3.4	2.6
10216	4324	77	210	1.1	3.4	3.9	3.5	2.8
10217	9786	69	181	0.94	3.1	3.6	3.2	2.4
10218	4253	58	105	0.7	2.8	3.1	2.9	2.1
10219	1223	62	135	0.8	2.9	3.1	2.9	2.1
10220	4395	76	195	0.95	3.0	3.5	3.1	2.4
10221	9596	69	145	0.95	3.0	3.5	3.1	2.4
10222	2434	66	120	0.86	3.2	3.5	3.2	2.3
10223	1255	60	152	0.82	3.0	3.4	3.1	2.3
10224	3844	65	147	0.85	3.1	3.5	3.2	2.4
10225	1234	68	193	1.3	3.5	4.0	3.6	2.9
10226	6564	74	135	1.1	3.2	3.7	3.3	2.7

Input Data Set 3

In this data set each entry is changed by small amount in one or more data positions. For every field the percentage of correct data and the False Match Rate are calculated.

10212	1902	75	203	0.95	3.2	3.6	3.2	2.4
10213	1038	64	100	0.85	3.0	3.3	3.0	2.3
10214	8934	67	125	0.86	3.2	3.5	3.2	2.2
10215	4823	75	169	1.0	3.3	3.6	3.4	2.6
10216	4324	76	210	1.1	3.4	3.9	3.5	2.8
10217	9786	69	181	0.94	3.1	3.6	3.1	2.4
10218	4253	58	100	0.7	2.8	3.1	2.9	2.1
10219	1223	62	125	0.8	2.9	3.1	2.9	2.1
10220	4395	76	195	0.94	2.9	3.4	3.0	2.3
10221	9596	69	145	0.95	3.0	3.5	3.0	2.4
10222	2434	66	121	0.86	3.2	3.5	3.2	2.3
10223	1255	61	151	0.85	3.1	3.5	3.2	2.4
10224	3844	64	147	0.85	3.1	3.5	3.2	2.4
10225	1234	68	293	1.3	3.5	4.0	3.6	2.9
10226	6564	74	135	1.1	3.2	3.7	3.3	2.7

The calculations with three sets of database are shown below in the tables.

With Input Data set 1

Table III: Calculated values with input dataset 1

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Height	Absolute Difference	0	98.23	1.77	0
		0 to 1 inch	90.23	9.77	0
	Ratio	1	98.23	1.77	0
		0.95 to 1.05	72.89	27.11	0
	Root Mean Square Value	Exact Value	98.23	1.77	0
		Exact value to ± 2	70.23	29.77	0
Weight	Absolute Difference	0	100	0	0
		0 to 5 lbs	93.78	6.22	0
		0 to 10 lbs	90.67	9.33	0
	Ratio	1	100	0	0
		0.95 to 1.05	92	8	0
		1 to 1.10	92.45	7.55	0
	Root Mean Square Value	Exact Value	100	0	0
		Exact value to ± 5	89.34	10.66	0
Finger Length L1	Absolute Difference	0	95.56	4.44	0
		0 to 0.05	89.34	10.66	0
	Ratio	1	95.56	4.44	0
		0.95 to 1.05	77.78	22.22	0
	Root Mean Square Value	Exact Value	92.89	7.11	0
Exact value to ± 0.05		60.45	39.55	0	
Finger Length L2	Absolute Difference	0	89.34	10.66	0
		0 to 0.05	89.34	10.66	0
	Ratio	1	89.34	10.66	0
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	90.67	9.33	0
Exact value to ± 0.05		62.67	37.33	0	

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Finger Length L3	Absolute Difference	0	88.45	11.55	0
		0 to 0.05	88.45	11.55	0
	Ratio	1	88.45	11.55	0
		0.95 to 1.05	64	36	0
	Root Mean Square Value	Exact Value	88.89	11.11	0
Exact value to ± 0.05		69.67	30.33	0	
Finger Length L4	Absolute Difference	0	87.56	12.44	0
		0 to 0.05	87.56	12.44	0
	Ratio	1	87.56	12.44	0
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	87.56	12.44	0
		Exact value to ± 0.05	62.67	37.33	0
Finger Length L5	Absolute Difference	0	88.45	11.55	0
		0 to 0.05	88.45	11.55	0
	Ratio	1	88.45	11.55	0
		0.95 to 1.05	70.23	29.77	0
	Root Mean Square Value	Exact Value	87.56	12.44	0
		Exact value to ± 0.05	67.12	32.88	0

With Input Data set 2
 Slight change in sample data file (Height, Weight and L1)

Table IV: Calculated values with input dataset 2

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Height	Absolute Difference	0	98.23	1.33	0.44
		0 to 1 inch	89.33	10.67	0
	Ratio	1	98.23	1.33	0.44
		0.95 to 1.05	73.33	26.67	0
	Root Mean Square Value	Exact Value	98.23	1.33	0.44
		Exact value to ± 2	70.67	29.33	0
Weight	Absolute Difference	0	99.12	0.44	0.44
		0 to 5 lbs	94.67	5.33	0
		0 to 10 lbs	88	12	0
	Ratio	1	99.12	0.44	0.44
		0.95 to 1.05	91.11	8.89	0
		1 to 1.10	92.44	7.56	0
	Root Mean Square Value	Exact Value	99.56	0	0.44
		Exact value to ± 5	89.33	10.67	0
	Finger Length L1	Absolute Difference	0	95.56	4
0 to 0.05			96	4	0
Ratio		1	98.23	1.33	0.44
		0.95 to 1.05	96	4	0
Root Mean Square Value		Exact Value	93.78	5.78	0.44
	Exact value to ± 0.05	60.89	39.11	0	
Finger Length L2	Absolute Difference	0	89.34	10.66	0
		0 to 0.05	89.34	10.66	0
	Ratio	1	89.34	10.66	0
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	90.67	9.33	0
Exact value to ± 0.05		62.67	37.33	0	

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Finger Length L3	Absolute Difference	0	88.45	11.55	0
		0 to 0.05	88.45	11.55	0
	Ratio	1	88.45	11.55	0
		0.95 to 1.05	64	36	0
	Root Mean Square Value	Exact Value	88.89	11.11	0
		Exact value to ± 0.05	69.67	30.33	0
Finger Length L4	Absolute Difference	0	87.56	12.44	0
		0 to 0.05	87.56	12.44	0
	Ratio	1	87.56	12.44	0
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	87.56	12.44	0
		Exact value to ± 0.05	62.67	37.33	0
Finger Length L5	Absolute Difference	0	88.45	11.55	0
		0 to 0.05	88.45	11.55	0
	Ratio	1	88.45	11.55	0
		0.95 to 1.05	70.23	29.77	0
	Root Mean Square Value	Exact Value	87.56	12.44	0
		Exact value to ± 0.05	67.12	32.88	0

With Input Dataset 3

Table V: Calculated values with input dataset 3

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Height	Absolute Difference	0	95.56	2.66	1.78
		0 to 1 inch	88.44	11.56	0
	Ratio	1	96	2.22	1.78
		0.95 to 1.05	73.78	26.22	0
	Root Mean Square Value	Exact Value	95.56	2.66	1.78
		Exact value to ± 2	71.11	28.89	0
Weight	Absolute Difference	0	96.89	0.89	2.22
		0 to 5 lbs	95.11	4.89	0
		0 to 10 lbs	89.33	10.67	0
	Ratio	1	96.89	0.89	2.22
		0.95 to 1.05	92.44	7.56	0
	Root Mean Square Value	Exact Value	96.44	0.89	2.67
Exact value to ± 5		90.22	9.78	0	
Finger Length L1	Absolute Difference	0	95.11	4	0.89
		0 to 0.05	79.56	20.44	0
	Ratio	1	95.11	4.0	0.89
		0.95 to 1.05	81.33	18.67	0
	Root Mean Square Value	Exact Value	89.34	9.33	1.33
Exact value to ± 0.05		64.44	35.56	0	
Finger Length L2	Absolute Difference	0	90.22	8.89	0.89
		0 to 0.05	90.22	8.89	0.89
	Ratio	1	91.11	8.89	0
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	91.11	8.89	0
Exact value to ± 0.05		65.78	34.22	0	

