THE UNIVERSITY OF HULL

Public Key Cryptosystems

-Theory Application and Implementation

being a Thesis submitted for the degree

Doctor of Philosophy

in the University of Hull

by

Anthony Joseph McAuley, BSc

April 1985

To my Mother and Father,

who given the same opportunity...

21 JUN 1985

SUMMARY

Summary of Thesis submitted for PhD degree

by Anthony Joseph McAuley

on

Public Key Cryptosystems

The determination of an individual's right to privacy is mainly a nontechnical matter, but the pragmatics of providing it is the central concern of the cryptographer. This thesis has sought answers to some of the outstanding issues in cryptography. In particalar, some of the theoretical, application and implementation problems associated with a Public Key Cryptosystem (PKC).

The Trapdoor Knapsack (TK) PKC is capable of fast throughput, but suffers from serious disadvantages. In chapter two a more general approach to the TK-PKC is described, showing how the public key size can be significantly reduced. To overcome the security limitations a new trapdoor was described in chapter three. It is based on transformations between the radix and residue number systems.

Chapter four considers how cryptography can best be applied to multi-addressed packets of information. We show how security or communication network structure can be used to advantage, then proposing a new broadcast cryptosystem, which is more generally applicable.

Copyright is traditionally used to protect the publisher

from the pirate. Chapter five shows how to protect information when in easily copyable digital format.

Chapter six describes the potential and pitfalls of VLSI, followed in chapter seven by a model for comparing the cost and performance of VLSI architectures. Chapter eight deals with novel architectures for all the basic arithmetic operations. These architectures provide a basic vocabulary of low complexity VLSI arithmetic structures for a wide range of applications.

The design of a VLSI device, the Advanced Cipher Processor (ACP), to implement the RSA algorithm is described in chapter nine. It´s heart is the modular exponential unit, which is a synthesis of the architectures in chapter eight. The ACP is capable of a throughput of 50 000 bits per second.

PREFACE

PREFACE.

Until fairly recently, the science of keeping information secret remained almost totally unknown outside the military and diplomatic communities. Even today most people only think of the romantic work of the codebreaker of cryptanalyst, whose efforts during WWII provided such a vital source of intelligence to the Allies. But in recent years there is increasing interest in the reciprocal science of the codemaker or cryptographer. This thesis is concerned with the latter; a challenging and exciting area of research.

Cryptography unites the diverse fields of electronics, pure mathematics and computer science. Although this thesis is written primarily for the electronics engineer, it is necessary to call on areas outside their usual knowledge. Therefore, Chapter 1, as well as introducing the basic cryptographic techniques, will provide an introduction to Galois fields, number theory and computational complexity. Those familiar with cryptography might still find it useful to familiarize themselves with the notations used in later chapters.

Cryptography has been revolutionized by two developments in the last ten years. Firstly, the concept of the Public Key Cryptosystem (PKC), which greatly enhances the capabilities offered. Secondly, the pervasive technology of Very Large Scale Integration (VLSI), which gives substance and impetus to these new ideas. The thesis covers the author's original work in the

design, application and implementation of cryptography to business and commercial telecommunication systems. A digest of these chapters is given below.

CHAPTER 2. Describes a generalization of the original Merkle-Hellman trapdoor knapsack PKC. A new scheme is proposed which allows reduced redundancy and a smaller public key directory.

CHAPTER 3. Puts forward a new trapdoor for the general knapsack PKC. It is based on the transformations between radix and residue number systems. The security does not rely on transformations from a superincreasing sequence, which have proven vulnerable to cryptanalysis.

CHAPTER 4. Describes how cryptography can efficiently be applied to information intended for more than one receiver. It shows how the natural security or communications network structure can be exploited, together with a new concept: the broadcast cryptosystem. The latter allows a message to be deciphered by more than one receiver, if reduced, security can be tolerated.

CHAPTER 5. Describes how cryptography can be applied to enforce copyright protection on a wide range of published material. The bases of the proposed solutions are the combination of tamper resistant modules and a PKC.

CHAPTER 6. Looks at the whole design cycle for a custom VLSI chip. It draws attention to the particularly important aspects the author has found in such a design.

CHAPTER 7. Describes a comprehensive model for designing

VLSI architectures. This model is intended to be adaptable to new technologies.

CHAPTER 8. Describes some new VLSI architectures for realizing the basic arithmetic functions used in digital signal processing. It concentrates on maximizing their asymptotic performance and minimizing design complexity.

CHAPTER 9. Describes the implementation of the RSA-PKC in CMOS. The finished design is a 24-pin DIL package, with Multibus interface.

CHAPTER 10. Draws conclusions from previous chapters and makes forecasts for the future.

One important point, which has not changed over the centuries, is that security is very difficult to gauge. The interpretations drawn should be read with a spark of scepticism. For in one hundred years, many interpretations may be regarded as erroneous.

The author wishes to thank all those who directly or indirectly contributed to this thesis. In particular, I am grateful to my supervisor, Dr R. Goodman, who has been a constant source of ideas and encouragement. I would also like to extend my appreciation to my colleages at Hirst Research Centre for their guideance on the VLSI apsect of my research. Especially to Mike Rome, Andrew McCabe and Nick Parker.

CONTENTS.

