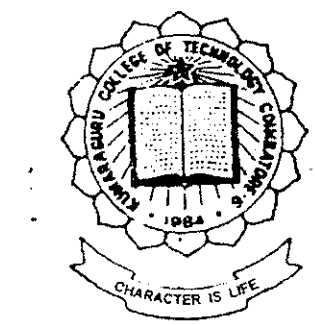
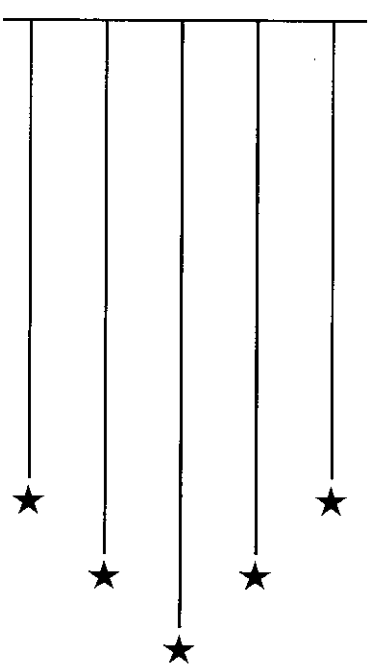
SECRET KEY ENCRYPTION AND DECRYPTION OF DATA USING PSEUDORANDOM SEQUENCE





Project Report 1998 - 99

Submitted by
BISHNOO ANANTH
JOTHISH .K
SIVAKUMAR .S
KARTHEE .S
RAMAGOPPALUN .J

Under the Guidance of
Mr. R. SIVAKUMAR, M.E.,

Submitted in partial fulfillment of the requirement for the degree of
BACHELOR OF ENGINEERING
in the Electronics & Communications Engineering Branch of the
BHARATHIAR UNIVERSITY,Coimbatore.

Department of Electronics & Communication Engineering
KUMURAGURU COLLEGE OF TECHNOLOGY
Coimbatore - 641 006

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641006

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the report titled

SECRET KEY ENCRYPTION AND DECRYPTION OF DATA USING PSEUDORANDOM SEQUENCE

has been submitted by

MR. BISHNOO ANANTH, J.RAMAGOPPALUN, K.JOTHISH, S.SIVAKUMAR, S.KARTHEE

In partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in the Electronics and Communication Engineering branch of the Bharathiar University, Coimbatore – 641006 during the academic year 1998-'99

R. Sria 13/3/99

(GUIDE) **(HEAD OF THE DEPT.)**

Certified that the candidate was examined by us in the project work VIVA-VOCE examination held on 15-03-1999 *and the University Register Numbers are*

9527D0183, 9527D0210, 9527D0183

9527D0227, 9527D0191

(Internal Examiner) **(External Examiner)**

INDEX

ACKNOWLEDGEMENT

At the offset we would like to express our deepest gratitude towards our principal **Dr. K.K.Padmanabhan B.Sc.(Engineering), M.Tech., Ph.D,** for his unerring support.

We would also like to thank sincerely our Head of the Department **Prof. M. Ramasamy M.E., MIEEE(USA), MISTE** for the innumerable facilities provided to us during the course of our project.

Our guide **Mr. R.Sivakumar M.E, D.B.A,M.I.S.T.E.,** has been the constant source of encouragement, suggestions and our guiding light. Nothing we say could hope to express completely the respect and gratitude that we feel for his perennial commitment towards us.

We would be failing in our duties if we forget to thank the highly capable faculty and assistants of our department without whom this project would not be the success that it is today.

In conclusion we would like to gratefully acknowledge the technical contribution of **Mr. BalajeeSrinivas** of ***Hi-Tech Solar Appliances*** and **Mr.Vinod Collins** of ***Micro Trend Systems*** towards the completion of this project.

SYNOPSIS

The proposed encryption system is a simulation of a One-time key pad cipher. The process involves basically the mixing of computer data with a pseudo-random noise sequence also referred to technically as PN sequence, so that it becomes unassuagable by an unauthorised interceptor.

The sequence, though a collection of bits, when generated at a very high speed, takes on the form of noise non- decryptable into the actual PCM plain text. This technique can be used to the benefit of the end user as the probabilities of the source are totally eliminated in the encrypted output.

Added security is provided by the means of programming the process onto a chip using VHDL software there by making it virtually impossible to reconstruct the PN generating circuit. The medium of communication may be either an optical fiber or coaxial cable link between the computers to transmit and receive data.

INTRODUCTION

Secure communication in a hostile environment is a challenging task. It is important that any message originating from a source in a clandestine operation reaches only the right ears. Not only does this pose the usual job of effecting communication, but also compels proper recoverable deformation of the original information before its journey towards its destination. We have just entered the world of cryptography - the art of changing a message that only the intended receiver can comprehend. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience and determination.

There are many approaches to encrypt messages right from primitive methods like letter to letter substitution to the more modern key approach.

Two means of implementation are

1. Using software and
2. Using hardware.

The hardware option is more appropriate in the sense that only possession of the hardware key gives anybody the authority to access the original information. Moreover given the present developments in the VLSI field, implementations in hardware becomes that much simpler and effective.

HISTORY OF CRYPTOGRAPHY

The story begins:

When Julius Caesar sent messages to his trusted acquaintances, he did not trust the messengers. So he replaced every 'A' by a 'D', every 'B' by an 'E' and so on through the alphabet. Only someone who knew the "Shift by Three" rule could decipher his messages.

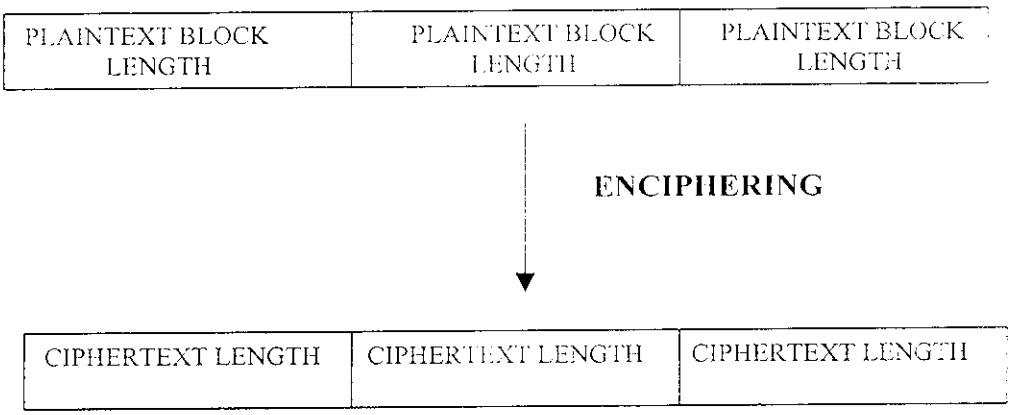
A **Crypto System** or **Cipher System** is a method of disguising messages so that only certain people can see through the disguise. The original message is called a **plain text**. The enciphered message is called a **Cipher text**. The technique by which the enciphered message is manipulated to get back the plain text is called a deciphering system. The crypto system is usually labeled. These labels are what are known as "**Keys**" .

ENCIPHERING TECHNIQUES :

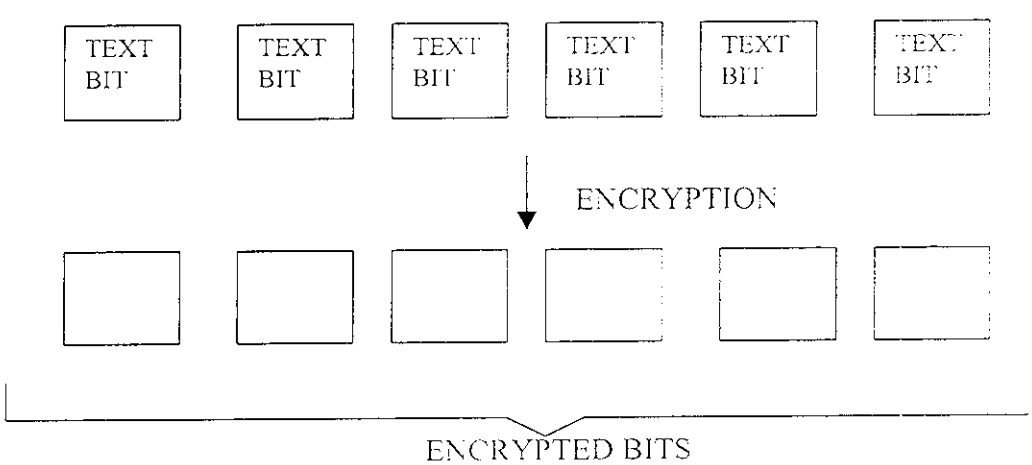
BLOCK ENCRYPTION:

The plain text is partitioned into blocks of data. Each block is then individually coded using the same key for all the blocks. This method is prone to attacks and hacking as even if starting or ending of one block is found out, the above message can be easily obtained.

BLOCK ENCIPHERMENT



STREAM CIPHERMENT



STREAM ENCRYPTION:

This algorithm works on a word to word (or) character-to - character basis so that similar plain text symbols have a different cipher text symbols. This can be either a synchronous cipher or asynchronous cipher. In a synchronous cipher, the encryption occurs according to the position of the data and is independent of preceding or succeeding information. The major disadvantage is that if an error occurs, it would affect the entire stream cipher. In asynchronous cipher some memory is provided. Also the coding is not made in terms of position and the memory is used for error elimination.

ENCRYPTION ALGORITHMS:

TRANSPOSITION CIPHERS

This is a primitive algorithm which changes the position of or sequence of the characters of the input blocks. To encipher, plain text is broken into 'N' symbols and the key specifies (N-1)! Permutations. Deciphering involves a reverse process. This maintains single letter frequency distribution but destroys the diagram.

SUBSTITUTION CIPHER:

Substitution ciphers substitute one set of characters by another. A monoalphabetical substitution cipher occurs where there is a one to one correspondence between the symbols and the substitutes. Homophonic substitution

involves each plain text symbol to be replaced by another form of cipher text symbol library. More important is the 'One key pad cipher' because the key is random, non-repeating and used only once.