		<i>Limit</i>	<i>% Correct</i>	<i>% Incorrect Acceptance</i>	<i>% Incorrect Rejection</i>
Finger Length L3	Absolute Difference	0	88.01	11.55	0.44
		0 to 0.05	91.11	8.89	0
	Ratio	1	88.01	11.55	0.44
		0.95 to 1.05	63.56	36.44	0
	Root Mean Square Value	Exact Value	88.44	11.56	0
		Exact value to ± 0.05	68	32	0
Finger Length L4	Absolute Difference	0	88.89	10.67	0.44
		0 to 0.05	89.34	10.66	0
	Ratio	1	89.34	10.22	0.44
		0.95 to 1.05	65.33	34.67	0
	Root Mean Square Value	Exact Value	89.34	10.22	0.44
		Exact value to ± 0.05	65.33	34.67	0
Finger Length L5	Absolute Difference	0	87.11	12.45	0.44
		0 to 0.05	87.55	12.45	0
	Ratio	1	87.11	12.45	0.44
		0.95 to 1.05	67.56	32.44	0
	Root Mean Square Value	Exact Value	88	12	0
		Exact value to ± 0.05	67.56	32.44	0

Appendix C

6811 Program Access.rtf

***** ACCESS.RTF *****

- * Simulator Program for the Access Verification System
- *
- * Nahreen Maherukh
- *
- * 10/1/03
- *
- * This Program will Simulate and show that it gives access only when the data is matched.
- *
- * It will calculate one least significant bit from each input (from database and the measured one)
- *
- * and store it, to calculate the signature.

- * Modified on 16th February
- *

- ** How to Run the Program
- * First start the program
- * Enter Account number on Serial Port (CRT). The Purple diode on PortA Pin 5 will indicate that the Account number is being input
- * Wait while the Serial port runs slow
- * Enter PIN (Personal Identification Number) on Serial Port (CRT). The Blue diode on port A pin6 will indicate that the PIN is being input
- * Wait while the Serial port runs slow
- * Enter Height
- * Enter Weight
- * Enter the Finger Lengths
- * The program compare the account number and the PIN then find the difference between the height, weight & the hand geometries (the finger lengths)
- * Calculate one least significant bit from each input (from database and the measured one) and store it.

accno	equ	0
pinno	equ	2
hg	equ	4
wtB1	equ	5
wtB2	equ	6
f11	equ	7
f12	equ	8
f13	equ	9
f14	equ	10
f15	equ	11
ac	equ	12
H	equ	14

```

P      equ    15
J      equ    17
K      equ    18
L      equ    19
M      equ    20
O      equ    21
wt     equ    22
pn     equ    24
tmpb   equ    26
bit1   equ    28
bit2   equ    29
bit3   equ    30
bit4   equ    31
bit5   equ    32
bit6   equ    33
bit7   equ    34
bit8   equ    35

PORTA  equ    $1000
PORTB  equ    $1004
PORTE  equ    $100A      Input Height, Weight and Hand Geometry
SCSR   equ    $102e      Serial status register
SCDR   equ    $102F      Serial input for Port D, Acc/No and Pin
BAUD   equ    $102b      Set serial input baud rate
SCR2   equ    $102D      Serial input control register

      org    $D000      Object code goes in ROM

*      Initialize all with the value 0
start  ldaa   #$00
      staa   BAUD        initial serial input baud rate to 125,000 bps
      ldaa   #$0C
      staa   SCR2        set serial input control register for input.
      lds    #$00f0      load stack pointer with start of stack

*      Read the Account number & PIN from Port D
again  ldaa   #$20
      staa   PORTA      Temporary indicator that account is being input
      ldx   #$0000      For a temporary 16 bit register to store A/C No
      stx   ac          Initialize ac to 0
      ldy   #$0005
inacc  ldaa   SCSR
      anda  #$20
      beq   inacc
      ldab  SCDR        Read the values from serial input pin
      andb  #$0F        convert ascii to decimal by masking top part.
      stab  tmpb        Temprary storage of next digit.

*      Warning this routine requires 5 digits for each Account number
*      Warning Account numbers cannot exceed 65534.
      ldx   ac
      xgdx                exchange contents of RegD with Regx
      lsld                16 bit logical left shift ro RegD
      lsld
      lsld
      addd  ac
      addd  ac

```

```

    xgdx          exchange contents of RegD with Regx
    ldab    tmpb
    abx
    stx    accno
    stx    ac
    dey
    cpy    #$0000
    bne    inacc

***    Read the PIN from
    ldaa    #$40
    staa    PORTA          Temporary indicator that pin is being input
    ldx    #$0000
    stx    pn
    ldy    #$0004
inpin  ldaa    SCSR
    anda    #$20
    beq    inpin
    ldab    SCDR          Read the values from serial input pin
*      Warning this routine requires 4 digits for each PIN
    andb    #$0F          convert ascii to decimal by masking top part.
    stab    tmpb
    ldx    pn
    xgdx          exchange contents of RegD with Regx
    lsl    16          16 bit logical left shift to RegD
    lsl
    lsl
    addd    pn
    addd    pn
    xgdx          exchange contents of RegD with Regx
    ldab    tmpb
    abx
    stx    pinno
    stx    pn
    dey
    cpy    #$0000
    bne    inpin

*      Read the Height, Weight and the Hand geometry from Port E

*      Read the Height
hgin   ldaa    SCSR
    anda    #$20
    beq    hgin
    ldab    SCDR
    ldaa    PORTE          Read switch values
    staa    hg

*      Read the Weight
wtb1in ldaa    SCSR
    anda    #$20
    beq    wtb1in
    ldab    SCDR
    ldaa    PORTE

```



```

        staa    wtB1
wtb2in  ldaa    SCSR
        anda   #$20
        beq    wtb2in
        ldab   SCDR
        ldaa   PORTE
        staa   wtB2

*      Read  the Hand Geometry

        ldaa   #$f1
fl1in  staa   PORTB
        ldaa   SCSR
        anda   #$20
        beq    fl1in
        ldab   SCDR
        ldaa   PORTE
        staa   fl1

        ldaa   #$f2
fl2in  staa   PORTB
        ldaa   SCSR
        anda   #$20
        beq    fl2in
        ldab   SCDR
        ldaa   PORTE
        staa   fl2

        ldaa   #$f3
fl3in  staa   PORTB
        ldaa   SCSR
        anda   #$20
        beq    fl3in
        ldab   SCDR
        ldaa   PORTE
        staa   fl3

        ldaa   #$f4
fl4in  staa   PORTB
        ldaa   SCSR
        anda   #$20
        beq    fl4in
        ldab   SCDR
        ldaa   PORTE
        staa   fl4

        ldaa   #$f5
fl5in  staa   PORTB
        ldaa   SCSR
        anda   #$20
        beq    fl5in
        ldab   SCDR
        ldaa   PORTE
        staa   fl5

```

```

*      LOOP 1 Account Number & PIN
*      Comparing the Account Number
      ldx    #$EF00
ckacc  ldd    0,x          Load in RegD = acc + [Regx]
      cpd    accno       16 bit compare RegD with memory
      beq    Found
      xgdx                   exchange contents of RegD with RegX
      stab   PORTB       store memeory from RegB
      addd   #$0C        add to RegD
      xgdx                   exchange contents of RegD with RegX
      cpx    eod         Check for end of Data Base
      blo    ckacc       branch if less than
      bra    Notfound
*      Test the limit--- Account number
Found  ldaa   #$FF
      staa  PORTB
      staa  PORTA

*      Comparing the PIN
      ldd    2,x          Load in RegD = acc + [Regx]
      cpd    pinno       16 bit compare RegD with memory
      beq    Fnd
      bra    Notfound
*      Test the limit--- PIN
Fnd    ldaa   #$FF
      staa  PORTB
      staa  PORTA

*      Getting the absolute difference from Heights
      ldaa   4,x          in RegD the value is =hgt+ [RegX]
      cmpa  hg           Comparing RegA with the input value
      blo   ckhg        branch when RegA < hg, i.e. hgt< hg
      suba  hg
      bra   sth
ckhg   ldaa   hg
      suba  4,x
sth    staa  H
*      Test the limits--- HEIGHT
      ldaa   #$01
      cmpa  H
      bhs   goto        branch When RegA >= H
      blo   Notfound    branch when RegA < H
goto   ldaa   #$FF
      staa  PORTB
      staa  PORTA
      jmp   next

*      To deny access.
Notfound ldaa  #$aa
      staa  PORTA
      stop

*      Calculating the absolute difference from Weights
next    ldaa   wtB1
      ldab  wtB2
      cpd   5,x