CONTENTS

CHAPTER 1

1.1 INTRODUCTION.

Many people think they could invent a perfect cipher. But a brief look through the history of cryptography (Ref.101) teaches caution. Though many systems have served their purpose for a brief period of time, the cryptanalyst has always proven too resourceful. However with the aid of cheap hardware, increased sophistication and more open discussion, it is now possible to talk of a cipher that will remain secure for a very long time.

This chapter introduces the basic techniques which are helping to achieve greater confidence in cryptography. It is aimed at covering the salient features which relate to subsequent chapters.

No attempt is made to cover the whole spectrum of modern cryptography. It only scratches the surface of important areas such as: secret sharing, standards, analogue techniques and cryptanalytic methods.

The thesis assumes an understanding of the recent advances in cryptography. Those unfamiliar with these developments are referred to one of the many excellent digests available in the literature (Refs.102-111).

1.2 TERMINOLOGY AND MATHEMATICAL NOTATION.

The problem of understanding a highly specialized subject is always hindered by the use of words used only in its limited field. Hopefully most such words will be explained when they are used, but the following brief list might be useful.

AUTHENTICITY: Determine the integrity of data and transmitter.

BREAKABLE SYSTEM: A cryptosystem where it is possible to cryptanalyse the ciphertext.

CIPHERTEXT: Secret form of a message.

CRYPTOLOGY: The science of transmission security, encompassing both cryptography and cryptanalysis.

CRYPTOGRAPHY: Science of rendering information intelligible only to a legitimate receiver.

CRYPTANALYSIS: Science of the techniques used to decipher ciphertexts, without the secret key.

CRYPTOSYSTEM: Method of hiding information, using a known algorithm and secret key.

ENCIPHER: To convert a message from plaintext to ciphertext.

KEY: Information needed to encipher/decipher a message/ciphertext.

MESSAGE/PLAINTEXT: Information in readable form.

SECURITY: Computational feasiblity of deciphering a message, without the secret key.

A list of abbreviations used in the following chapters is given at the end the thesis. In addition, a number of non-standard mathematical shorthands will be used. These are listed below.

$$l(n) = \begin{pmatrix} 1 \text{ if } n = 0, \text{ else} \\ \\ \text{biggest integer less than or equal to } \log n \end{pmatrix}$$

A xor B = A.B´ + A´.B

A >= B. A is greater than or equal to B.

A <= B. A is less than or equal to B.

A <> B. A not equal to B.

A >> B. A is very much greater than B.

A << B. A is very much less than B.

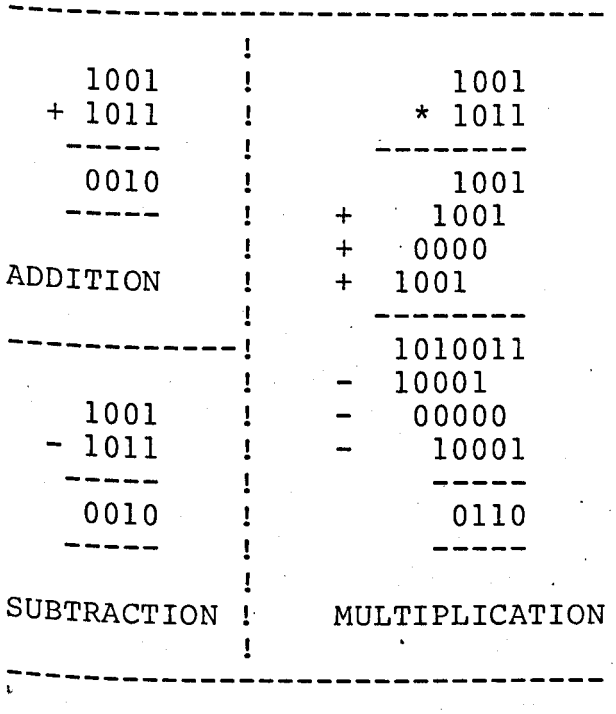
$A \doteq B$. A is approximately equal to B.

1.3 MATHEMATICAL BACKGROUND.

A cryptosystem can be thought of as a set of transformations, from the set of possible messages into the set of possible ciphertexts. Each transformation will correspond to encipherment with a particular key. For cryptographic applications these transformations must be injective and ideally bijective. In the PKCs considered here, these transformations are defined by simple arithmetic functions.

In order to keep the size of the cryptosystem within bounds, the arithmetic is either done in the Galois fields GF(2^n), GF(p) or the ring R(q): where p is a prime and q is the product of primes. This has the added advantage of making cryptanalysis more difficult, since it both reduces the amount of information and introduces an extra element of diffusion. More information on these algebraic systems can be obtained in the literature (Refs.112-113).

Arithmetic in GF(2^n) is always done mod p(x): where p(x) is an irreducible polynomial of degree n. Figure 1.1a gives an example of how two numbers can be added and multiplied. Arithmetic in R(n) is done mod n; figure 1.1b gives an example of how the same operations are done here.



FIGRE 1.1a ARITHMETIC IN GF(2^4), WITH IRREDUCIBLE P(X) = 10001.