TYPES OF CRYPTANALYTICAL ATTACKS

Cryptanalytical attacks are of three types:

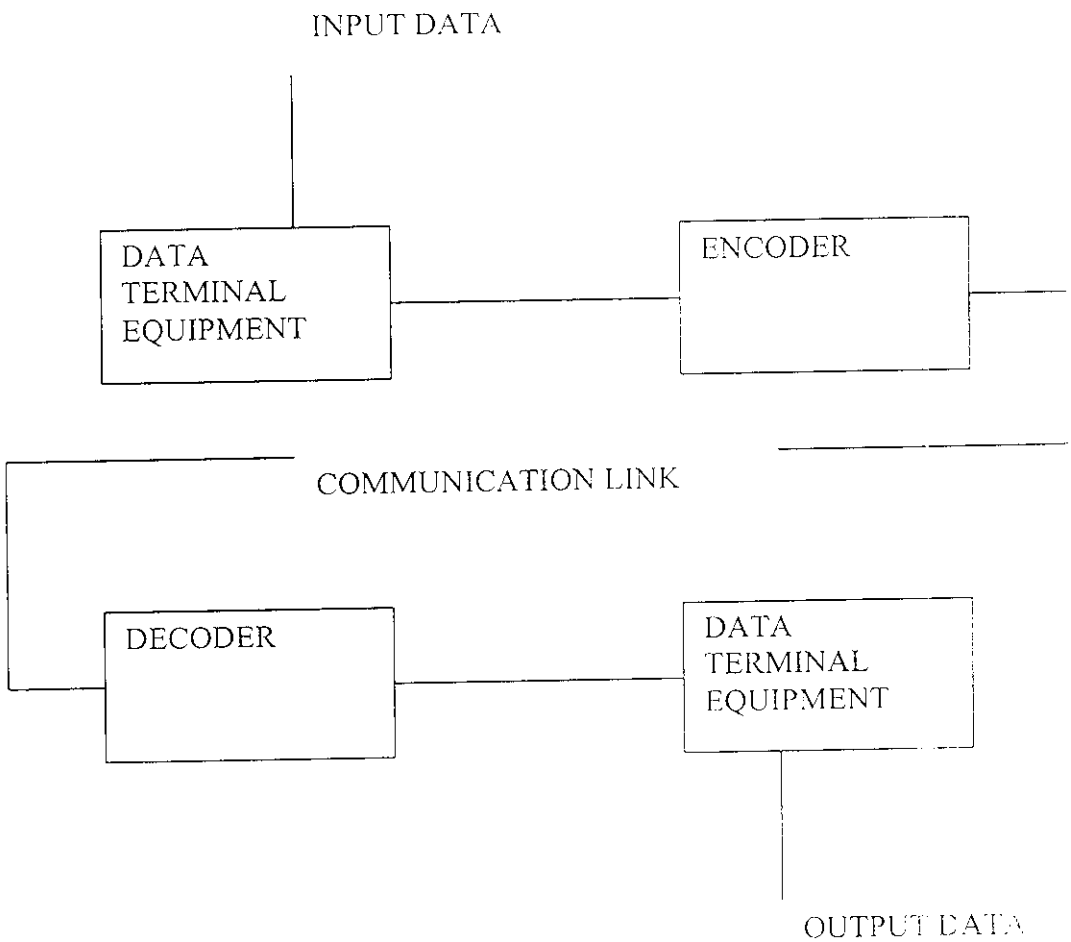
1. The cipher only attack
2. Known plain text attack
3. Chosen text attack

The chosen text attack is the most powerful, as the cryptanalyst has both the plain and the cipher text. It is assumed that he also knows the enciphering algorithm altogether. Hence the question of cryptosecurity comes in.

Ciphers are either unconditionally or computationally secure. Unconditional ciphers are impossible to break, as the information contained in the cipher text is not enough to solve them. An example is the one time keypad where the keys required are as long as the text as messages.

Current security techniques involve subjecting the cipher in the worst possible attack in order to determine the applicability and robustness of the algorithm.

PICTORIAL REPRESENTATION OF A CRYPTOSYSTEM



DESCRIPTION OF OUR CRYPTO SYSTEM

TERMINAL:

Any Equipment, from which plain text, which is to be encrypted, is obtained. If the data that is obtained is in parallel form then a suitable interface is used to convert it to a serial form and is then fed to the Encoder module. Some examples of terminal are Computer, Serial Printers etc.,

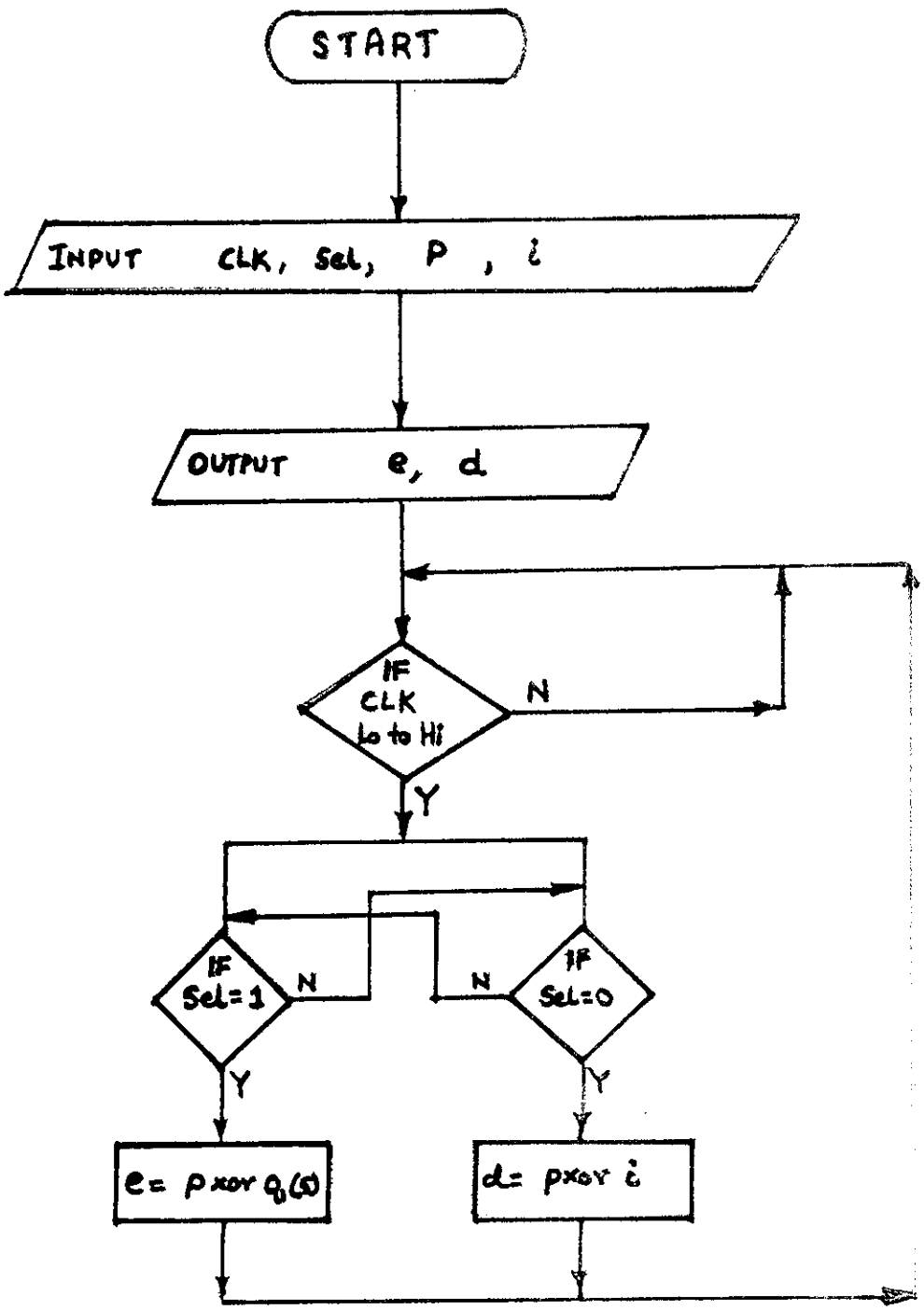
ENCODER:

This Block consists of both an Encryptor and a Frequency Translator and takes care of manipulating bits to obtain the Cipher text. A pseudo random key generator is used to encrypt the message obtained from the Terminal and to perform Frequency Translation. The encrypted message, which is in the form of PN sequence when accessed by an unauthorised interceptor, yields unassuagable information.

COMMUNICATION LINK:

It is the Communication media through which the encoded data is transmitted. It serves as a Bridge between Encoder and Decoder. The link can take the form of Coaxial cable, Optical fiber or air. The medium is chosen based on performance, cost and application. Naturally our choice of the Communicating medium based on the above said criteria is the coaxial cable.

Encryption / Decryption Module



DECODER:

It is the receiver end unit, where in the Decipherment of the Cipher text is accomplished to get back the original Plain text. In our Design the Decoder resembles the Encoder, but for the additional Circuitry to get back the original message.

ENCRYPTION:

In our System the input is fed to the encoder which comprises of a shift register, Modulo-2 adder and a clock. The PN sequence is generated using Shift register , clock and a combinational circuit. The combinational circuit is used to deform the linearity of the sequence. The various building blocks of our encryptor module have been developed using VHDL programming to effectively maintain utmost security.

ENCRYPTION ALGORITHM

i) Define the inputs Clk, P, Sel

ii) Define output pin E

iii) Check for clock

iv) Check for clock low to high transtion Check for Sel = high

v) At each transition

E= (P xor $q_1(5)$)

vi) goto step iii

DECRYPTION:

The Decryptor circuit, which is kept at the receiving end does the process of extracting the original message from the random and the unintelligible encrypted form. This module consists of a 8-bit Serial In Parallel Out Shift register, manipulator module pertaining to a Boolean expression and a couple of modulo-2 adders. They are designed in such a manner that the original message is received correctly and efficiently.