```

```

        blo    ckwg
        subd   5,x
        bra    stw
ckwg    ldd    5,x
        subd   wtB1
stw     std    P
*       Test the limit--- WEIGHT
        ldaa  #$05
        cmpa  P
        bhs   go1
        blo   Notfound
go1     ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt2

*       Calculating the absolute difference from the Finger Lengths
*       First Finger
nxt2    ldaa  7,x          Load in RegD = fn1 + [Regx]
        cmpa  fl1
        blo   ckf1       branch if less than
        suba  fl1
        bra   stf1
ckf1    ldaa  fl1
        suba  7,x
stf1    staa  J
*       Test the limit--- First Finger
        ldaa  #$05
        cmpa  J
        bhs   go2
        blo   Notfound
go2     ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt3

*       Second Finger
nxt3    ldaa  8,x          Load in RegD = fn2 + [Regx]
        cmpa  fl2          16 bit compare RegD with memory
        blo   ckf2       branch if less than
        suba  fl2
        bra   stf2
ckf2    ldaa  fl2
        suba  8,x
stf2    staa  K
*       Test the limits--- Second Finger
        ldaa  #$01
        cmpa  K
        bhs   go3
        blo   Nofound
go3     ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt4

*       Third Finger

```

```

nxt4   ldaa  9,x           Load in RegD = fn3 + [Regx]
        cmpa  fl3         16 bit compare RegD with memory
        blo   ckf3        branch if less than
        suba  fl3
        bra   stf3
ckf3   ldaa  fl3
        subd  9,x
stf3   staa  L
*      Test the limits--- Third Finger
        ldaa  #$01
        cmpa  L
        bhs   go4
        blo   Nofound
go4   ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt5

*      Fourth Finger
nxt5   ldaa  $0A,x        Load in RegD = fn4 + [Regx]
        cmpa  fl4         16 bit compare RegD with memory
        blo   ckf4        branch if less than
        suba  fl4
        bra   stf4
ckf4   ldaa  fl4
        subd  $0A,x
stf4   staa  M
*      Test the limits--- Fourth Finger
        ldaa  #$01
        cmpa  M
        bhs   go5
        blo   Nofound
go5   ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt6

*      Fifth Finger
nxt6   ldaa  $0b,x        Load in RegD = fn5 + [Regx]
        cmpa  fl5         16 bit compare RegD with memory
        blo   ckf5        branch if less than
        suba  fl5
        bra   stf5
ckf5   ldaa  fl5
        subd  $0b,x
stf5   staa  O
*      Test the limits--- Fifth Finger
        ldaa  #$01
        cmpa  O
        bhs   go6
        blo   Nofound
go6   ldaa  #$FF
        staa  PORTB
        staa  PORTA
        jmp   nxt7

```

```

*      To deny access
Nofound ldaa #$aa
      staa  PORTA
      stop

;      stop          temporay stop

*      Create and store one least significant bit from each inputs Using the Database
*      From Pin number
nxt7   ldx   #$EF00
fst    ldd   2,x
      andb  #$01
      stab  bit1
*      From Height
      ldab  4,x
      andb  #$01
      stab  bit2
*      From Weight
      ldd   5,x
      andb  #$01
      stab  bit3
*      From Finger length1
      ldab  7,x
      andb  #$01
      stab  bit4
*      From Finger length2
      ldab  8,x
      andb  #$01
      stab  bit5
*      From Finger length3
      ldab  9,x
      andb  #$01
      stab  bit6
*      From Finger length4
      ldab  $0A,x
      andb  #$01
      stab  bit7
*      From Finger length5
      ldab  $0B,x
      andb  #$01
      stab  bit8

*      Showing the bits in LCD's
      jsr  lfsr

*      Calculating the bits using the database
      xgdx          exchange contents of RegD with RegX
      addd  #$0C    add $0C to RegD
      xgdx          again exchange contents of RegD with RegX
      cpx   #$EFB4
      blo   fst

*      Create and store one least significant bit from each inputs
*      From Pin number
      ldd   pinno
      andb  #$01          least significant bit mask

```

```

    stab    bit1           store memory from RegB
*
    From Height
    ldab    hg
    andb    #$01
    stab    bit2
*
    From Weight
    ldd     wtB1
    andb    #$01
    stab    bit3
*
    From Finger length1
    ldab    fl1
    andb    #$01
    stab    bit4
*
    From Finger length2
    ldab    fl2
    andb    #$01
    stab    bit5
*
    From Finger length3
    ldab    fl3
    andb    #$01
    stab    bit6
*
    From Finger length4
    ldab    fl4
    andb    #$01
    stab    bit7
*
    From Finger length5
    ldab    fl5
    andb    #$01
    stab    bit8

*
    Showing the bits in LED's & LCD's
    jsr     lfsr
    jmp     again
    stop

```

*** Calculating the signature

```

    ORG    $E700

*
    Showing the bits in LCD's

*
    Subroutine lfsr
lfsr    ldaa    #$00
        oraa    bit1      8 bit Logical OR to RegA with bit1
        lsla    bit1      8 bit logical left shift to RegA
        oraa    bit2
        lsla
        oraa    bit3
        lsla
        oraa    bit4
        lsla
        oraa    bit5
        lsla
        oraa    bit6
        lsla
        oraa    bit7

```

```

Isla
oraa bit8
staa PORTA
staa PORTB
stop temporary pause at end of display and calculation
rts

```

* -----

ORG \$EF00

```

acc fdb 10212 1st Persons Account Number
pin fdb 1902 Pin Number
hgt fcb 75 Height
wgt fdb 202 Weight
fn1 fcb 95 Hand geometry (Multiplied by 100)
fn2 fcb 32 Hand geometry (Multiplied by 10)
fn3 fcb 36 Hand geometry (Multiplied by 10)
fn4 fcb 32 Hand geometry (Multiplied by 10)
fn5 fcb 24 Hand geometry (Multiplied by 10)

```

```

fdb 10213 2nd Persons Ac/No
fdb 1038
fcb 64
fdb 100
fcb 84
fcb 30
fcb 33
fcb 30
fcb 23

```

```

fdb 10214 3rd Persons Ac/No
fdb 8934
fcb 67
fdb 125
fcb 86
fcb 31
fcb 35
fcb 32
fcb 22

```

```

fdb 10215 4th Persons Ac/No
fdb 4823
fcb 75
fdb 169
fcb 10
fcb 33
fcb 37
fcb 34
fcb 26