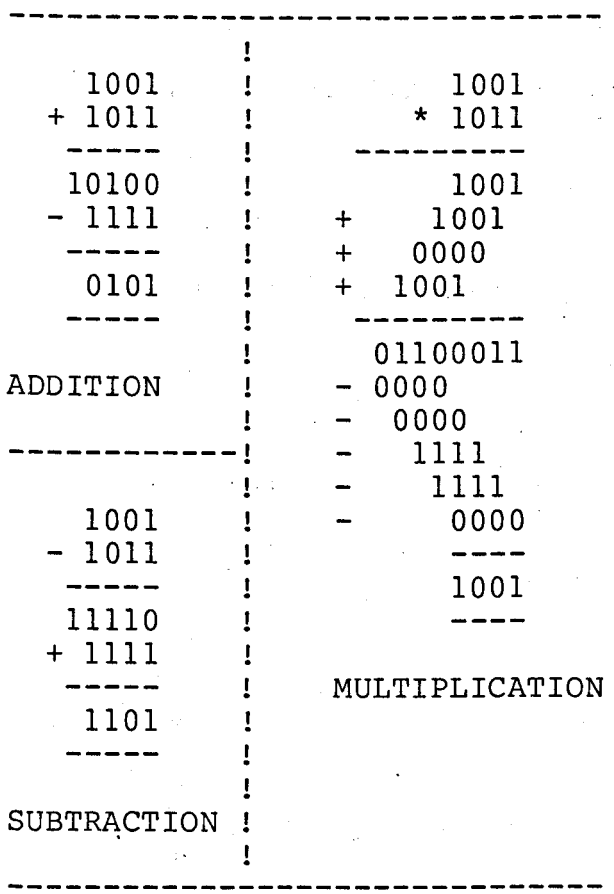


FIGURE 1.1b ARITHMETIC IN R(15).

An important difference between the two systems is that in GF(2^n) there are no carries. This allows faster and simpler implementations, but does reduce the desirable element of confusion. Because of this latter point, systems based on GF(2^n) will not be considered here.

1.3.1 Some properties of a ring of integers.

In the ring R(n) only integers between 0 and n-1 are allowed. If the results are larger or smaller than this, then the answer is found by reducing modulo n.

Division is not strictly a ring operation, since only

integer results are allowed. The equivalent ring operation is called inversion. For a given number (x), this involves finding another number (y) with the property:

$x * y = 1 \bmod n$

This operation is not possible if x is a factor of n.

For the ring R(n), a key parameter is the Euler totient function U(n). This is the number of integers which are relatively prime to n. Let n be the product of k primes:

$$n = \prod_{i=1}^{k} p_i \text{ ,then}$$

$$U(n) = \prod_{i=1}^{k} (p_i - 1)$$

For an integer x, it can be shown that:

$$x^{U(n)} \bmod n = 1, \text{ if GCD } (x,n) = 1 \qquad \ldots 1.1$$

If $x * y = 1 \bmod n$

$$\text{then } y = x^{U(n) - 1} \qquad \ldots 1.2$$

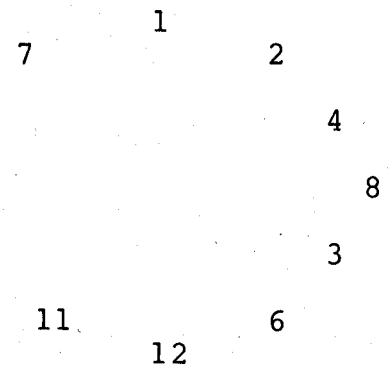


FIGURE 1.2 RING R(13).

An interesting way of looking at the ring R(13) is shown in figure 1.2. Each number, going clockwise round the circle, is twice (mod 13) the previous number. From this diagram we see:

U(13) = 12, the size of the circle.

$2^{3} = 8$, at three o'clock.

$2^{12} = 2^{U(13)} = 1$, at twelve o'clock (equation 1.1).

$2^{11} = 2^{U(13)-1} = 7$, where 2 * 7 = 1 (equation 1.2)

A more computationally efficient method of finding the inverse is to use Euclid's extended algorithm (Ref.114). Inverses always exist if GCD (x,n) = 1.

1.3.2 Residue arithmetic.

In its residue form a number in R(n) is represented by its remainders, when divided by the factors of n. For a number x let:

$$x_i^{(i)} = x \bmod p_i$$

Then by the Chinese Remainder Theorem (Ref.115):

$$x \leftrightarrow x^{(1)}, x^{(2)}, \ldots, x^{(k)}$$

is a bijective mapping. That is the transformation is one-to-one for all x's between 1 and n-1.

The ring R(n) can be represented in radix or residue

form: the two forms being homomorphic. The radix form is the 'normal' way of representing a number. However the residue form has a number of advantages.

In the residue form arithmetic can easily be split up, allowing greater parallelism. In chapter 3 we shall show how the residue form of a number can be useful for cryptography. Figure 1.3 shows how the basic arithmetic operations are performed, in both radix and residue number systems.

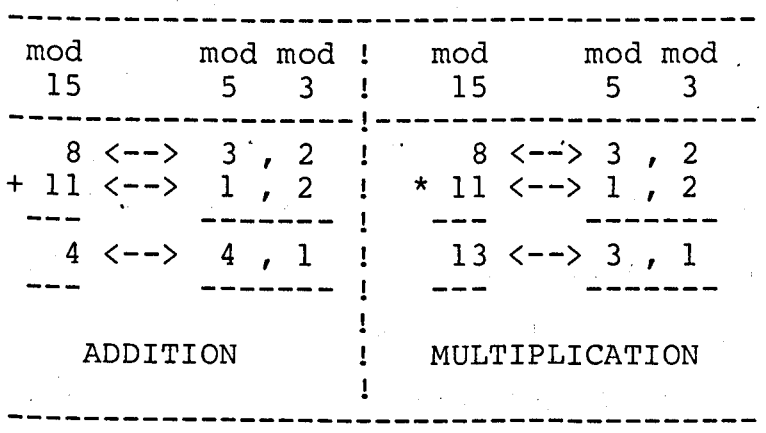


FIGURE 1.3 ARITHMETIC IN RADIX AND RESIDUE FORM.