Our system ensures complete security in terms of data management and has the flexibility to be adopted to any kind of system.

DECRYPTION ALGORITHM

i) Define inputs clk, P,sel

ii) Define output pin D

iii) Check for clock

iv) Check for clock low to high, check for sel = low

v) At each transition

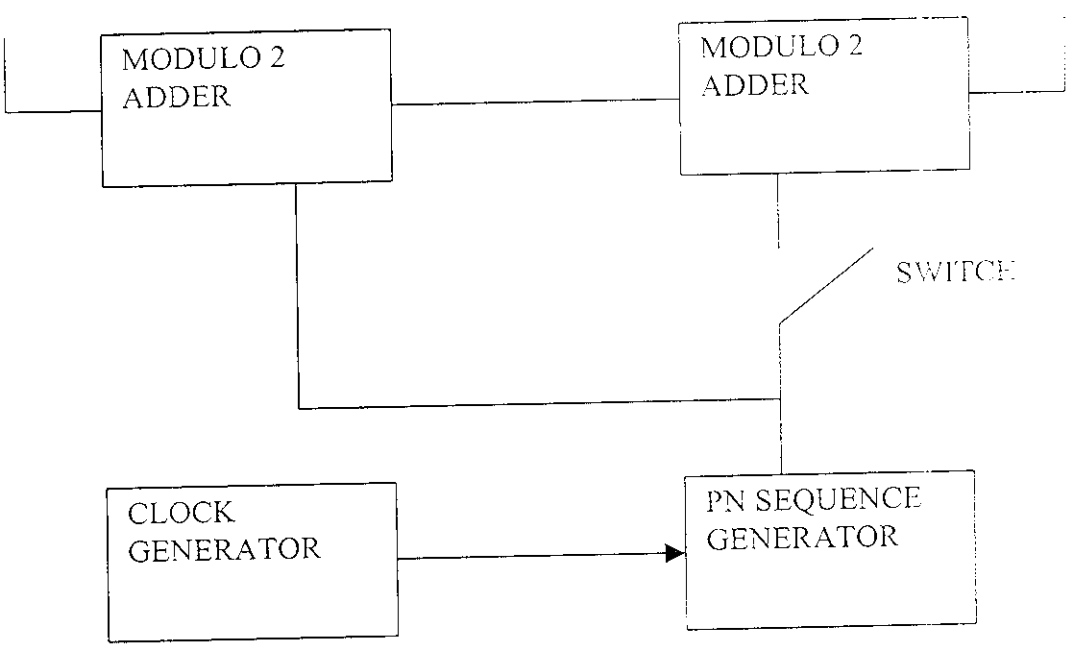
D= (P xor i)

vi) Go to step iii

ENCRYPTION DECRYPTION MODULE

DATA

DATA



A DETAILED STUDY

SUMMARY:

The project deals with semi-one-time pad enciphering in the form of pseudorandom noise encryption. Data is got into the chip in the form of PCM signals and is modulo-2 added with the PN sequence generated by shift register according to the input given and also according to the clock.

Mode selection as to whether the chip works in encrypt or decrypt state in done via a select pin, which has an appropriate input.

Decryption in done on the basis of the shift register output that was used initially to encrypt the data by performing a second modulo -2 addition.

Since the advent of the idea of communication between computers, there has been a mad rush to improve speed, accuracy and expandability of the link.

Somewhere down the line the attention was shifted to the actual security and secrecy in the data network. The long journey that cryptography has under taken from the times of Julius Caesar to the present day algorithms has indeed been one filled with technological advancements and breakthroughs. The chip presented here boasts to be one small step in the giant path that is cryptography.

SPECIFICATION: -

The IC used for programming is the PALC-22V10 IC which has been programmed in the 1164/VHDL mode.

The various inputs to the chip are CLOCK (CLK), SELECT (Sel), PCM i/p (P), Vcc (+5V) and GROUND (Gnd).

The outputs are ENCRYPT (E) and DECRYPT (D) pin

DESCRIPTION OF FUNCTION

The various pins defined and their respective specifications and ranges are as follows

CLK : This is the clock-input pin to which the 1 MHz (square wave) is given via the crystal. The range is from 800KHz to any upper limit

P : This is where the PCM input is given to the chip. The input is of square nature and is taken from the TXD pin of the computer. This is the signal, which is to be encrypted.

E : This is an output pin and it is where the PN Sequence is modulo-2 added with the signal at P. There is no definite structure for the output at this pin and is predominantly noise.

D : This is the decrypt pin from where, on giving the E pin signal to the P pin, the original PCM signal is taken and fed to a computer. The signal here is of square nature.

SEL : The SEL or select pin is responsible for the chip working in encrypt or decrypt mode. On giving high, the chip works in Encrypt mode with decrypt pin disabled and on low it works as Decrypting chip with encrypt pin disabled.

SYSTEM DESCRIPTION: -

In this project the operation and construction of a digital security system is described. The block diagram is shown in appendix. The system has a pulse code input, PN sequence generator with encryptor / decryptor and pulse code output.

Modulo - 2 adder is used to obtain the enciphered signal by adding the PCM signal with the output of pseudorandom number generator.

When the switch S is low, the output of the modulo-2 adder will be the deciphered PCM signal. If the switch S is high, the output will be just noise.

CIRCUIT DESCRIPTION: -

CLOCK: -

A 1 MHz TTL compatible clock is generated using

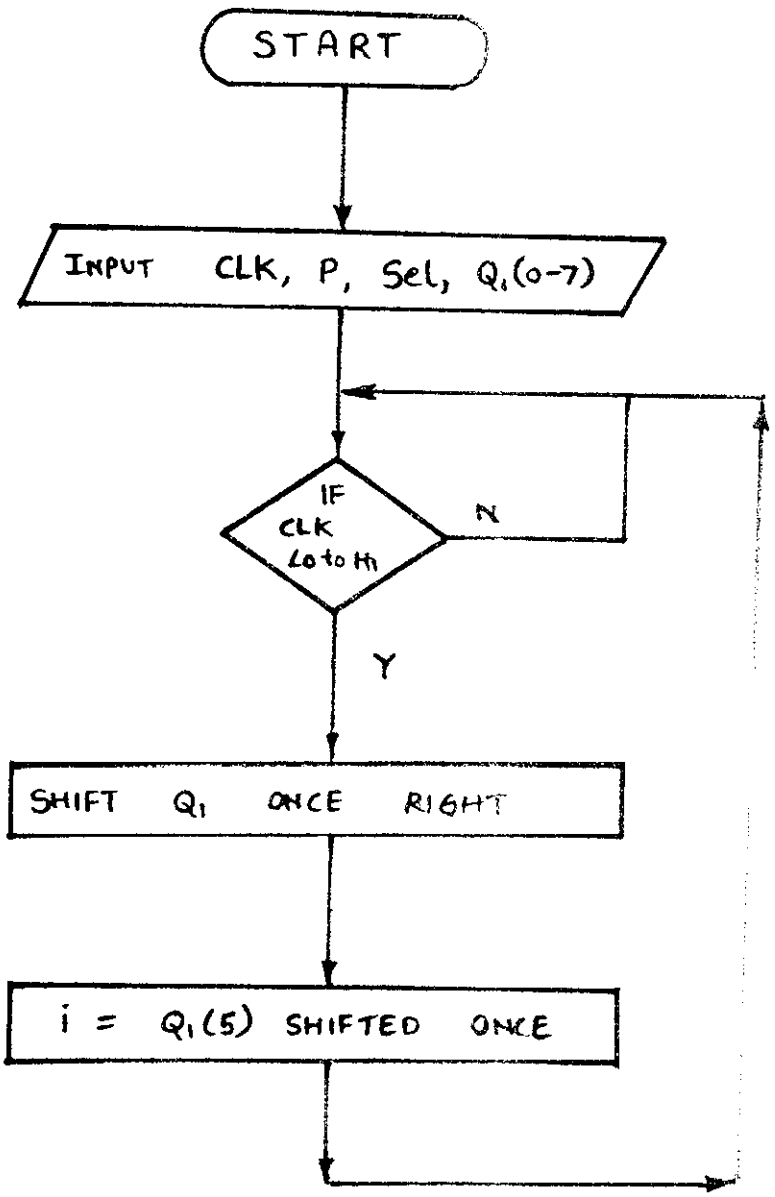
Invertor gates in crystal stabilized astable mode. As the crystal has properties of stability and accuracy, the 1MHz clock output is highly accurate and stable. It is to be noted that the clock to PCM rate ratio has to be a minimum of 1:80 for proper encryption of data.

PN SEQUENCE

This system uses a six stage PN (pseudorandom number) sequence generator as noise generator. The need is for a serial in parallel out shift register. To generate a truly pseudorandom sequence, there has to be feedback connections from at least two outputs of the shift register through an EX-OR gate. But if at the starting instant, all the six stages of the shift register are filled with zeroes, then the PN sequence generator output will be all zeroes and therefore of no use.

FLOWCHART

SHIFT REGISTER



To avoid this problem, we have to set the first stage output of the shift register to one, when the contents of all the six stages are zeroes. This is achieved automatically by using combinational logic circuits. Using these we get a PN sequence at the 'i' pin of the encryptor / decryptor circuit.

ENCRYPTOR / DECRYPTOR: -

The PN sequence and the digitized serial PCM information are subjected to modulo - 2 addition, using xor gates. The buffered output of the gate is the encrypted PCM output.

After giving a signal of low at the pin "Sel", the same encrypted signal can be given as the input at the PCM pin and so the decryptor will, through the second EX-OR gate decrypt the input. Under this condition, we recover the original digital PCM signal.