```

```

fdb 10216 5th Persons Ac/No
fdb 4324
fcb 77
fdb 210
fcb 11
fcb 34

```

fcb	39	
fcb	35	
fcb	28	
fdb	10217	6th Persons Ac/No
fdb	9786	
fcb	69	
fdb	181	
fcb	94	
fcb	31	
fcb	36	
fcb	32	
fcb	24	
fdb	10218	7th Persons Ac/No
fdb	4253	
fcb	58	
fdb	105	
fcb	70	
fcb	28	
fcb	31	
fcb	29	
fcb	21	
fdb	10219	8th Persons Ac/No
fdb	1223	
fcb	62	
fdb	135	
fcb	80	
fcb	29	
fcb	31	
fcb	29	
fcb	21	
fdb	10220	9th Persons Ac/No
fdb	4395	
fcb	76	
fdb	195	
fcb	95	
fcb	30	
fcb	35	
fcb	31	
fcb	24	
fdb	10221	10th Persons Ac/No
fdb	9596	
fcb	69	
fdb	145	
fcb	95	
fcb	30	
fcb	35	
fcb	31	
fcb	24	
fdb	10222	11th Persons Ac/No
fdb	2434	


```

fcb    66
fdb    120
fcb    86
fcb    32
fcb    35
fcb    32
fcb    23

fdb    10223 12th Persons Ac/No
fdb    1255
fcb    60
fdb    152
fcb    82
fcb    30
fcb    34
fcb    31
fcb    23

fdb    10224 13th Persons Ac/No
fdb    3844
fcb    65
fdb    147
fcb    85
fcb    31
fcb    35
fcb    32
fcb    24

fdb    10225 14th Persons Ac/No
fdb    1234
fcb    68
fdb    193
fcb    13
fcb    35
fcb    40
fcb    36
fcb    29

fdb    10226 15th Persons Ac/No
fdb    6564
fcb    73
fdb    134
fcb    10
fcb    32
fcb    37
fcb    33
fcb    27

eod    fcb    0000  End of data base -----
        stop
        org    $FFFE
        dc.w  start
end

```

Appendix D

Verilog Program

Design.v

```
// This module describe the way the component modules should be connected
// together for the access verification system
// Created on 18th February
// Modified on 25th February
```

```
module alu_cell(d,g,p,a,b,c,S);
    output d,g,p;
    input a,b,c;
    input[2:0]S;
```

```
    wire bint,cint,dint;
```

```
    assign bint = S[0] ^ b;
    assign g = a & bint;
    assign p = a ^ bint;
    assign cint = S[1] & c;
    assign d = ((p ^ cint) & S[2]) | (dint & (~S[2]));
```

```
    assign dint = ((~S[1] & (~S[0]) & (a & b)) | ((~S[1] & S[0] & (a & b))
| (S[1] & (~S[0]) & (~a|b))) | (S[1] & S[0] & (a|b)) ;
```

```
endmodule
```

```
// _____
```

```
module alu16(d,a,b,Cin,S);
    output[15:0]d;
    input[15:0]a,b;
    input Cin;
    input[2:0]S;
```

```
    wire[15:0]c,g,p;
```

```
    alu_cell cell[15:0](
```

```
.d(d),  
.g(g),  
.p(p),  
.a(a),  
.b(b),  
.c(c),  
.S(S) );
```

```
endmodule
```

```
// _____
```

```
module multiplexer(Q,I0,I1,I2,I3,I4,I5,I6,I7,S0,S1,S2);  
input [15:0] I0,I1,I2,I3,I4,I5,I6,I7;  
input S0,S1,S2;  
output [15:0] Q;
```

```
mx81 mux[15:0](  
.Q(Q),  
.I0(I0),  
.I1(I1),  
.I2(I2),  
.I3(I3),  
.I4(I4),  
.I5(I5),  
.I6(I6),  
.I7(I7),  
.S0(S0),  
.S1(S1),  
.S2(S2));  
endmodule
```

```
// _____
```

```
module regN1(D,Q,Clk);  
input [15:0]D;  
output [15:0]Q;  
input Clk;
```

```
df001 ff1[15:0](  
.Q(Q),  
.C(Clk),  
.D(D));  
endmodule
```

```
// _____
```

```

module regN2(D,Q,Clk);
input [7:0]D;
output [15:0]Q;
input Clk;

df001 ff0(
.Q(Q[0]),
.C(Clk),
.D(D[0]));

df001 ff1(
.Q(Q[1]),
.C(Clk),
.D(D[1]));

df001 ff2(
.Q(Q[2]),
.C(Clk),
.D(D[2]));

df001 ff3(
.Q(Q[3]),
.C(Clk),
.D(D[3]));

df001 ff4(
.Q(Q[4]),
.C(Clk),
.D(D[4]));

df001 ff5(
.Q(Q[5]),
.C(Clk),
.D(D[5]));

df001 ff6(
.Q(Q[6]),
.C(Clk),
.D(D[6]));

df001 ff7(
.Q(Q[7]),
.C(Clk),
.D(D[7]));

cvss gnd15( .Q(Q[15]));

```

```

cvss gnd14( .Q(Q[14]));
cvss gnd13( .Q(Q[13]));
cvss gnd12( .Q(Q[12]));
cvss gnd11( .Q(Q[11]));
cvss gnd10( .Q(Q[10]));
cvss gnd9( .Q(Q[9]));
cvss gnd8( .Q(Q[8]));
endmodule

```

```
//
```

```

module lfsr8ty2(Reset, testpoints, lfsr8ty2, clk);
input Reset;
input[7:0]testpoints;
input clk;
output[7:0]lfsr8ty2;

```

```

wire pin1, hg1, wg1, fl1, fl2;
wire fl3, fl4, fl5, opeo8;
wire output10, output9, output8;

```

```

eo21 eor82_0(
.Q(pin1),
.A(testpoints[0]),
.B(lfsr8ty2[7]));

```

```

eo21 eor82_1(
.Q(hg1),
.A(testpoints[1]),
.B(lfsr8ty2[0]));

```

```

eo21 eor82_2(
.Q(wg1),
.A(output10),
.B(lfsr8ty2[1]));

```

```

eo21 eor82_3(
.Q(fl1),
.A(output9),
.B(lfsr8ty2[2]));

```

```

eo21 eor82_4(
.Q(fl2),
.A(testpoints[4]),
.B(lfsr8ty2[3]));

```

```
eo21 eor82_5(  
.Q(f13),  
.A(testpoints[5]),  
.B(lfsr8ty2[4]));
```

```
eo21 eor82_6(  
.Q(f14),  
.A(testpoints[6]),  
.B(lfsr8ty2[5]));
```

```
eo21 eor82_7(  
.Q(f15),  
.A(output8),  
.B(lfsr8ty2[6]));
```

```
eo21 eor82_8(  
.Q(output8),  
.A(testpoints[7]),  
.B(lfsr8ty2[7]));
```

```
eo21 eor82_9(  
.Q(output9),  
.A(testpoints[3]),  
.B(lfsr8ty2[7]));
```

```
eo21 eor82_10(  
.Q(output10),  
.A(testpoints[2]),  
.B(lfsr8ty2[7]));
```

```
reset re82_0(  
.Q(lfsr8ty2[0]),  
.C(clk),  
.D(pin1),  
.RN(Reset));
```

```
reset re82_1(  
.Q(lfsr8ty2[1]),  
.C(clk),  
.D(hg1),  
.RN(Reset));
```

```
reset re82_2(  
.Q(lfsr8ty2[2]),  
.C(clk),  
.D(wg1),
```

```

.RN(Reset));

reset reg82_3(
.Q(lfsr8ty2[3]),
.C(clk),
.D(f11),
.RN(Reset));

reset re82_4(
.Q(lfsr8ty2[4]),
.C(clk),
.D(f12),
.RN(Reset));

reset re82_5(
.Q(lfsr8ty2[5]),
.C(clk),
.D(f13),
.RN(Reset));

reset re82_6(
.Q(lfsr8ty2[6]),
.C(clk),
.D(f14),
.RN(Reset));

reset re82_7(
.Q(lfsr8ty2[7]),
.C(clk),
.D(f15),
.RN(Reset));

endmodule
// _____

module reset(Q,D,C,RN);
output Q;
input C;
input D;
input RN;

df011 ff(
.Q(Q),
.C(C),
.D(D),
.RN(RN));