1.4 CRYPTANALYSIS AND COMPUTATIONAL COMPLEXITY.

Cryptanalysis is concerned with finding an optimum way of breaking a cipher. This is usually expressed in terms of the computational complexity of the algorithm used to solve it.

Complexity is measured in terms of time and area: where area is the number of processors and memory used (Ref.115).

1.4.1 Measure of performance.

How can we measure the goodness of one algorithm or architecture against another? A useful shorthand is the 'big O' notation. An algorithm f(n) is said to be of order O(g(n)) if there exists some constant c such that f(n) < c(g(n)). For example the function:

$8n^7 + 3n^3 + \log n$, is $O(n^7)$ and the function

$7n.\log n + 5n + 9$, is $O(n.\log n)$

An algorithm of higher order of magnitude might have a smaller constant of proportionality. In which case, the higher order algorithm might be superior for slower or smaller machines. However the asymptotic complexity is a good measure, and one that becomes increasingly important as technology advances.

The use of 'order statistics' allows a designer to concentrate on the important issues. For example a 32-bit microprocessor might be very much faster than an 8-bit one, but an algorithm O(n) for one machine will be O(n) on the other.

1.4.2 NP problems.

Algorithms of O(k^n) have an explosive rate of growth. There is general agreement that problems requiring an algorithm with this exponential growth are intractable (Ref.115). A problem of this form is known as a non-deterministic polynomial problem:

denoted by NP.

There are a class of problems which are NP-complete. That is, if one problem can be solved in polynomial time, they all can (Ref.115). The solution of an NP problem is often the target which a cryptographer tries to give a cryptanalyst. But it is important to realize that NP-complete refers to only the hardest instances of a problem.

1.4.3 Cryptographically secure.

The definition of cryptographic security might be that it is impossible to cryptanalyse a message. However this is unnecessarily restrictive; indeed only the one time pad posseses this property (Ref.101). Consider a computer operating:

1. Using every atom (10^100) in the universe as a Processing Element.
2. Each Processing Element operating in the time required by light to traverse a nuclear radius (10^-23 seconds).
3. Been operating since the beginning of the universe (4 * 10^10 years).

Using this rather conservative model, the computer could still not have finished an algorithm requiring 2^500 steps. For cryptographic intractablity, a figure of 2^128 is sufficient.