MODULAR ALGORITHM

a) SHIFT REGISTER MODULE

i) Define the inputs Clk, Sel, P, q_1 (0to7).

ii) Define the outputs d,e.

iii) Check for the signal at the Clk pin.

iv) If the Clk transition from low to high is sensed, then continue process else go to step iii.

v) Begin process.

vi) Start the shift right operation using the technique of back shifting

$q_1(7) = q_1(6)$

$q_1(6) = q_1(5)$

$q_1(5) = q_1(4)$

$q_1(4) = q_1(3)$

$q_1(3) = q_1(2)$

$q_1(2) = q_1(1)$

$q_1(1) = q_1(0)$

$q_1(0) = d\_in$

at each transition.

vii) d_in to the shift register in defined as

$d\_in=(q_1(5)$ xor $q_1(0))$ or(not(((((($q_1(0)$ or $q_1(2))$ or $q_1(3))$ or $q_1(4))$ or $q_1(5))$

viii) Shift $q_1(5)$ once right and move it to i as the PN sequence

ix) goto step vi

NONLINEAR COMBINATION OF LINEAR SHIFT REGISTER SEQUENCES:

As the running key generators in stream ciphers, pseudo - random sequence generator from shift-registers combined by some non-linear function have been proposed by several cryptologists for crypto-applications. PN sequences exhibit certain statistical properties and linear complexity. The number of memory cells and the feedback connections in the shift register largely govern these properties. Many proposed key stream generators consist of a number of maximum-length shift registers combined by non-linear functions. The non-linear function should destroy the linearity in such a manner that it gives the output sequence a large linear comp.exity. Many cryptologists have striven to find non-linear combination techniques of shift register sequence that will produce the output sequence with both good pseudo-random properties and high linear complexity. The implementation of this technique is the main idea of our project. The purpose of the non-linear combinations is to produce an equal system, which can withstand any cryptanalytic attack. In order to avoid correlation attacks the combining function must be correlation-immune and the output sequence from the non-linear combiner should also be statistically independent of the input sequences.

"RANDOMNESS" IN CRYPTOGRAPHY

Cryptographic applications demand much out of pseudorandom number generators than most applications. For a source of bits to be cryptographically random, it must be computationally impossible to predict what the N^{th} random bit will be, given the complete knowledge of the algorithm or hardware generating the stream and the sequence of 0^{th} through N-1 bits, for all N, unto the lifetime of the source.

The software generator (also known as pseudorandom) has the function of expanding a truly random seed to a longer string of apparently random bits. This seed must be large enough not to be guessed by the opponent. Ideally it should also be truly random (perhaps generated by a hardware random number source).

PSEUDORANDOM NUMBER AS KEY STREAM

Chaotic equations and fractals produce an apparent randomness from relatively compact generators. Perhaps the simplest example is a linear congruential sequence, one of the most popular types of random number generators, where there is no obvious dependence between the seeds and outputs. Unfortunately the graph of any such sequence will, in a high enough dimension, show up as a regular lattice. Mathematically this lattice corresponds to a structure, which is notoriously simple for cryptanalysts to exploit. More complicated generators have more complicated structures, which is why they make interesting pictures. A cryptographically strong sequence will have no computable structure at all. Also a computer based pseudorandom generator is in close proximity to a one-time keypad except for its deterministic properties.

COMPUTER - COMPUTER COMMUNICATION

The ability of the computer to service many input - output devices simultaneously have made data communication essential. It was the ability to handle multiple tasks and numerous remote terminals, which promoted the rise of the data transmission industry. Initially standardization was sought for the interconnections needed between the computer and the various peripheral devices. Standardization took the form of standard connectors, signaling formats and signal levels. As these standards become recognized by the industry, it become desirable to extend them to the transmission media used for medium and long - haul transmission of data.

The need for transmission standards became really acute when computer facilities began to use the telephone system for their transmission requirements. The pervasiveness of the telephone system made it ideal for the interconnection of computers with remote sites. As the need for data transmission increased, it became advantageous for data uses to be accommodated over standard voice grace channels. Data communication now has its own language, equipment and standards. It is an industry in itself and is certainly an integral part of the current computerized society.

CHARACTERISTICS OF DATA TRANSMISSION CIRCUITS

Data in most instances consists of pulse type energy. The data stream is similar to a square wave signal with rapid transmissions from one voltage level to another, with a repetition rate depending on the binary representation of the data

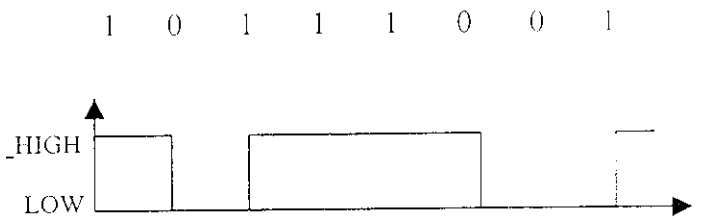
word. For instance if an 8-bit word has a value 01010101, the resulting voltage graph would appear as a series of four square waves with each negative half cycle. It can be seen that data circuits must provide a bandwidth for the data transmissions that they carry. This will be governed by the pulse rate variations and even a single square wave occupies a frequency range because of harmonics present.

When data is sent over telephone channels, the speed must be limited to ensure that the bandwidth required by the data transmission will not exceed the telephone channel bandwidth. The faster the data transmitted, the greater the bandwidth will need to be accommodating it.

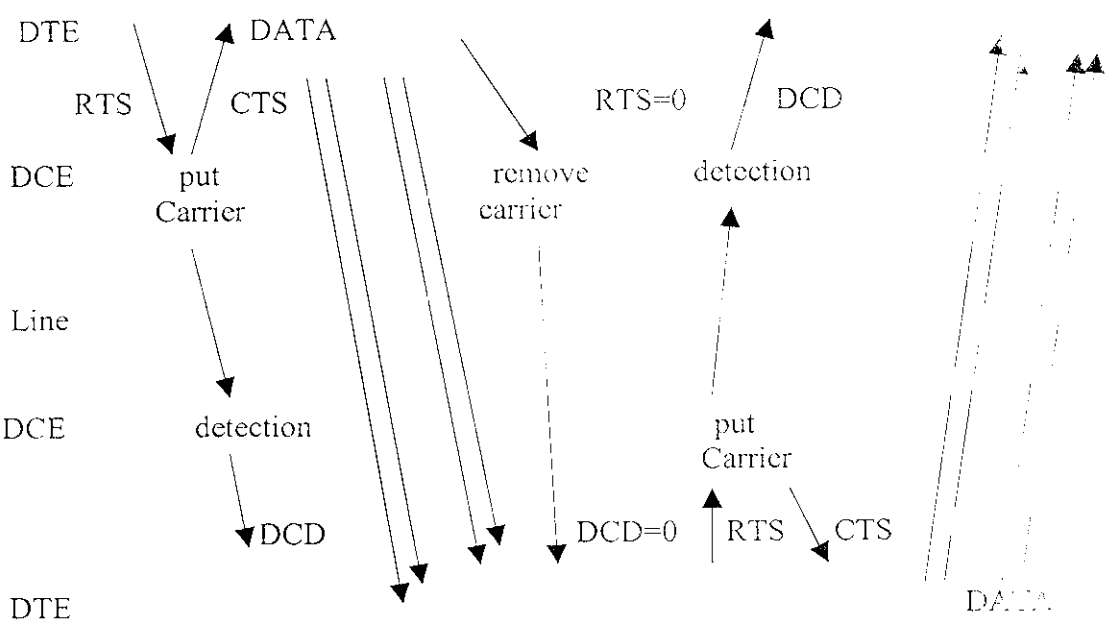
DATA TRANSMISSION SPEEDS

The rate of data transfer depends on several aspects of the transmission channel of which signaling speed is very important. Transmission Engineers often refer to the transmission speed of a communication channel as the channel baud rate. The baud is an important unit of signaling speed. In a system in which all pulses have equal duration, the speed in bauds is equal to the maximum rate at which signal pulses are transmitted. This should be recognized as different from the information bit rate. In a system which uses only one information bit per signaling pulse i.e., a binary system, the baud rate and the bit rate happen to be the same. In systems which encode the data in such a way that more than one information bit can be placed on each signaling pulse, the information bit rate will exceed the baud rate. To relate baud rate to bandwidth, the observations of Nyquist are used. Nyquist determined

BIT SEQUENCE



SEQUENCE FOR CONNECTING TWO DTEs



that one cycle of a transmission could contain a maximum of two bauds. The result is that maximum signaling speed in bauds is equal to twice the bandwidth of the channel. This is theoretical however, and could be achieved only in an ideal channel, which has no noise or distortion.

As indicated above the baud is a unit of signaling speed, but information transfer can occur at a rate equal to or different from the baud rate. Encoded data elements can be used to provide information transfer rates at speeds greater than the baud rate.

REQUIREMENTS OF COMMUNICATION SYSTEMS LINKING COMPUTER

The essential requirements of a complete data Communication system are as follows.

The physical connector that holds wires and mates with other wires and chassis must be compatible.

The voltages or currents used must be the same, so that the circuitry can handle them without damage and what values represent binary 0 and binary 1 should be known. The format or code used to represent each character must be agreed, since the system has been using just binary or digital values. Usually **ASCII** codes (**A**merican Standard **C**ode for **I**nformation **I**nterchange) is used. It represents each symbol or character by a field of 7 or 8 letters. The less commonly used code is **EBCDIC** (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode). It uses 8 bits to represent a character.

Serial communications are of two types -synchronous and asynchronous. The synchronous serial data communication scheme has the characteristics of single data path, moderate to high performance. It has a cost variation of moderate to high. It is mainly used in computer to computer, telephone and central office inter communication.

Serial asynchronous communication is characterized by single data path with low performance and low cost. Uses are computer printer and computer to telephone line communication.

SERIAL DATA COMMUNICATION

Communication plays a vital role in computer systems while exchange of information takes place between various modules. This information exchange could take place either serially or parallely. In serial communication, information is transmitted bit by bit and in parallel, it is transmitted byte(s) by byte(s). For long distance communication, cost considerations vote for serial mode of data transmission.