```

```

endmodule
// _____

module design(accno, pin, height, weight, fl1, fl2, fl3, fl4, fl5, clk,
              indata, control32bit, s0, s1, s2, Access, Signature,
              Sign2, Reset, Result);
input[15:0] accno, pin, weight;
input[7:0] height, fl1, fl2, fl3, fl4, fl5;
input clk;
input Reset;
input[15:0] indata;
input s0, s1, s2; // control 3 bit address
output[1:0] Access;
output[7:0] Signature;
output[7:0] Sign2;
output[15:0] Result;
input[31:0] control32bit;

wire [15:0] accout,pout,hout,wout,flout,f2out,f3out,f4out,f5out;
wire [15:0] aluin1, aluout;
wire [15:0] threshold, difference;
wire [15:0] pthr, hthr, wthr, flthr, f2thr, f3thr, f4thr, f5thr;
wire zero;
wire[2:0] S;
wire[7:0] testpoints;
wire[7:0] tstps;

buf(S[0], s0);
buf(S[1], s1);
buf(S[2], s2);

// The 9 inputs are connected to the registers and then the outputs
// from the registers are connected to the multiplexers and the
// output connection are taken from the multiplexers to alu
// the database are also connected to the alu. Arithmetic
// operation is done by the alu and the output from the alu is
// connected to a register.

cvss ground(.Q(zero));

buf(testpoints[7], pout[0]);
buf(testpoints[6], hout[0]);
buf(testpoints[5], wout[0]);
buf(testpoints[4], flout[0]);
buf(testpoints[3], f2out[0]);
buf(testpoints[2], f3out[0]);

```



```
buf(testpoints[1], f4out[0]);  
buf(testpoints[0], f5out[0]);
```

```
regN1 reg0(  
  .Q(accout),  
  .Clk(clk),  
  .D(accno));
```

```
regN1 reg1(  
  .Q(pout),  
  .Clk(clk),  
  .D(pin));
```

```
regN2 reg2(  
  .Q(hout),  
  .Clk(clk),  
  .D(height));
```

```
regN1 reg3(  
  .Q(wout),  
  .Clk(clk),  
  .D(weight));
```

```
regN2 reg4(  
  .Q(f1out),  
  .Clk(clk),  
  .D(f1));
```

```
regN2 reg5(  
  .Q(f2out),  
  .Clk(clk),  
  .D(f2));
```

```
regN2 reg6(  
  .Q(f3out),  
  .Clk(clk),  
  .D(f3));
```

```
regN2 reg7(  
  .Q(f4out),  
  .Clk(clk),  
  .D(f4));
```

```
regN2 reg8(  
  .Q(f5out),  
  .Clk(clk),
```

```
.D(f15));

buf(tstps[7], access[1]);
buf(tstps[6], result[6]);
buf(tstps[5], result[5]);
buf(tstps[4], result[4]);
buf(tstps[3], result[3]);
buf(tstps[2], result[2]);
buf(tstps[1], result[1]);
buf(tstps[0], result[0]);
```

```
multiplexer mux1(
.Q(aluin1),
.I0(pout),
.I1(hout),
.I2(wout),
.I3(f1out),
.I4(f2out),
.I5(f3out),
.I6(f4out),
.I7(f5out),
.S0(s0),
.S1(s1),
.S2(s2));
```

```
// module control is in tesbench file..
```

```
alu16 alu1(
.d(aluout),
.a(aluin1),
.b(indata),
.Cin(zero),
.S(S));
```

```
regN1 reg9(
.Q(difference),
.Clk(clk),
.D(aluout));
```

```
lfsr8ty2 lfsr(
.Reset(Reset),
.testpoints(testpoints),
.lfsr8ty2(Signature),
.clk(clk));
```

```
multiplexer mux2(
```

```
.Q(threshold),  
.I0(pthr),  
.I1(hthr),  
.I2(wthr),  
.I3(f1thr),  
.I4(f2thr),  
.I5(f3thr),  
.I6(f4thr),  
.I7(f5thr),  
.S0(s0),  
.S1(s1),  
.S2(s2));
```

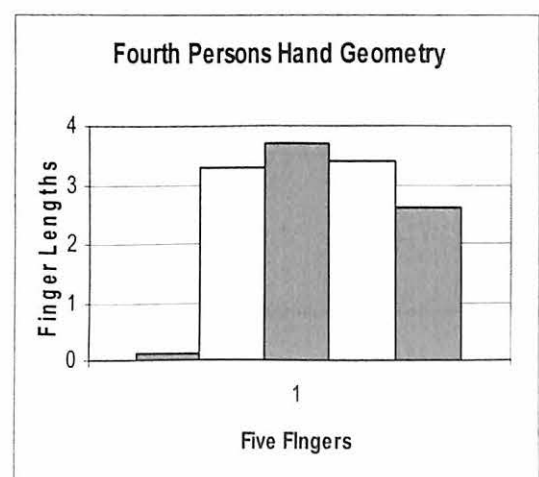
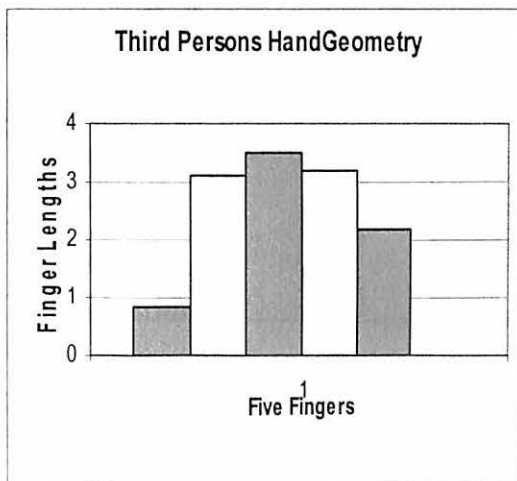
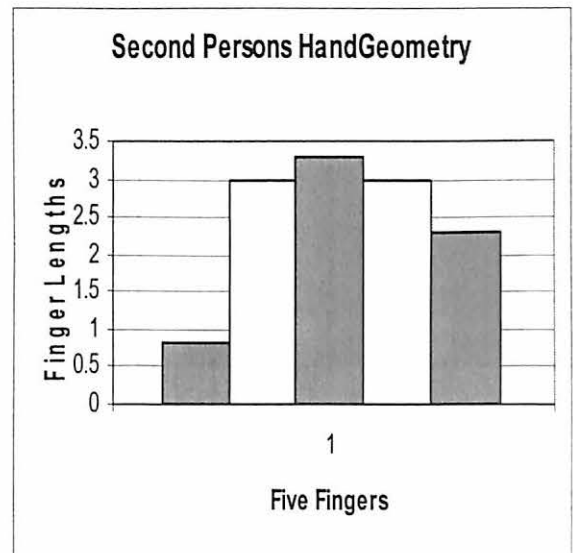
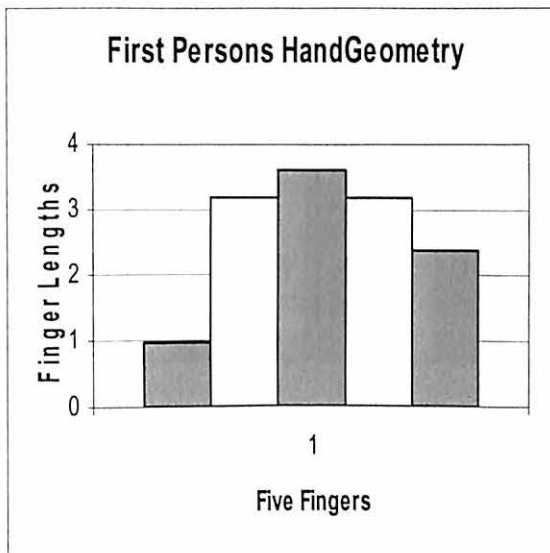
```
alu16 alu2(  
.d(Result),  
.a(difference),  
.b(threshold),  
.Cin(zero),  
.S(S));
```