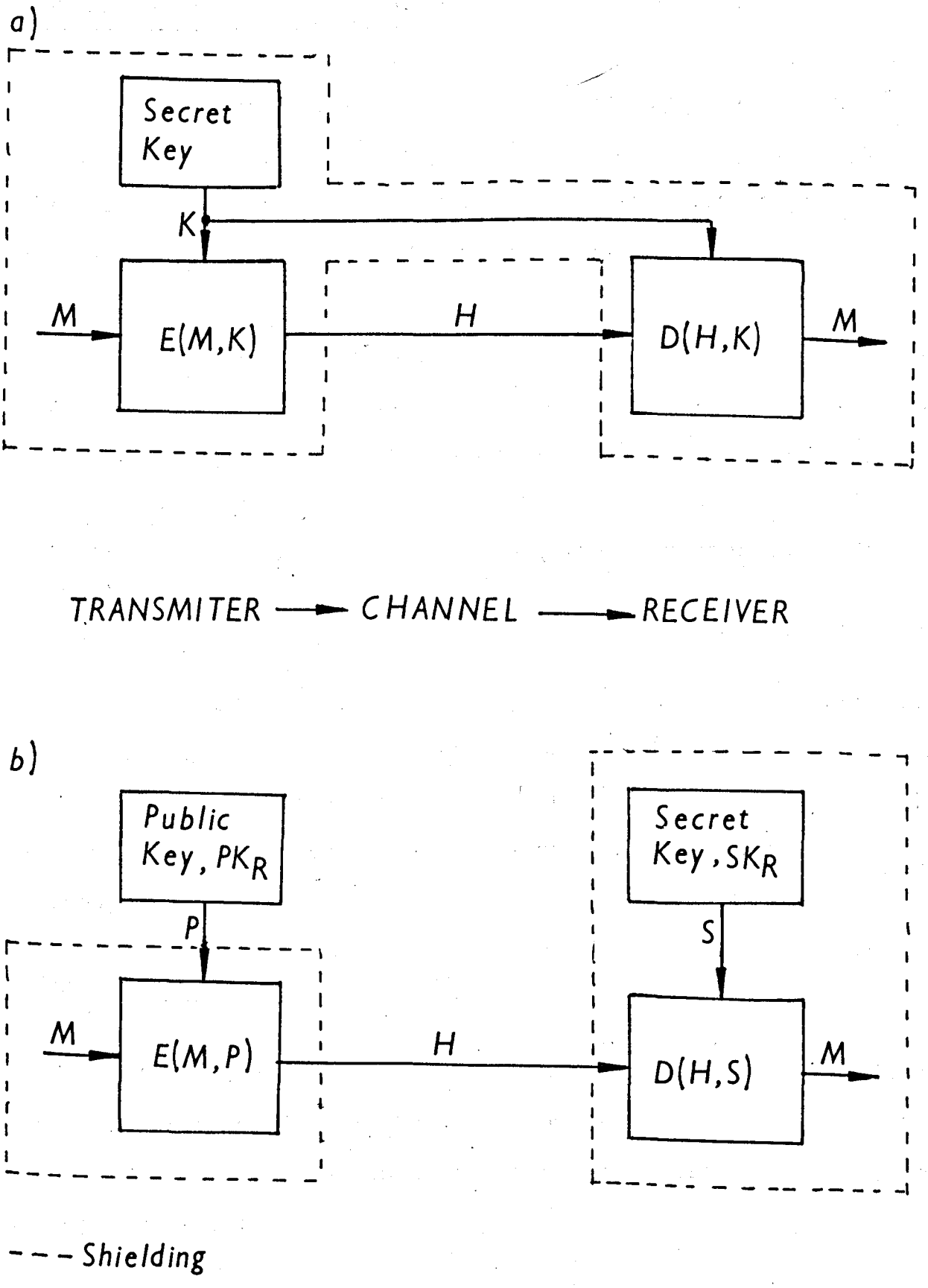


FIGURE 1.4 ONE AND TWO KEY CRYPTOSYSTEMS.

1.5 CRYPTOGRAPHIC TECHNIQUES.

In modern cryptography, the security of a transmission lies in keeping the key secure, rather than the whole system. This not only allows greater confidence in a cryptosystem, but permits the adoption of standards (Ref.111).

In a conventional cryptosystem, such as the US Data Encryption Standard (DES), the keys used for encryption and decryption are the same (see figure 1.4a). However such a system leads to a key distribution problem. A network with 2000 users would require over 2 million keys to allow each user to talk to every other.

A recent scheme proposed by Diffie and Hellman (Ref.116) overcomes the key distribution problem. It does so by using a different key for encryption and decryption (see figure 1.4b). This use of asymmetric keys has become known as a Public Key Cryptosystem (PKC).

1.5.1 A Public Key Cryptosystem.

The PKC derives its name from the fact that it publishes the encryption key (Public Key, PK), but keeps the decryption key hidden (Secret Key, SK). To help understand this principle, an analogy is useful.

A room is full of English speaking people, none of whom

speak French. The Engish to French half of a language dictionary is freely available; but someone has destroyed all but one copy of the French to English half. In this scenario anyone can write a message in French. However only the person who has the French to English dictionary could decipher these messages.

In the above analogy the English to French dictionary represents the encryption function and the French to English dictionary the decryption function. It is possible to break the system by searching the English to French dictionary. But in a good cryptosystem the analogous operation would be too computationally demanding.

Encryption and decryption can be expressed mathematically by equations 1.3 and 1.4.

$$\text{Sender, S: } e(M1, PK_R) = C \qquad \ldots 1.3$$

$$\text{Receiver, R: } d(C1, SK_R) = M1 \qquad \ldots 1.4$$

Where e(X,K) denotes encryption and d(X,K) decryption of a message X under a key K. The two functions are not necessarily different.

A PKC is not only able to offer WRITE ONLY communication with easy key distribution. The real power of PKC stems froms its ability to offer READ ONLY communication.

If the sender transforms a message M2 under his own

secret key, anyone can look up the public key to recover M2. But since only the sender has the secret key, only he could generate a message 'deciphered' under it. This operation, which is equivalent to a written signature, is described by equations 1.5 and 1.6.

$$\text{Sender, } S: d(M2, SK_S) = C2 \qquad \ldots 1.5$$

$$\text{Receiver, } R: e(C2, PK_S) = M2 \qquad \ldots 1.6$$

It is possible to combine read and write only protection:

$$\text{Sender, } S: e(d(M3, SK_S), PK_R) = C3 \qquad \ldots 1.7$$

$$\text{Receiver, } R: d(e(C3, PK_S), SK_R) = M3 \qquad \ldots 1.8$$

Using the transformations described above it is possible to protect against both active and passive eavesdropping.

The position of cryptography in a communication system is shown in figure 1.5. Source coding must be done before the cryptosystem, since the cryptosystem hides the structure inherent in the English language. While error correction coding must be applied after the cryptosystem, because of the error propagation effects of good cryptosystem.