REASONS FOR PREFERENCE

When data has to be transmitted over long distances it is not practical to use bit - parallel transmission because too many wires would be needed which will increase the cost and complexity of transmission. The following attributes above all say the reasons for preferring serial communication

For long-distance communication we prefer to use bit-serial transmission. For distances over 50 feet, the cost of running multiple data lines become prohibitive. The extra logic required to convert parallel data to serial data for transmission over a single data line, and then to reconstitute the parallel byte at the destination, is less expensive than the hardware required to effect a parallel transfer over long distances. It may be noted that in high-speed parallel communication bits transmitted at equal intervals may not reach the receiver with the same equal spacing on each line due to the presence of noise. This can cause synchronizing problems, which implies that it might not always be possible to faithfully reproduce the transmitted signal at the receiving end. Therefore, when the distance between two points in a communication link is more than roughly 50 feet one would prefer to use bit - serial transmission instead of bit - parallel transmission.

Due to presence of slow devices we use bit serial transmission. In bit-parallel transmission all the bits, are transmitted at the same time through multiple wires, hence the transmission takes place at a much faster rate. But we cannot increase the speed of transmission arbitrarily. When we are using low speed devices, it is not possible to use bit-parallel transmission and we are bound to use bit-serial transmission.

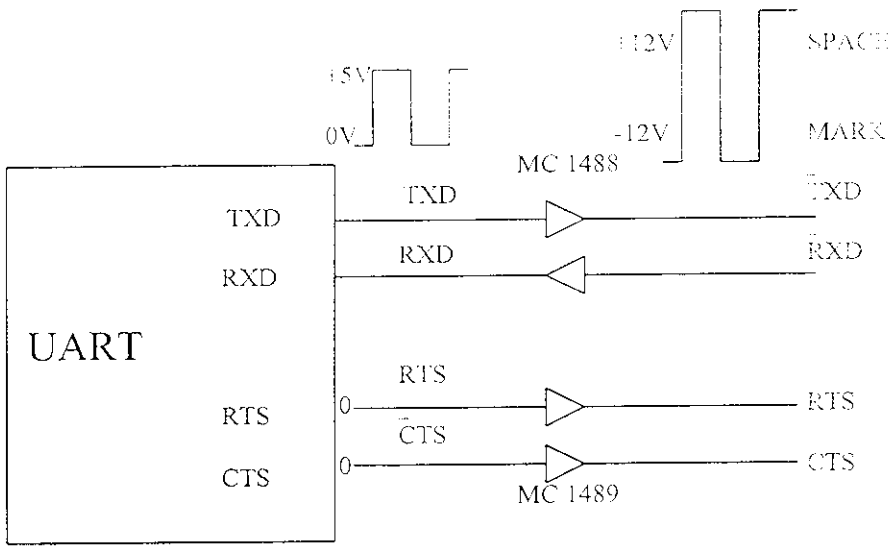
When telephone network is considered to be a communication channel, bit - serial transmission should be used. Bit-Parallel transmission cannot be used because that requires 8 wires or more than 8 wires, but the telephone network use only a pair of wires.

The main problem that must be solved for two devices to successfully communicate over a serial link is mutual synchronization. The transmitter and receiver must coordinate their operation well enough to ensure that bits sent are sampled and read as bits received.

TYPICAL INTERFACE BETWEEN DTE AND RS 232C LINE

RS - 232C like data transfers are usually asynchronous with a start bit (**SPACE**) and one or two stop bits (**MARK**). Microprocessor controllers called **UART** (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) serialize the information. They work between O and +5V; interface circuits which inverts the electrical levels create the RS-232C +12V and –12V. MC 1488 is used to convert TTL signal to RS - 232 signal and MC 1489 is used to convert RS-232 signal to TTL signal.

The two data transfer lines are called **TXD** (**T**ransmit **D**ata) and **RXD** (**R**eceive **D**ata). There is an inconsistency associated with the mnemonic names of these two lines; since the data bit values on the lines are inverted (logical 0 is +12 V), these mnemonics should be inverted and an inverting circle shown on the side of the line. Due to the fact that the stop bit is a 1 level, and the start bit is a low state, the line looks like an active high line with a steady state at "Zero".



RS-232C drivers
and Rectifiers.

TYPICAL INTERFACE BETWEEN A UART AND THE RS-232C LINE

The start bit is usually considered as an active signal and nothing in the notation shows that the data bits are inverted on the line. We will be more precise in our documentation and write TXD', RXD' to show that the data signals on the line are considered as non-inverted signals, but transfer inverted data information. On the serial interface chip, the data is called. as all manufacturers do, TXD and RXD. The control signals RTS and CTS generally use correct naming conventions; they are inverted on the **UART** pins (active low) and usually shown as much.

PROTOCOL

Two types of protocols should be considered in a data communication environment: terminal protocols and data link protocols.

TERMINAL PROTOCOL

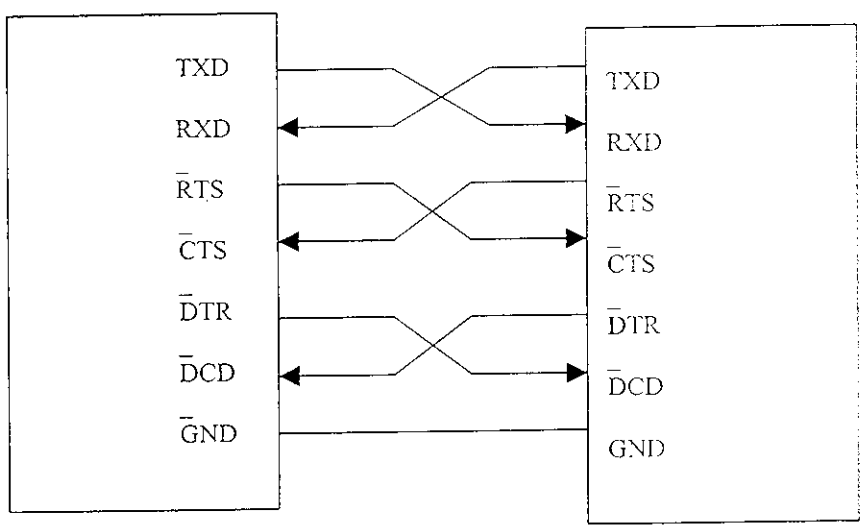
A protocol is an agreed set of rules, required between a DTE and a DCE in order to transmit digital data synchronously or asynchronously.

The protocol between a DTE and a DCE is known as terminal protocol. Handshake signals or control signals are required to maintain terminal protocol. Some commonly used hand shake signals are as follows: -

DTR (**D**ata **T**erminal **R**eady):

After the terminal is turned on and terminal runs any self-checks, this signal is asserted to tell the modem it is ready.

NULL MODEM CONNECTING TWO RS – 232C DTE DEVICES



DTE DTE

DSR (**D**ata **S**et **R**eady):

It is asserted when the modem is ready to transmit or receive data.

RTS (**R**equest **T**o **S**end):

Asserted by the terminal when it is ready to send a character.

DCD (**D**ata **C**arrier **D**etected):

Asserted by modem to indicate that it has established contact with computer.

CTS (**C**lear **T**o **S**end):

Indication of the DCE to the DTE that DCE is ready (Clear) to send data to DTE.

RI (**R**ing **I**ndicator):

Indication of the DCE (Modem) to the DTE terminal that a ring, indicator signal is received.

SERIAL COMMUNICATION ALGORITHM: -

ALGORITHM FOR TRANSMITTING DATA SERIALLY: -

1. Start
2. Create a transmitting end file text_file.
3. Open the text_file.
4. Accept the message to be sent character - by - character and store that in the text_file.
5. After the EOF character is encountered the file is closed.
6. When the message is to be transmitted, the file text_file is again opened.
7. Take a character from the file text_file.
8. If the character is the EOF character, go to 11.
9. Otherwise, send the character to the serial port of the computer.
10. Go to step 7.
11. Close the text_file and stop the operation.

ALGORITHM FOR RECEIVING SERIAL DATA

1. Start.
2. Create a receive file RX_file.
3. Receive a character from the serial port of the computer.
4. If the character is EOF character go to step 7
5. Otherwise open the RX_file .
6. Store the character in it.
7. Close the RX_file.
8. Stop.

VHDL - An Overview

A new possibility, which has changed the world of electronic designers, is the way in which several thousand gates and flip-flops can be programmed as an IC circuit in just a few minutes on a single PC without the need for expensive equipment. This means that it is possible to generate a file automatically from VHDL for programming circuits. This method is called rapid prototyping. At present there are circuits with up to 1,00,000 gates in a single circuit. Rapid prototyping can also be used for small production series. Two important reasons for using VHDL instead of traditional schematic design are shorter development times for electronic design and simple maintenance.

The code for a VHDL component can be verified functionally in a simulator. The simulator Simulates ("executes") the VHDL code with input signals and produces a signal diagram and error messages on the basis of the components. The input signals are defined either in VHDL or in the simulator's language. When the VHDL code is simulated, functional verification takes place. At a later stage, time verification of the design is also possible.

VHDL supports the development environment for digital design. VHDL also supports various development methods, top-down, bottom-up or any mix. **VHDL** is an acronym of VHSIC **H**ardware **D**escription **L**anguage and **VHSIC** is an acronym of Very **H**igh **S**peed **I**ntegrated **C**ircuit.

SYNTHESIS

Using synthesis software means that the designer avoids having to translate, minimize and meet time constraints from VHDL code. Synthesis is defined for the following different classes.

Logic Synthesis - Translates and minimizes Boolean functions only into gates.

RTL Synthesis - Same as logic synthesis but also translates sequential language constructions into gates and flip flops (state machines) .

Behavioral Synthesis - Can reuse one hardware component for more than one parallel sequential language construction.

SIMULATION

Simulating models is an effective way of verifying the design. The model in the computer is only a time-discrete model, however while reality is continuous.

The computer model is more or less like reality. It is least like reality at a high abstraction level (behavioral) and most like it at the lowest level (layout).