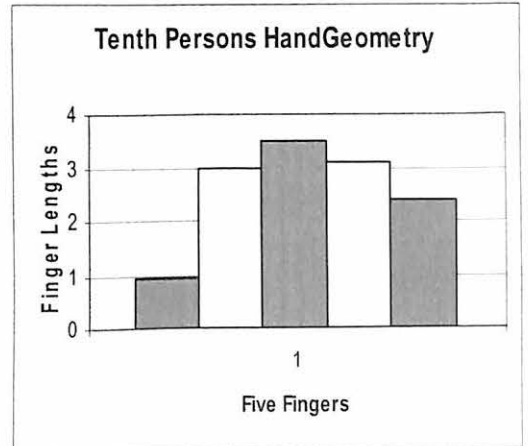
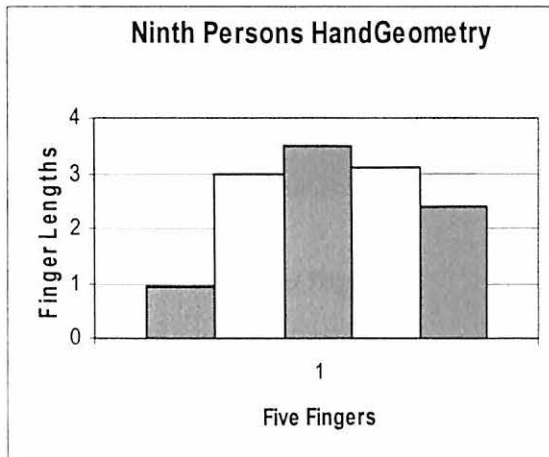
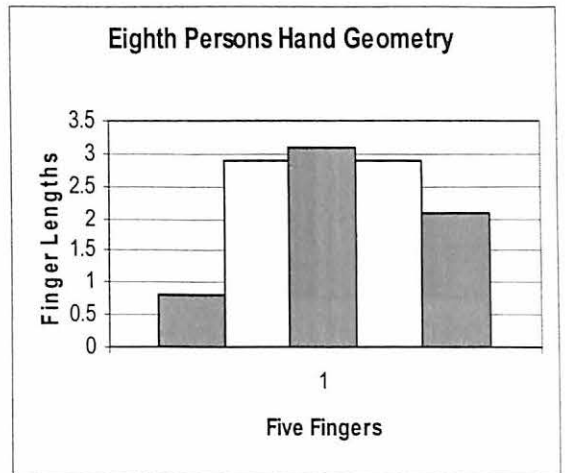
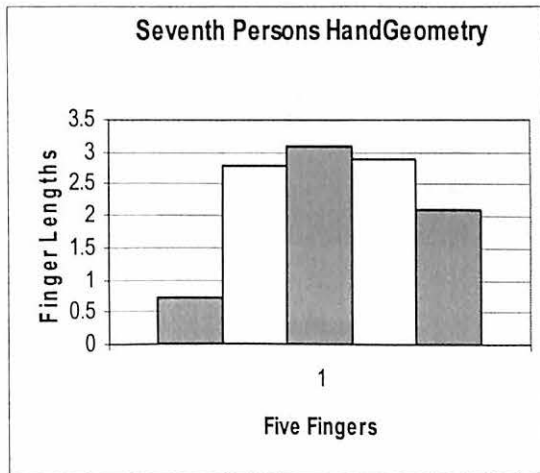
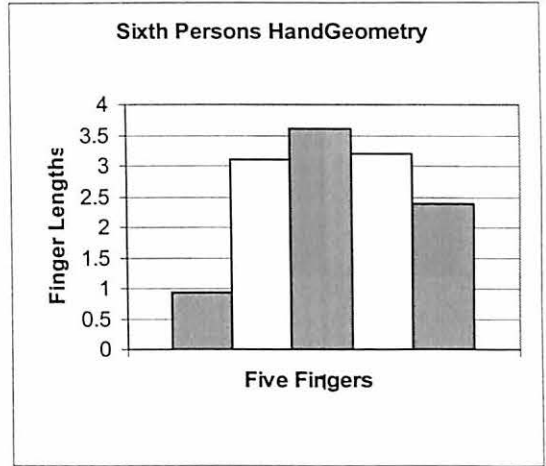
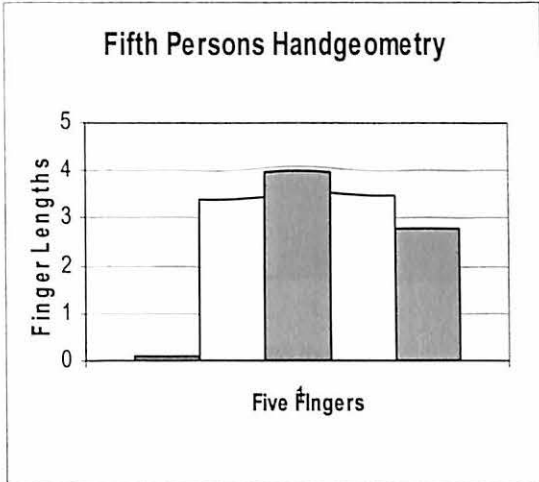
```
lfsr8ty2 lfsr2(  
.Reset(Reset),  
.testpoints(tstps),  
.lfsr8ty2(Sign2),  
.clk(clk));
```

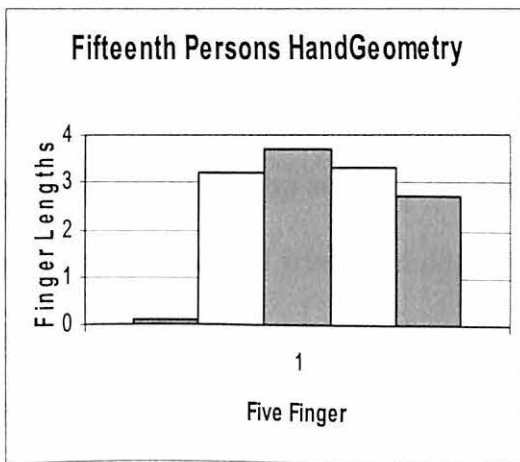
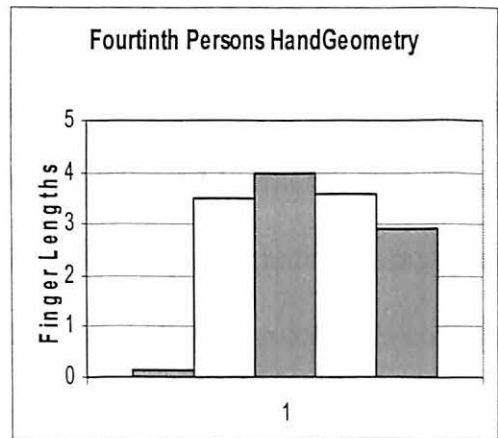
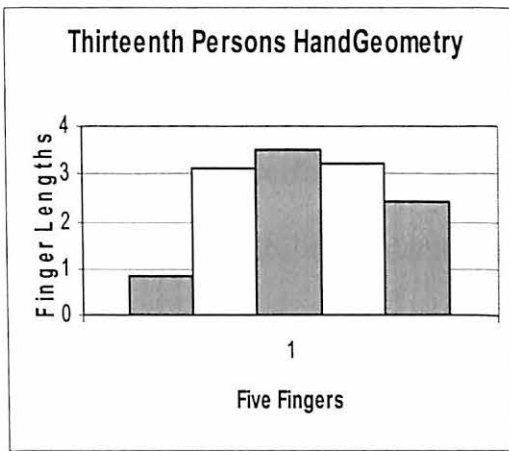
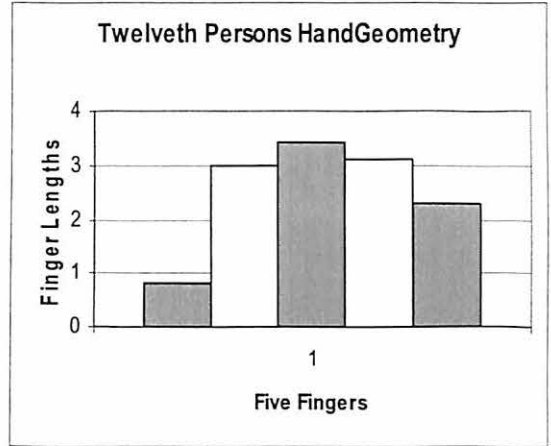
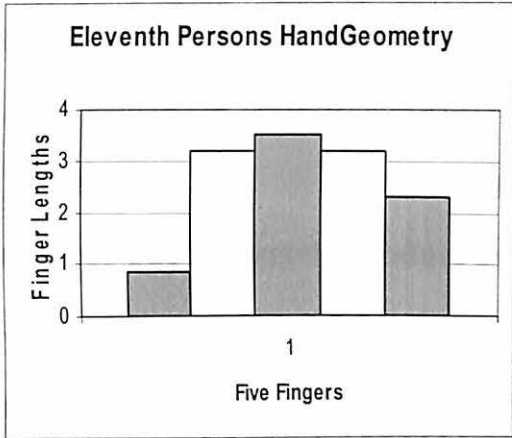
```
endmodule
```