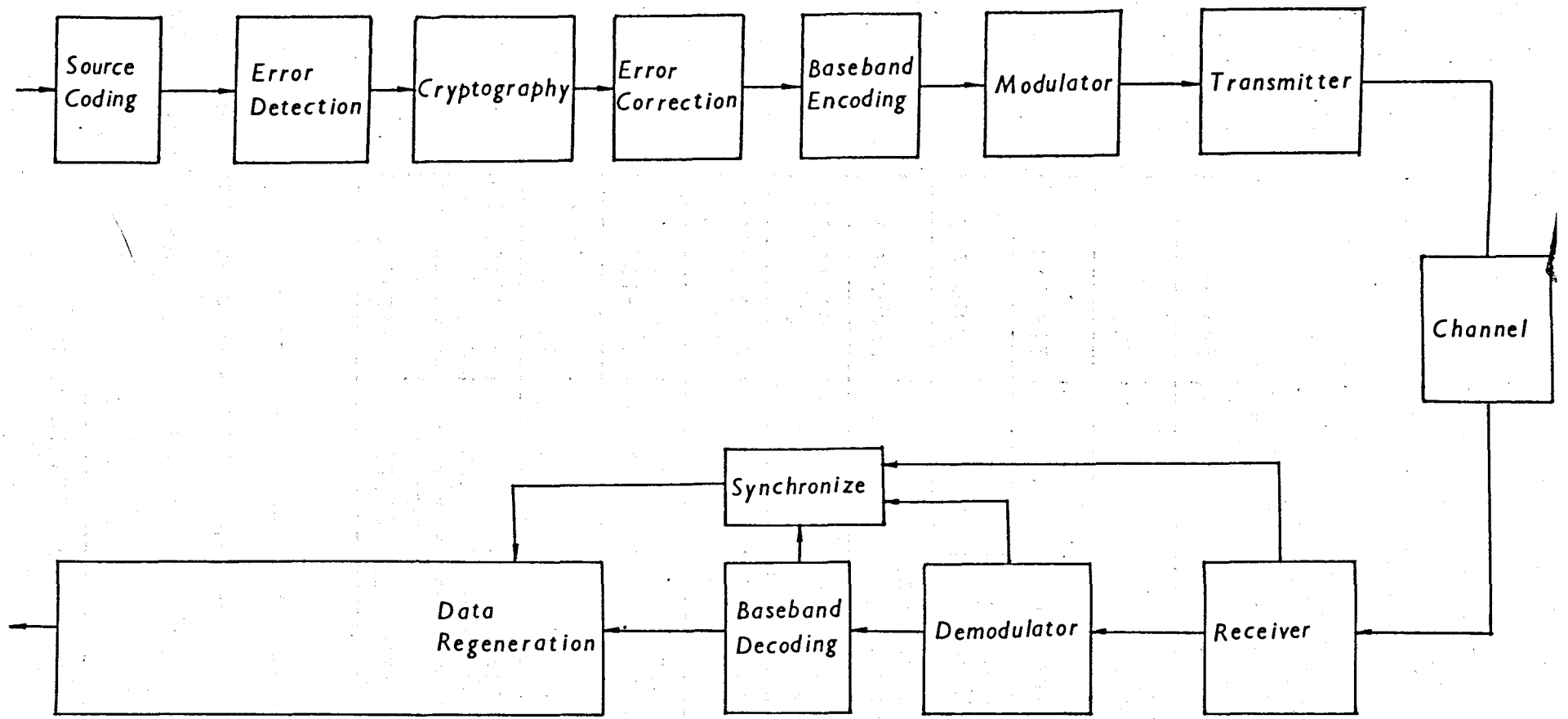


FIGURE 1.5 DIGITAL COMMUNICATION CHANNEL.

1.6 PRACTICAL PUBLIC KEY CRYPTOSYSTEMS.

The PKC is more than a nice pedagogical idea. Since its inception many practical schemes have been proposed. A summary of some of these are given in table 1.2.

Cryptosystem	Still Secure?	REFERENCES
DH key distribution scheme in GF(2^n)	NO	117 sec. 1.6.1
DH key distribution scheme in GF(p)	YES	116 sec. 1.6.1
RSA modular exponen. scheme in GF(2^n)	NO	121-122 sec. 1.6.3
RSA modular exponen. scheme in R(n)	YES	123 sec. 1.6.3
Binary knapsack (superinc. trapdoor)	?	126 sec. 1.6.2
General knapsack (superinc. trapdoor)	?	Chapter 2
General knapsack (residue trapdoor)	?	Chapter 3
Lu-Lee modular sum scheme	NO	118-120
Shamir knapsack signature scheme	NO	124
McEliece Goppa code scheme	YES	125

TABLE 1.2 PRACTICAL PKC´S

Most practical PKC's have proven weak under cryptanalysis. The security of the McEliece PKC is difficult to gauge, because it has received surprisingly little attention. However, it is still a valid possibilty as no successful cryptanalysis has been published. The other three schemes which have proven resistant are described below.

The notations R and S will be used in the examples to represent the receiver and sender respectively.

1.6.1 DH key distribution scheme.

The first and simplest PKC was proposed by Diffie and Hellman in their original paper (Ref.116). The DH key distribution scheme is based on two reverse transformations: exponentiation and logarithms. Their original proposal was for arithmetic in GF(p), but later unwisely extended to GF(2^n).

Suppose a Sender (S) and Receiver (R) wish to establish a secret key. First S calculates x^c, sending x and x^c to R. R then calculates x^d and returns this to S. Both users can now compute x^cd, which they can use as their key. If x, c and d have been chosen randomly, a cryptanalyst is faced with a very hard problem. He must find either log(x^c) or log(x^d), since he only possesses x^c, x^d, x.

At the present time the best algorithm for finding logs depends on which field is used. For GF(2^n), Coppersmith has

shown (Ref.117) how to break the scheme. However the method does not work in GF(p), where the security is much higher.

Example.

S: 3^4 mod 11 = 4

R: 3^7 mod 11 = 9

S: 9^4 mod 11 = 5

R: 4^7 mod 11 = 5

Giving a session key of 5.

1.6.2 MH trapdoor knapsack PKC.

Merkle and Hellman (Ref.126) intoduced a PKC based on the binary knapsack problem. This is a problem of the form: given a vector of n integers (the weight of each component) and the sum of a subset of these integers (the total weight), find which subset was used.