ABSTRACTION LEVELS

A computer model has several refinements which are better than a physical prototype. Setting the worst case for process parameters and temperature range provides better verification than building a prototype. With a prototype, only the prototype is verified.

The development flow from specification is a prototype and can be divided up as follows.

The **ANALYSIS PHASE** consists of writing a specification. The specification can be written in VHDL or ordinary language. This can be described in VHDL and then verified in a VHDL simulator.

The **DESIGN PHASE** means transforming the specification into an architecture and VHDL code. It is not yet possible to do this automatically. The phase starts with defining an architecture (block diagram). When the architecture is ready, the VHDL code is written for the various components (blocks) or ready-made components copied from a library. Now the function of the design is verified in a simulator.

The next stage is **TECHNOLOGY MAPPING**. It is parameters such as price, performance and supply etc., that determine which technology will be selected. An approved synthesis produces a technology-dependent net list (Schematic) which is an input file for other tools.

A prototype is then built and compared with the specification. If the RESULT is same as the SPECIFICATION the circuit is ready. This comparison is called **VALIDATION**.

DESIGN ENTITY

The term design entity may intuitively be perceived as a model for the chip. Associated with each design entity is a unique entity declaration that establishes the design entity's communication links to the outside world. These interface links are referred to as **ports** in VHDL.

Though symbolized as a digital chip, the design entity may in fact represent any device that possesses some form of intercommunication characteristic. It could model a single gate, a Central Processing Unit (CPU),a board populated by discretes, a theoretical protocol format, or even a complete system. In the latter case, the entity declaration might possibly not even have any interfaces at all. Instead, the design entity will merely represent a top level self- contained system that can be simulated without any interactions to other design entity models.

ARCHITECTURE BODY

Associated with each design entity is an architecture body that describes the behavior and/or the structure of a modeled device. The behavior denotes the simulatable functionality of the design entity and structure indicates the decomposition of a design entity into subcomponents.

DEVELOPMENT FLOW:-

The development phase for the product is one of the firsts in the product lifecycle. During this phase the product is specified, designed and verified. One common development model is called the ***WATERFALL*** model. It starts with specification and leads in defined stages to a functioning prototype. The waterfall model is ideal for well written specifications.

-- SECRET KEY ENCRYPTION AND DECRYPTION OF DATA

```
library ieee;
use ieee.std_logic_1164.all;
use work.numeric_std.all;
```

-- SHIFT REGISTER MODULE

```
entity shift is
port(
        clk,p,sel   :      in std_logic;
        d_in,i      :      buffer std_logic;
        d           :      out std_logic;
       e            :     out std_logic
    );
end;
```

--ARCHITECTURE DEFENITION

```
architecture rtl of shift is

        signal q1: std_logic_vector(7 downto 0);

      begin
            process(clk,p)
      begin
```

--LOAD DATA AT RISING EDGE

```
 if
        clk'event and clk='1'
then
```

```
q1(0)<=d_in;

q1(7)<=q1(6);

q1(6)<=q1(5);

q1(5)<=q1(4);

 i   <=q1(5);

q1(4)<=q1(3);

q1(3)<=q1(2);

q1(2)<=q1(1);

q1(1)<=q1(0);
```

--SHIFT REGISTER INPUT

d_in<=(q1(5) xor q1(0)) or (not (((((q1(0) or q1(1))or q1(2)) or q1(3))or q1(4)) or q1(5))));

-- ENCRYPTION MODE SELECT

if sel = '1' **then**

--ENCRYPT DATA

```
e<=(p xor q1(5));
```

```
-- DECRYPTION MODE SELECT

    elsif sel = '0' then

-- DECRYPT DATA

        d<=(p xor i);

        end if;

    end if;

end process;

end rtl;
```

ADVANTAGES OF VHDL PROGRAMMING

VHDL has the following advantages with regards to discrete circuit synthesis.

1. It is a standard medium for inter changing digital design information.

2. It is a standard pathway for reimplementing electronics parts.

3. It is a standard approach to determine which commercial off the shelf parts to use.

4. It has reusable hardware description language (HDL) Software Components .

5. It provides an electronic communication medium in the procurement cycle.

6. It has portable simulation models for digital systems.

7. It is an efficient technique to replace obsolete electronic components.

In a static limiting analyzer or simulator, it is possible to verify / simulate different timing cases such as WORST CASE, TYPICAL CASE and BEST CASE.

SYSTEM ENHANCEMENT

No system is complete unless it is has the power of compatibility with many different media. Our project is consummate with the power to alter all the different modules to suit the needs of the end user.

The transmission media can be chosen to be either of optical nature, which provides an even higher level of security, or if the prohibitive nature of the various components with regard to optical communication are to be taken into consideration, the user can go in for a coaxial cable which is much more feasible economically than the optical fiber.

Last but nor least, the chip encryption itself can be modified on the basis of the modulo-2 addition input. Changing the PN sequence shift input to the modulo-2 adder can change the encrypted output using which a specified amount of bits can be made to be received as they are for the purpose of synchronization between end users.

ADVANTAGES OF OUR SYSTEM: -

1. Decoding of the circuit from the chip is impossible as it was programmed using VHDL.

2. Our system comprises of a PN generator key, which gives an encrypted output which resembles noise making it difficult to be deciphered.

3. Our hardware key comprises of a complex encryption/ decryption model in a compact form.

4. As we are going for hardware implementation of the Crypto circuit, only those who have the key can access the original information.

5. Deciphering the actual message transmitted from the cipher-text is very difficult as encrypted output is very random and is time varying.

6. Our system is based on a self-improvised non-

Standardized expression making it a very reliable approach.

7. Possible use of optical communication makes our system immune to channel tapping.

8. One of the major advantages of our system is that it has been designed for PC-PC communication but does not required much of an effort to standardize for networks.

9. Our hardware enabler is in close proximity to "one time pad" which itself portrays its high efficiency level.

APPLICATIONS: -

1.Military applications: -

In defense applications, as strategic information regarding defense operations are invaluable, this compels the usage of crypto-system.

2. Business applications: -

Important information regarding business ventures should be communicated secretively.

3.Networking environment: -

In a networking environment, communicating between two terminals secretly, together with other benefits of a network makes crypto-system a necessity.

4. Government applications: -

All government level transactions require a high level of secrecy.

5.Applications in banking sector: -

Possible use of cryptography in credit cards reduces embezzlement of funds using duplicate cards.

6. Applications in printing media:-

On-the-spot receivers of sensational events should reach the publishing house before any other dailies can interpret it.

7. Applications in investigating business: -

Passing secret information regarding a highly influential culprit model in criminal activities from one branch to another, requires utmost security.

8. Application in marketing field: -

Proper conveyance of marketing concepts between advertising agencies and industries necessitates implementation of crypto-systems.

9. Key propagation: -

Can be used for propagation of encryption keys of existing algorithms.

CONCLUSION: -

Data encryption is a technique, no longer the exclusive preserve of the Military for use in their cloak and dagger operations. Nowadays any information if improperly used can lead to disastrous consequences. Hence encryption has become an integral part of any data transmission process. Encryption has its boons but can create problem with overhead times and memory management. Different techniques are involved in the process of encryption but they all dependant on system consideration and speed of nodal transmission required when concerned with networks.

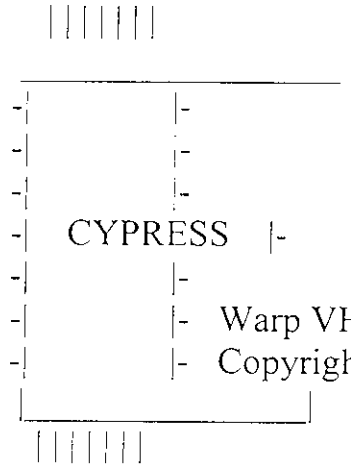
As far as cryptography is concerned, pseudorandom noise has been glorified as the ultimate in encryption techniques as it is very close to the perfect model - the one time keypad. Our PN sequence encryption technique successfully eliminates key holding, memory and concession time over head. It provides a very high degree of encryption secrecy to meet the requirements of modern

security needs. This, when reinforced with the programming power of VHDL Language impacts another facet to secrecy of the system. Both these techniques when combined with appropriate digital data communication systems can provide enhanced encryption qualities with lesser constraints on key holding and memory considerations.

```
/*PROGRAM TO SEND THROUGH PORT*/
#include <stdio.h>
#include <dos.h>
#include <io.h>
main( )
{
        FILE *fp;
        int cc;
        fp = fopen("adc.c","r");
        clrscr();
        while(feof(fp)== 0)
        {
        cc = fgetc(fp);
        printf("%c",cc);
        outportb(0x3f8,cc);
/*      sleep(1);*/
        }
        fclose(fp);
}
```

```
/*PROGRAM TO RECEIVE THROUGH PORT*/
#include <stdio.h>
#include <dos.h>
main()
{
        FILE *f;
        int c,count=0;
        f = fopen("rec.c","w");
        clrscr();
        while(!kbhit())
        {
        c=inportb(0x3f8);
        fprintf(f,"%c",c);
        count++;
        if (count ==75)
        {
        fprintf(f,"\n");
        count=0;
        }
        printf("%c",c);
        sleep(1);
        }
fclose(f);
}
```

REPORT AND SPECIFICATIONS



Warp VHDL Synthesis Compiler: Version 4 IR x66
Copyright (C) 1991, 1992, 1993,
1994, 1995, 1996 Cypress Semiconductor

==========

Compiling: encrypt.vhd
Options: -q -e10 -w10 -o2 -ygs -fP -v10 -yb -yp -dc22v10 -pPALC22V10-20PC/PI encrypt.vhd

==========

C:\WARP\BIN\VHDLFE.EXE V4 IR x66: VHDL parser
Thu Mar 04 09:10:08 1999

Library 'work' => directory 'lc22v10'
Library 'ieee' => directory 'C:\warp\lib\ieee\work'
Using 'C:\warp\lib\ieee\work\stdlogic.vif'.
Using 'C:\warp\lib\common\stdlogic\mod_genu.vif'.