Appendix E

Graphs showing the Hand Geometry of each person
(From the stored database)







Appendix F

Summary of The Biometric Product Testing -Final Report

“The Biometric product testing – Final Report” describes the performance evaluation of seven biometric systems conducted by National Physical Laboratory (NPL) over the period May to December 2000; sponsored by the Communications Electronics Security Group (CESG) as part of their Biometrics work program to support modernizing government and other initiatives. The objectives of the test program were:

1. To show the level of performance attainable by a selection of biometric systems
2. To determine the feasibility of demonstrating satisfactory performance through testing
3. To encourage more testing to be sponsored, and to promote methodologies contributing to the improvement of biometric testing

With this test program it is also hoped that this initial evaluation will promote the methodology to a wider audience and contribute to the improvement of biometric testing by other organizations and will encourage further testing to be sponsored. CESG and the Biometric Working Group selected the criteria for selection of systems to test. They are

1. Fingerprint, hand and iris technologies must be included. Other systems tested should use different technologies, except for fingerprint where two systems might be tested.
2. Within a technology, selection should be on the basis of wide availability and commonality of use.
3. Systems should be capable of meeting basic CESG performance requirements.

4. Systems should be testable under the agreed methodology (and, implicitly, the system performance should not be adversely affected by the proposed test protocol).
5. The vendor should be able to support the trials within the required timescales.

With these criteria, seven systems were selected for testing. Face, Fingerprint, Hand Geometry, Iris, Vein pattern and Voice recognition systems were tested for a positive identification in a normal office environment with cooperative users who will use the system a few times only, to avoid using volunteers who have extensively used one of the system under test. There were two fingerprint systems: one using optical fingerprint capture, the other a chip sensor. The evaluation was conducted in accordance with the “Best Practices in testing and Reporting Performance of Biometric Devices” produced by the UK Government Biometrics Working Group, and used 200 volunteers over a three-month period.

The tests were conducted in a room, which was previously in normal office use. The lighting levels were controlled. The devices were sited in accordance with recommendations of the product suppliers, and those most sensitive to changes in illumination were positioned away from the window. Similarly one device that was sensitive to background noise was located in a quieter area off the main test laboratory. The temperature and humidity of the test laboratory were not controlled.

The performance could be potentially affected by the order in which the devices were used. Upon on arriving at the test laboratory, if the volunteers have hurried to make their appointment they could be out of breath or have cold hands/fingers depending on the cold weather, can recover to a more normal state after a few minutes. The illumination for the face recognition system increased the amount of iris visible (i.e. reduces pupil size), with

a potential effect on iris recognition when this occurs shortly after the face recognition system. Feedback from one fingerprint device might affect user behavior (e.g. finger pressure) on the other. The transactions on the Voice system were not conducted until the volunteer had regained their breath. Other than this the order effects may also exist, but they are considered insignificant.

During preliminary testing it was observed that often more than two attempts would be required to obtain an enrollment. Particularly in the case with the Voice and both Fingerprint systems, to obtain a good quality image is more dependent on user behavior and familiarity. The enrollment software did not provide for re-enrollment for some systems. The problem enrollments were deleted, using the underlying operating system, before reenrollment was possible, and some systems did not automatically record every enrollment attempt failure. For these reasons the protocol for dealing with enrollment failures was modified. Immediate re-enrollment was attempted. At subsequent visits, whenever a volunteer had failed to enroll on one of the devices, they were asked to try reenrolling regardless of the number of previous enrollment attempts.

Data collection errors were avoided by asking the users to always use their right index finger, eye or hand as appropriate. For the supervisors it would be difficult to observe and prevent use of the wrong finger, hand or eye at enrollment or verification without this consistency. The saved images allow further checks that correct iris, hand or finger was used, though this is easier for iris and hand images than for fingerprint images. Each user was allocated a PIN for the trials. When using wrong PIN only the Voice, Face and Iris systems provided feedback on the claimed identity. This would show the individual and supervisor that failures were due to the wrong PIN being used. The PINs used to claim an

identity were chosen to minimize the chance that mistyping would produce another valid identity. If a PIN not being entered causes attempts to be recorded against the previous user's identity, these will be the 4th or subsequent attempts. However, these will be ignored as only the first 3 attempts per user per session are analyzed.

The 'Failure to enroll (FTE)' rate measures the proportion of individuals for whom the system is unable to generate repeatable templates. This includes those unable to present the required biometric feature (for example the Iris system failed to enroll the iris of a blind eye), those unable to produce an image of sufficient quality at enrollment, as well as those unable to reproduce their biometric feature consistently. For the systems tested it showed that the Enrolment failure rates are 1.0% for Fingerprint – Chip, 2.0% for Fingerprint – Optical and 0.5% for Iris. Face, hand vein and voice all have 0.0% FTE rate. In cases of difficulty, several attempts were allowed to achieve an enrollment.

The 'failure to acquire (FTA)' rate measures the proportion of attempts for which the system is unable to capture or locate an image of sufficient quality. This includes cases where the user is unable to present the required biometric feature (e.g. having a plaster covering his/her fingerprint), and cases where an image is captured, but does not pass the quality checks. Failure-to-acquire rates for the systems tested are for Face 0.0%, Finger Print Chip 2.8%, FP (Finger Print) chip (2) 0.4%, FP Optical 0.8%, Hand 0.0%, Iris 0.0%, Vein 0.0%, Voice 2.5%. The figures exclude the cases where the image was not captured due to user error (e.g. the user not positioning themselves correctly) as in these cases the attempt was simply restarted.

The fundamental operation of a biometric system is the comparison of a captured biometric image against an enrollment template. The false match and false non-match

rates (FMR and FNMR) measure the accuracy of this matching process. There is a trade-off between false match and false non-match errors by adjusting the decision criteria so the performance is best represented by plotting the relationship between these error rates in a detection error trade-off graph. Matching algorithm performance for each system, over a range of decision criteria, is shown in the figure. The iris system had no false matches in over 2 million cross-comparisons. The images corresponding to false non-matches showed that some of matching failures were due to poor quality images.

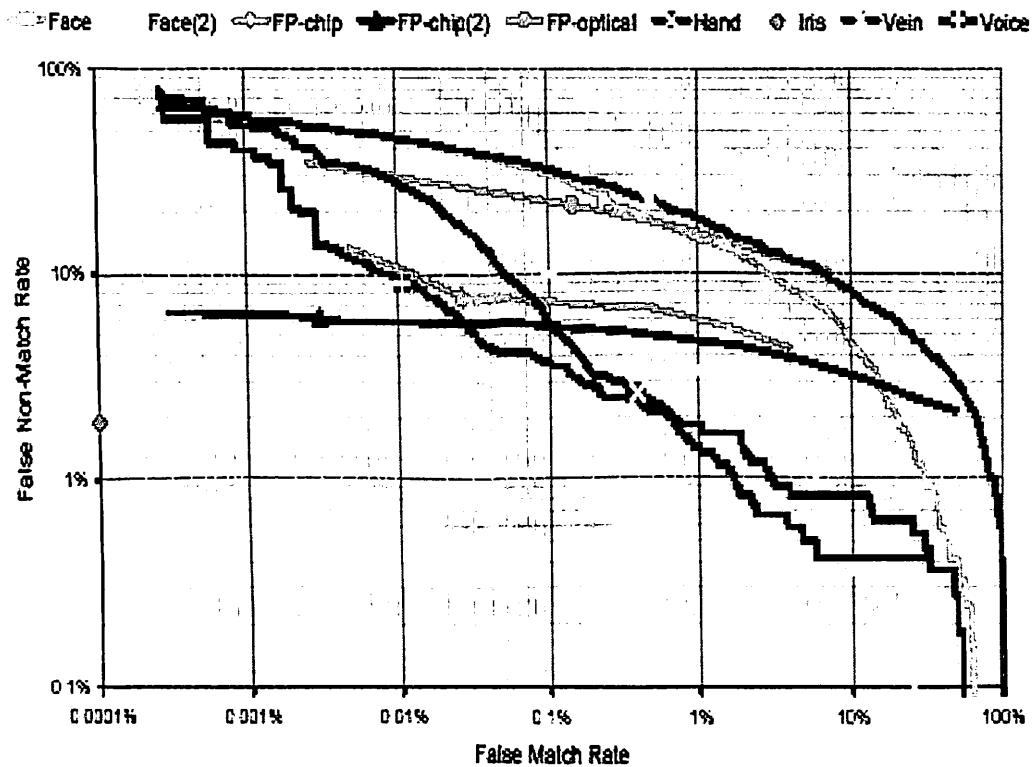


Figure13: Detection error trade-off: FMR vs FNMR

The system dealing with poor quality images, some will fail to acquire such images but some will accept poor image. That's why matching error rates should not be considered in isolation from the failure to acquire and failure to enroll rates. False acceptance rate (FAR) and False rejection rates (FRR) measure the decision errors for the whole system.