The hardest instances of the knapsack problem are known to be NP-complete. However for certain vectors (e.g. 1,2,4,8,16) the solution is trivial. The Merkle-Hellman scheme is based on using a superincreasing sequence, which has been disguised by modular multiplication (*W mod Z).

Someone wishing to transmit an n bit message X, would transmit the sum of those components whose corresponding message bit were 1. The publisher of A could then use his secret key (W and Z), to transform this back into the easy superincreasing

knapsack problem.

The non-bijective mapping not only causes message expansion, but makes authentication difficult. Also the method requires a very large public key size. Despite this the scheme initially proved very popular. The reason was the simplicity of encoding and decoding, which meant it could run at the same speed as conventional cryptosystems.

Recently the original Merkle-Hellman scheme has been broken. Even stronger variants, such as the Graham-Shamir system (Ref.204) and the iterated knapsack method, have proven vulnerable. Despite this nobody has yet shown a general method of cracking all knapsack problems. So it is still feasable that a fast PKC, based on the knapsack problem, can be found.

Example.

R: A´ = 8, 4, 2, 1

Z = 17, W = 11 --> 1/W = 14

A = 3, 10, 5, 11

S: Message = 1101

Ciphertext = 3 + 10 + 11 = 24

R: 24 * 14 mod 17 = 13 --> 1101

1.6.3 RSA modular exponential PKC.

The most elegant and powerful PKC is that due to Rivest,

Shamir and Adleman (RSA). It uses the exponentiation function for both the encryption and decryption transformations (Ref.123): with arithmetic done in the ring R(n).

The two exponents e and d are chosen to be inverses modulo U(n) (see section 1.3.1). This means we can write:

C = M^e mod n

M = C^d mod n

In the RSA-PKC each user publishes a unique e and n, but keeps their d secret. It is believed that the best way to find d, and hence cryptanalyse a message, is to factor n. However, if n is chosen to be the product of two large (say 256 bits) safe primes (Ref.127), factoring is too computationally demanding. Therefore the RSA-PKC is still regarded as secure.

Example.

R: n = 5 * 11 = 55

U(n) = (5-1).(11-1) = 40

e = 7, giving d = 23

S: Message = 1101 = 13

Ciphertext = 13^7 = 7 mod 55

R: Message = 7^23 = 13 mod 55

1.7 REFERENCES.

101 D.KAHN.
The codebreakers, the story of secret writing.
MACMILLAN, 1967.

102 IEE WORKING PARTY.
Privacy and security in civil telecommunications.
IEE, LONDON, SEP.1984.

103 J.A.GORDON.
Recent trends in cryptology.
ELEC. & POWER, VOL.26, NO.2, PP.162-165, FEB.1980.

104 W.DIFFIE & M.E.HELLMAN.
Privacy and authentication: An introduction to cryptography.
PROC. IEEE, VOL.67, PP.397-427, MAR.1979.

105 G.J.SIMMONS.
Cryptography: the mathematics of secure communication.
THE MATH. INTELLIGENCER, VOL.1, PP.233-246, JAN 1979.

106 M.E.HELLMAN.
An overview of public key cryptography.
IEEE COMM. SOC. MAG., VOL.16, PP.24-32, NOV.1978.

107 D.E.DENNING.
Cryptography and data security.
ADDISON WESLEY, 1982.

108 M.WILLETT.
A tutorial on public key cryptography.
COMPUTERS AND SECURITY, VOL.1, PP.72-79, 1982.

109 H.J.BEKER & F.PIPER.
Cipher systems: the protection of communications.
Northwood books, London, 1982.

110 W.PRICE.
Developments in data security.
NPL TECH. MEMO TTCC 14/83, OCT 1983.

111 W.L.PRICE.
Standards for data security, a status report.
ONLINE CONFERENCE "NETWORKS 84", 1984.

112 F.AYRES,JR.
Theory and problems of modern algebra.
MCGRAW-HILL, 1965.

113 G.BRKHOFF & T.C.BARTEE.
Modern applied algebra.
MCGRAW-HILL, 1970.

114 D.E.KNUTH.

The art of computer programming: vol.2, seminumerical algorithms.

ADDISON-WESLEY, SECOND EDITION, 1981.

115 A.V.AHO, J.E.HOPCROFT, J.D.ULLMAN.

The design and analysis of computer algorithms.

ADDISON-WESLEY, 1974.

116 W.DIFFIE & M.HELLMAN.

New directions in cryptography.

IEEE TRAN. INFO. THEORY, VOL.22, PP.644-654, NOV.1976.

117 D.COPPERSMITH.

Fast evaluation of logarithms in fields of characteristic two.

IEEE TRAN. INFO. THEORY, VOL.30, PP.587-594, JUL.1984.

118 S.C.LU & L.N.LEE.

A simple and effective public key cryptosystem.

COMSAT TECH. REV., VOL.9, NO.1, PP.15-24, 1979.

119 J.M.GOETHALS & C.COUVREUR.

A cryptanalytic attack on the Lu-Lee public key cryptosystem.

PHILLIPS J. RES., VOL.35, PP.301-306, 1980.

120 M.J.KOCHANSKI.

Remarks on Lu and Lee's proposal for a public key cryptosystem.

CRYPTOLOGIA, VOL.4, NO.4, PP.204-212, 1980.