C:\WARP\BIN\VHDLFE.EXE: No errors.

C:\WARP\BIN\TOVIF.EXE V4 IR x66: High-level synthesis
Thu Mar 04 09:10:12 1999

C:\WARP\BIN\TOVIF.EXE: No errors.

C:\WARP\BIN\TOPLD.EXE V4 IR x66: Synthesis and optimization
Thu Mar 04 09:10:16 1999

Detecting unused logic.

User names
q1_7
q1_6

Deleted 2 User equations/components.
Deleted 2 Synthesized equations/components.

Alias Detection

Aliased 0 equations, 7 wires.

Circuit simplification

Circuit simplification results:

Expanded 0 signals.
Turned 0 signals into soft nodes.
Maximum expansion cost was set at 10.

Created 27 PLD nodes.

C:\WARP\BIN\TOPLD.EXE: No errors.

PLD Optimizer Software: DSGNOPT.EXE 17/JUL/96 [v3.22] 4 IR x66

DESIGN HEADER INFORMATION (09:10:19)

Input File(s): encrypt.pla
Device : C22V10
Package : PALC22V10-20PC/PI
ReportFile : encrypt.rpt

Program Controls:
None.

Signal Requests:
GROUP USEPOL ALL

Completed Successfully

PLD Optimizer Software: DSGNOPT.EXE 17/JUL/96 [v3.22] 4 IR x66

OPTIMIZATION OPTIONS (09:10:19)

Messages:
Information: Process virtual 'eD' ... expanded.
Information: Process virtual 'dD' ... expanded.
Information: Process virtual 'd_inD' ... expanded.
Information: Process virtual 'q1_1' ... converted to NODE.
Information: Process virtual 'q1_2' ... converted to NODE.
Information: Process virtual 'q1_3' ... converted to NODE.
Information: Process virtual 'q1_4' ... converted to NODE.
Information: Process virtual 'q1_5' ... converted to NODE.
Information: Process virtual 'q1_0' ... converted to NODE.
Information: Optimizing logic using best output polarity for signals:
e.D d.D d_in.D

Information: Selected logic optimization OFF for signals:
q1_1.D q1_1.C q1_2.D q1_2.C q1_3.D q1_3.C q1_4.D q1_4.C q1_5.D q1_5.C
q1_0.D q1_0.C e.C d.C i.D i.C d_in.C

Summary:
Error Count = 0 Warning Count = 0

Completed Successfully

PLD Optimizer Software: MINOPT.EXE 17/JUL/96 [v3.22] 4 IR x66

LOGIC MINIMIZATION (09:10:20)

Messages:

Summary:
Error Count = 0 Warning Count = 0

Completed Successfully

PLD Optimizer Software: DSGNOPT.EXE 17/JUL/96 [v3.22] 4 IR x66

OPTIMIZATION OPTIONS (09:10:20)

Messages:
Information: Optimizing Banked Preset/Reset requirements.

Summary:
Error Count = 0 Warning Count = 0

Completed Successfully

PLD Compiler Software: PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

DESIGN EQUATIONS (09:10:21)

i.D =
    q1_5.Q

i.AR =
    GND

i.SP =
    GND

i.C =
    clk

q1_0.D =
    d_in.Q

q1_0.AR =
    GND

q1_0.SP =
    GND

q1_0.C =
    clk

q1_5.D =
    q1_4.Q

q1_5.AR =
    GND

q1_5.SP =
    GND

q1_5.C =
    clk

q1_4.D =
    q1_3.Q

q1_4.AR =
GND

q1_4.SP =
GND

q1_4.C =
clk

q1_3.D =
q1_2.Q

q1_3.AR =
GND

q1_3.SP =
GND

q1_3.C =
clk

q1_2.D =
q1_1.Q

q1_2.AR =
GND

q1_2.SP =
GND

q1_2.C =
clk

q1_1.D =
q1_0.Q

q1_1.AR =
GND

q1_1.SP =

```
    GND

q1_1.C =
    clk

d_in.D =
    /q1_5.Q * /q1_4.Q * /q1_3.Q * /q1_2.Q * /q1_1.Q
  + /q1_0.Q * q1_5.Q
  + q1_0.Q * /q1_5.Q

d_in.AR =
    GND

d_in.SP =
    GND

d_in.C =
    clk

d.D =
    p * /sel * /i.Q
  + /p * /sel * i.Q
  + sel * d.Q

d.AR =
    GND

d.SP =
    GND

d.C =
    clk

e.D =
    q1_5.Q * /p * sel
  + /q1_5.Q * p * sel
  + /sel * e.Q

e.AR =
    GND
```

e.SP =
GND

e.C =
clk

Completed Successfully

PLD Compiler Software: PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

DESIGN RULE CHECK (09:10:21)

Messages:
None.

Summary:
Error Count = 0 Warning Count = 0

Completed Successfully

PLD Compiler Software: PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

DESIGN SIGNAL PLACEMENT (09:10:22)

Messages:
Information: Checking for duplicate NODE logic.
None.

C22V10

clk =| 1|	|24|* not used
sel =| 2|	|23|= d
p =| 3|	|22|= q1_1
not used *| 4|	|21|= q1_3
not used *| 5|	|20|= q1_5
not used *| 6|	|19|= i
not used *| 7|	|18|= q1_0
not used *| 8|	|17|= q1_4
not used *| 9|	|16|= q1_2
not used *|10|	|15|= d_in
not used *|11|	|14|= e
not used *|12|	|13|* not used

Summary:

Error Count = 0 Warning Count = 0

Completed Successfully

PLD Compiler Software: PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

RESOURCE ALLOCATION (09:10:22)

Information: Macrocell Utilization.

Description	Used	Max
Dedicated Inputs	2	11
Clock/Inputs	1	1
I/O Macrocells	10	10

13 / 22 = 59 %

Information: Output Logic Product Term Utilization.

Node#	Output Signal Name	Used	Max
14	e	3	8
15	d_in	3	10
16	q1_2	1	12
17	q1_4	1	14
18	q1_0	1	16
19	i	1	16
20	q1_5	1	14
21	q1_3	1	12
22	q1_1	1	10
23	d	3	8
25	Unused	0	1

16 / 121 = 13 %

Completed Successfully

PLD Compiler Software: PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

JEDEC ASSEMBLE (09:10:22)

Messages:
Information: Output file 'encrypt.jed' created.

Summary:
Error Count = 0 Warning Count = 0

Completed Successfully at 09:10:22

Cypress C22V10 Jedec Fuse File: encrypt.jed

This file was created on 03/04/99 at 17:55:48
by PLA2JED.EXE 17/JUL/96 [v3.22] 4 IR x66

22V10*
QP24* Number of Pins*
QF5828* Number of Fuses*
F0* Note: Default fuse setting 0*
G0* Note: Security bit Unprogrammed*
NOTE DEVICE C22V10*
NOTE PACKAGE PALC22V10-20PC/PI*
NOTE PINS clk:1 sel:2 p:3 e:14 d_in:15 q1_2:16 q1_4:17 q1_0:18 i:19
q1_5:20 *
NOTE PINS q1_3:21 q1_1:22 d:23 *
NOTE NODES *
L00000
00000000000000000000000000000000000000000000
* Node clk[1] => BANK : 1 *

L00044
11111111111111111111111111111111111111111111
11111011011111111101111111111111111111111111
11111011101111111110111111111111111111111111
11100111111111111111111111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
* Node d[23] => OE : 1 ,LOGIC : 8 *

L00440
00000000000000000000000000000000000000000000
11111111111111111111101111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
* Node q1_1[22] => OE : 1 ,LOGIC : 10 *

L00924
00000000000000000000000000000000000000000000
11111111111111111111111111111110111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
* Node q1_3[21] => OE : 1 ,LOGIC : 12 *

L01496
00000000000000000000000000000000000000000000
11111111111111111111111111110111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000

* Node q1_5[20] => OE : 1 ,LOGIC : 14 *

L02156
11111111111111111111111111111111111111111111111
11111111111111101111111111111111111111111111111
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
* Node i[19] => OE : 1 ,LOGIC : 16 *

L02904
00000000000000000000000000000000000000000000000
11111111111111111111111111111111111101111111111
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000

* Node q1_0[18] => OE : 1 ,LOGIC : 16 *

L03652
0000000000000000000000000000000000000000
1111111111101111111111111111111111111111
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
* Node q1_4[17] => OE : 1 ,LOGIC : 14 *

L04312
0000000000000000000000000000000000000000
1111111011111111111111111111111111111111
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
* Node q1_2[16] => OE : 1 ,LOGIC : 12 *

L04884
1111111111111111111111111111111111111111
1111110111011101111111111101110111111111
1111111111111110111111101111111111111111

1111111111111011111110111111111111111111
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
* Node d_in[15] => OE : 1 ,LOGIC : 10 *

L05368
1111111111111111111111111111111111111111
1111011110111110111111111111111111111111
1111011101111101111111111111111111111111
1111101111111111111111111111111111101111
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
* Node e[14] => OE : 1 ,LOGIC : 8 *

L05764
0000000000000000000000000000000000000000
* Node sel[2] => BANK : 1 *

L05808
10* Note: 23 *

L05810
10* Note: 22 *

L05812
10* Note: 21 *

L05814
10* Note: 20 *

L05816
10* Note: 19 *

L05818
10* Note: 18 *

L05820
10* Note: 17 *

L05822
10* Note: 16 *

L05824
10* Note: 15 *

L05826
10* Note: 14 *

C6A13* Note: Fuse Checksum*
95

NOVA Simulation Printout

File: ENCRYPT.psd - View: e

Printout produced: Wed Mar 03 20:34:05 1999

```
    2 1
   1 3 4 3 2


    c d e p s
    l | | | e
    k | | | l
    | | | | |
^^^^^^^^^^^^^^^^^^^
```