These measures combine matching error rates, and failure to acquire rates in accordance with the system decision policy. The false acceptance false rejection trade-off curve is shown in the figure. The curve for the face, hand geometry, iris and vein systems are unchanged, as these systems had no failures to acquire.

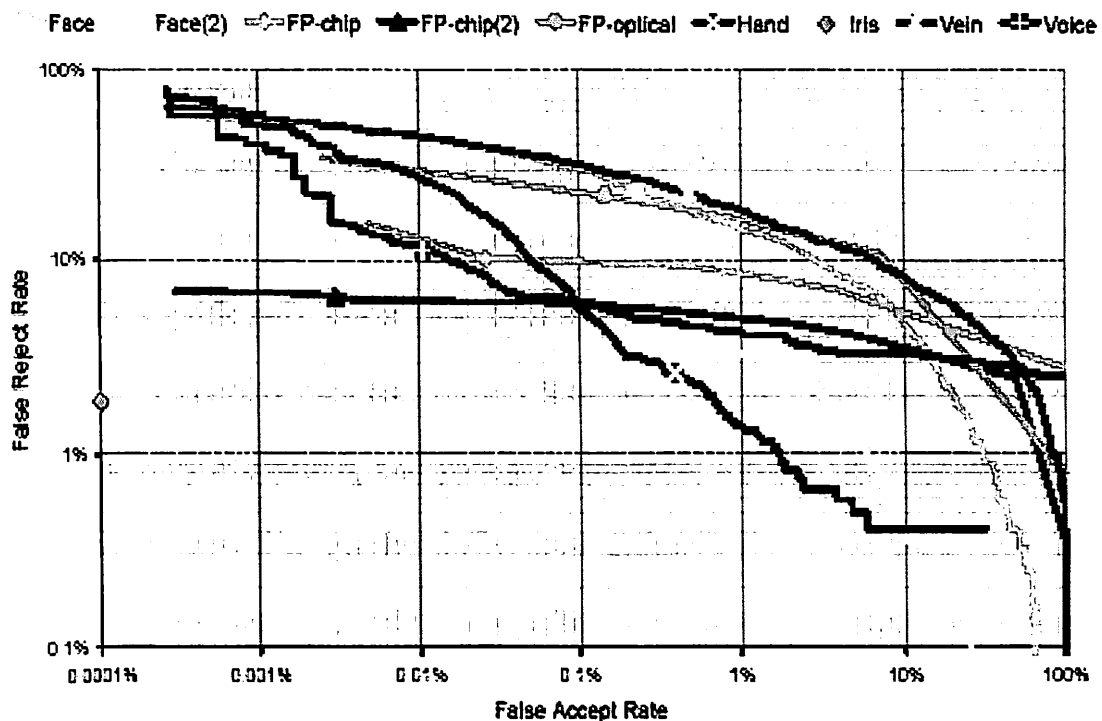


Figure 14: Detection error trade-off : FAR vs FRR

The time for a user transaction has been calculated using the time differences logged between consecutive transactions. User throughput measures the elapsed time of a single transaction. All attempts are timed at a consistent point during the transaction (e.g. the start time). The difference in times between the first and second, or second and third attempts, by an individual on one day approximates the total transaction time.

Performance differences by user and attempt types are also categorized. Attempts can be categorized by

- i. Whether the volunteer made at enrollment visit or at the second or third visit
- ii. The gender of the volunteer
- iii. Age of the volunteer
- iv. Whether the volunteer was wearing spectacles in the case of Face and Iris systems
- v. The length of the user's pass-phrase in the case of the Voice system.

The False rejection rates for attempts made immediately following enrollment were generally significantly lower (less than half) than those made at volunteer's second or third visit. Generally men had a lower false rejection rate than women (the voice system being the only exception), and younger volunteers a lower false rejection rate than their older colleagues. The gender differences appeared the more significant for the Face, Hand and Vein systems, and the age differences the more significant for the Fingerprint systems.

The evaluation has implemented the Biometric Working Group (BWG) proposed methodology for biometric testing, validating many aspects of the methodology. For example demonstrating the feasibility of the methodology, showing that the number of volunteers used (200) is sufficient to evaluate performance of biometric systems at their current level of accuracy. The practical significance of issues described in "Best Practices" has been demonstrated such as the need for time separation between enrollments and verification attempts, the need to minimize the chance of labeling errors, the modified procedures to simulate unknown impostor attempts when there are

dependencies between templates. A single evaluation cannot demonstrate repeatability of the results. However, Some of the devices evaluated have been tested elsewhere in similar scenarios, and the results are consistent.

In many biometric systems, a sequence of images is processed in a single verification attempt. For example, with the trial system it appears that the face system collects images over a period of 10 seconds, and gives the best match obtained, the Chip-based Fingerprint system collects images until a match is obtained, or until timeout, the Optical Fingerprint system scans for fingerprints until an image of sufficient quality is obtained, or the timeout is reached, the Hand Geometry system occasionally requires a second hand placement, when the score is very close to the decision threshold, the Iris system collects images until a match is achieved or until timeout. The current version of Best Practices does not explicitly deal with these cases, yet this mode of operation can sometimes bias off-line calculations using the collected data.

In the case of the tested Optical Fingerprint, Hand Geometry and Iris systems, the image collected does not depend on the template being matched. With the Fingerprint Chip, the collected image might instead be last before timeout; and, apart from image quality, should be equivalent to the image saved from a genuine attempt.

Different systems handle poor quality input in different ways. With some systems this may result in a failure to acquire, and with others a matching failure. In this respect the FAR-FRR trade-off graph (Figure:14) provides a better comparison of performance than the FMR-FNMR trade-off graph (Figure:13). Systems may have other adjustable parameters affecting performance in addition to an adjustable decision threshold. These allow different performance trade-offs (which, depending on the application, may be

more important than the FAR-FRR trade-off). For example, with the Face, Iris, and Chip-Fingerprint systems, which try to match collected images over a fixed time period, there is a trade-off between the time allowed and the false rejection rate.

The performance of the Biometric system is dependent on the application, environment and population. Therefore it was mentioned that the performance results presented in the report should not be expected to hold for all other applications, or in all environmental conditions. In particular caution should be exercised when comparing these results with those of other systems tested under different conditions.

VITA #1

NAHREEN MAHERUKH

Candidate for the Degree of

Master of Science

Thesis: A NEW DESIGN METHODOLOGY FOR HIGH QUALITY DESIGN

Major Field: Electrical Engineering

Biographical:

Personal Date: Born in Dhaka, Bangladesh, October 5, 1971, daughter of Abdul Mannan Chowdhury and Fatema Chowdhury.

Education: Graduated from Begum Badrunnesa College, Dhaka in 1988; received Bachelor of Science Degree in Physics from University of Dhaka, Dhaka in 1994. Received Master of Science Degree in Physics from University of Dhaka, Dhaka in 1996. Completed the requirements for the Master of Science Degree with a major in Electrical Engineering at Oklahoma State University in May 2004.