0: 1 L L 1 1
1: 0 L L 1 1
2: 1 L L 1 1
3: 0 L H 1 1
4: 1 L H 1 1
5: 0 L H 1 1
6: 1 L H 1 1
7: 0 L H 1 1
8: 1 L H 1 1
9: 0 L H 1 1
10: 1 L H 1 1
11: 0 L H 1 1
12: 1 L H 1 1
13: 0 L H 1 1
14: 1 L H 1 1
15: 0 L L 1 1
16: 1 L L 1 1
17: 0 L L 1 1
18: 1 L L 1 1
19: 0 L L 1 1
20: 1 L L 1 1

21: 0 L L 1 1
22: 1 L L 1 1
23: 0 L L 1 1
24: 1 L L 1 1
25: 0 L L 1 1
26: 1 L L 1 1
27: 0 L L 1 1
28: 1 L L 1 1
29: 0 L H 1 1
30: 1 L H 1 1
31: 0 L H 1 1
32: 1 L H 1 1
33: 0 L L 1 1
34: 1 L L 1 1
35: 0 L L 1 1
36: 1 L L 1 1
37: 0 L H 1 1
38: 1 L H 1 1
39: 0 L H 1 1
40: 1 L H 1 1
41: 0 L L 1 1
42: 1 L L 1 1
43: 0 L H 1 1
44: 1 L H 1 1
45: 0 L L 1 1
46: 1 L L 1 1
47: 0 L L 1 1
48: 1 L L 1 1
49: 0 L H 1 1
50: 1 L H 0 1
51: 0 L L 0 1
52: 1 L L 0 1
53: 0 L L 0 1
54: 1 L L 0 1
55: 0 L L 0 1
56: 1 L L 0 1
57: 0 L L 0 1
58: 1 L L 0 1
59: 0 L H 0 1
60: 1 L H 0 1

61: 0 L H 0 1
62: 1 L H 0 1
63: 0 L H 0 1
64: 1 L H 0 1
65: 0 L L 0 1
66: 1 L L 0 1
67: 0 L H 0 1
68: 1 L H 0 1
69: 0 L L 0 1
70: 1 L L 0 1
71: 0 L H 0 1
72: 1 L H 0 1
73: 0 L H 0 1
74: 1 L H 0 1
75: 0 L L 0 1
76: 1 L L 0 1
77: 0 L L 0 1
78: 1 L L 0 1
79: 0 L L 0 1
80: 1 L L 0 1
81: 0 L H 0 1
82: 1 L H 0 1
83: 0 L L 0 1
84: 1 L L 0 1
85: 0 L L 0 1
86: 1 L L 0 1
87: 0 L H 0 1
88: 1 L H 0 1
89: 0 L L 0 1
90: 1 L L 0 1
91: 0 L H 0 1
92: 1 L H 0 1
93: 0 L L 0 1
94: 1 L L 0 1
95: 0 L L 0 1
96: 1 L L 0 1
97: 0 L L 0 1
98: 1 L L 0 1
99: 0 L L 0 1
100: 1 L L 0 1

101: 0 L H 0 1
102: 1 L H 0 1
103: 0 L L 0 1
104: 1 L L 0 1
105: 0 L L 0 1
106: 1 L L 0 1
107: 0 L L 0 1
108: 1 L L 0 1
109: 0 L L 0 1
110: 1 L L 0 1
111: 0 L L 0 1
112: 1 L L 0 1
113: 0 L L 0 1
114: 1 L L 0 1
115: 0 L H 0 1
116: 1 L H 0 1
117: 0 L H 0 1
118: 1 L H 0 1
119: 0 L H 0 1
120: 1 L H 0 1
121: 0 L H 0 1
122: 1 L H 0 1
123: 0 L H 0 1
124: 1 L H 0 1
125: 0 L H 0 1
126: 1 L H 0 1
127: 0 L H 0 1
128: 1 L H 0 1
129: 0 L L 0 1
130: 1 L L 0 1
131: 0 L L 0 1
132: 1 L L 0 1
133: 0 L H 0 1
134: 1 L H 0 1
135: 0 L H 0 1
136: 1 L H 0 1
137: 0 L L 0 1
138: 1 L L 0 1
139: 0 L L 0 1
140: 1 L L 0 1

141: 0 L H 0 1
142: 1 L H 0 1
143: 0 L L 0 1
144: 1 L L 0 1
145: 0 L H 0 1
146: 1 L H 0 1
147: 0 L H 0 1
148: 1 L H 0 1
149: 0 L L 0 1
150: 1 L L 0 1
151: 0 L H 0 1
152: 1 L H 0 1
153: 0 L L 0 1
154: 1 L L 0 1
155: 0 L L 0 1
156: 1 L L 0 1
157: 0 L L 0 1
158: 1 L L 0 1
159: 0 L H 0 1
160: 1 L H 0 1
161: 0 L H 0 1
162: 1 L H 0 1
163: 0 L H 0 1
164: 1 L H 0 1
165: 0 L L 0 1
166: 1 L L 0 1
167: 0 L H 0 1
168: 1 L H 0 1
169: 0 L L 0 1
170: 1 L L 0 1
171: 0 L H 0 1
172: 1 L H 0 1
173: 0 L H 0 1
174: 1 L H 0 1
175: 0 L L 0 1
176: 1 L L 0 1
177: 0 L L 0 1
178: 1 L L 0 1
179: 0 L L 0 1
180: 1 L L 0 1

181: 0 L H 0 1
182: 1 L H 0 1
183: 0 L L 0 1
184: 1 L L 0 1
185: 0 L L 0 1
186: 1 L L 0 1
187: 0 L H 0 1
188: 1 L H 0 1
189: 0 L L 0 1
190: 1 L L 0 1
191: 0 L H 0 1
192: 1 L H 0 1
193: 0 L L 0 1
194: 1 L L 0 1
195: 0 L L 0 1
196: 1 L L 0 1
197: 0 L L 0 1
198: 1 L L 0 1
199: 0 L L 0 1
200: 1 L L 0 1
201: 0 L H 0 1
202: 1 L H 0 1
203: 0 L L 0 1
204: 1 L L 0 1
205: 0 L L 0 1
206: 1 L L 0 1
207: 0 L L 0 1
208: 1 L L 0 1
209: 0 L L 0 1
210: 1 L L 0 1
211: 0 L L 0 1
212: 1 L L 0 1
213: 0 L L 0 1
214: 1 L L 0 1
215: 0 L H 0 1
216: 1 L H 0 1
217: 0 L H 0 1
218: 1 L H 0 1
219: 0 L H 0 1
220: 1 L H 0 1

221: 0 L H 0 1
222: 1 L H 0 1
223: 0 L H 0 1
224: 1 L H 0 1
225: 0 L H 0 1
226: 1 L H 0 1
227: 0 L H 0 1
228: 1 L H 0 1
229: 0 L L 0 1
230: 1 L L 0 1
231: 0 L L 0 1
232: 1 L L 0 1
233: 0 L H 0 1
234: 1 L H 0 1
235: 0 L H 0 1
236: 1 L H 0 1
237: 0 L L 0 1
238: 1 L L 0 1
239: 0 L L 0 1
240: 1 L L 0 1
241: 0 L H 0 1
242: 1 L H 0 1
243: 0 L L 0 1
244: 1 L L 0 1
245: 0 L H 0 1
246: 1 L H 0 1
247: 0 L H 0 1
248: 1 L H 0 1
249: 0 L L 0 1
250: 1 L L 0 1
251: 0 L H 0 1
252: 1 L H 0 1
253: 0 L L 0 1
254: 1 L L 0 1
255: 0 L L 0 1

C
N
q1_0_streg
18 1
N
q1_1
9999 11
N
q1_1_streg
22 1
N
q1_2
9999 11
N
q1_2_streg
16 1
N
q1_3
9999 11
N
q1_3_streg
21 1
N
q1_4
9999 11
N
q1_4_streg
17 1
N
q1_5
9999 11
N
q1_5_streg
20 1
N
sel
9999 11
V

PINS ONLY
19 4 1
clk d d_in e i p q1_0 9001 q1_1 9003 q1_2 9005 q1_3 9007 q1_4 9009
q1_5 9011 sel
V

PINS and REGS
19 4 1
clk d d_in e i p q1_0 9001 q1_1 9003 q1_2 9005 q1_3 9007 q1_4 9009
q1_5 9011 sel
V

e
5 4 1
clk d e p sel
T
0
S
256
256 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1
clk
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
000000000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
2
sel
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
000000000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

9999
3
p
FFFFFFFFFFFFC00000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
14
e
1FFE00078798601F99E0619806001FFF87867981F99E0619806001FFF87
86798
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
15
d_in
7FFE1E19E607E6781866018007FFE1E19E607E6781866018007FFE1E19
E607E6
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
16
q1_2
0000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

9999
17
q1_4
0000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
18
q1_0
0000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
19
i
0001FFF87867981F99E0619806001FFF87867981F99E0619806001FFF878
6798
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
20
q1_5
0000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

9999
21
q1_3
0000000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
22
q1_1
0000000000000000000000000000000000000000000000000000000000000000
000
0000000000000000000000000000000000000000000000000000000000000000
000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
23
d
0000000000000000000000000000000000000000000000000000000000000000
000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999
25
jed_node25
0000000000000000000000000000000000000000000000000000000000000000
000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

9999
26
jed_node26
00000000000000000000000000000000000000000000000000000000000
000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9999

BIBLIOGRAPHY

1. **Cryptography and Secure Communications**

   by *Man Young Rhee.*

2. **Computer Communication**

   by *Andrew S. Tenenbaum*

3. **Electronic Communication Systems**

   by *Kennedy & George*

4. **VHDL – An Understanding**

   by *Stephen Sjoholm*

5. **VHDL for Researchers and Scientists**

   by *Joseph Pick*

6. **IMPACT SERIES on Serial Communication**

   by *IIT Education Cell*

7. **EFY Magazines**

8. **Spread Spectrum Scene Magazines**

9. **Digital Communication**

   by *Simon Haykin*

10. **THE INTERNET**