121 D.W.KRAVITZ & I.S.REED.

Extension of the RSA crypto-structure: a Galois approach.

ELECTRONICS LETTERS, VOL.18, NO.6, PP.255-256, 18-MAR.1982.

122 P.DELSARTE & P.PIRET.

Comment on, extension of the RSA crypto-structure: a Galois approach.

ELECTRONICS LETTERS, VOL.18, NO.13, PP.582-583, 24-JUN.1982.

123 R.L.RIVEST, A.SHAMIR & L.ADLEMAN.

A method for obtaining digital signatures and public key cryptosystems.

COMM. OF ACM, VOL.21, NO.2, PP.120-126, FEB.1978.

124 A.M.ODLYZKO.

Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme.

IEEE TRAN. INFO. THEORY, VOL.30, PP.594-598, JUL.1984.

125 R.J.MCELIECE.

A public-key cryptosystim based on algebraic coding theory.

JPL DSN PROG. REPORT 42-44, PP.114-116, JAN.-FEB.1978.

126 R.C.MERKLE & M.E.HELLMAN.

Hiding information and signatures in trapdoor knapsacks.

IEEE TRAN. INFO. THEORY, VOL.24, PP.525-530, SEP.1978.

127 G.R.BLAKLEY & I.BOROSH.

Rivest-Shamir-Adleman public key cryptosystems do not always conceal messages.

COMP. & MATH. WITH APPLIC., VOL.5, PP.169-178, 1979.

CHAPTER 2

2.1 INTRODUCTION.

One of the most promising PKCs has been the Trapdoor Knapsack PKC (TK-PKC). The TK-PKC (see section 1.6.2) algorithm appeared to offer privacy and authentication (Ref.201) at high data rates (Ref.202). However it was apparent that, when compared with the RSA scheme (see section 1.6.3), this system had a number of drawbacks.

The most obvious problem with the TK-PKC is the size of the public key. At 80K bits per user, this would produce a very large public directory.

It was suggested in the original Merkle-Hellman paper, (Ref.126), that a possible means of reducing the key size was to use a non-binary knapsack problem. In this paper we shall build a model for this non binary TK-PKC and try to reduce the size of the public key.

Two other parameters are also considered in the design of this new system: the data expansion and security. The TK-PKC expands data by over 100%. This redundancy is unacceptable for many applications and we shall try to minimize it. More recent critisism of the TK-PKC has been on the security (Refs.202-206). In this chapter we shall just try to achieve a security equivalent to the original scheme. Improving security is subject of chapter 3.

2.2 NOTATION.

a_i = i´th published knapsack component.

a'_i = i´th secret knapsack component.

E = ratio of (no. message bits : no. ciphertext bits).

g = number of bits on $x_{i,max}$.

k = number of iterative modular multiplications.

m_i = i´th modulus in iterative modular multiplications.

n = number of knapsack components.

P = number of bits in the public key.

Q = number of bits in the secret key.

S = hard knapsack problem: ciphertext.

S´ = easy knapsack problem.

t = number of bits in transmitted ciphertext.

t´ = number of bits in message.

v = number of random bits in the lsbs of each x_i.

w_i = i´th multiplier in iterative modular multiplications.

x_i = g-bit message block.

y = number of random bits in a´.

z = number of bits in each knapsack component.

2.3 THE GENERAL METHOD.

A general knapsack problem is one in which you are given a vector of n integers, together with an integer S equal to their weighted sum. The problem is to find which weights were used. The less general binary knapsack problem is one in which the weights are restricted to 0 or 1. In our case we shall let the weights be integers within the limits:

$$0 <= x_i < 2^g$$

Though this general knapsack problem is known to be NP-complete (Ref.207) there do exist subsets which are easy to solve. One such case is when the knapsack components a´ are chosen such that:

$$a'_i > \left(\sum_{j=1}^{i-1} a'_j \right) * x_{max} \qquad ...2.1$$

Merkle and Hellman described a method of hiding this superincreasing sequence, using k iterations of modular multiplication. The resulting trapdoor knapsack is hard to solve without additional information. Each component is given by:

$$a_i = ((((a'_i * w_1) \bmod m_1) * \ldots) * w_k) \bmod m_k \qquad ...2.2$$

where $GCD(w_j, m_j) = 1$ ...2.3

$$2 < w_j < m_j \qquad ...2.4$$

$$m_1 > \sum_{i=1}^{n} a'_i$$

$$m_j > 2^{g+l(n)+1} * m_{j-1}$$

where $l(n) = 1$ if $n = 0$, else the smallest integer greater than or equal to log n.

If all the knapsack components were published, anyone wishing to transmit a message x could calculate:

$$S = \sum_{i=1}^{n} x_i * a_i \qquad \ldots 2.5$$

The receiver could transform this hard knapsack problem into:

$$S' = \sum_{i=1}^{n} x_i * a'_i$$

where:

$$S' = ((((\ S * w_k^{-1}\ ) \bmod m_k\ ) * \ldots\ ) * w_1^{-1}\ ) \bmod m_1$$

which can easily be deciphered (Ref.126).

2.4 EFFECT OF PARAMETER SELECTION.

The superincreasing knapsack components are defined according to the structure shown in figure 2.1. Thus the third component would have its most significant bit set to 1 followed by 2g-1 0's and finally y random bits (R3). We know the number of bits (z) in each component is given by:

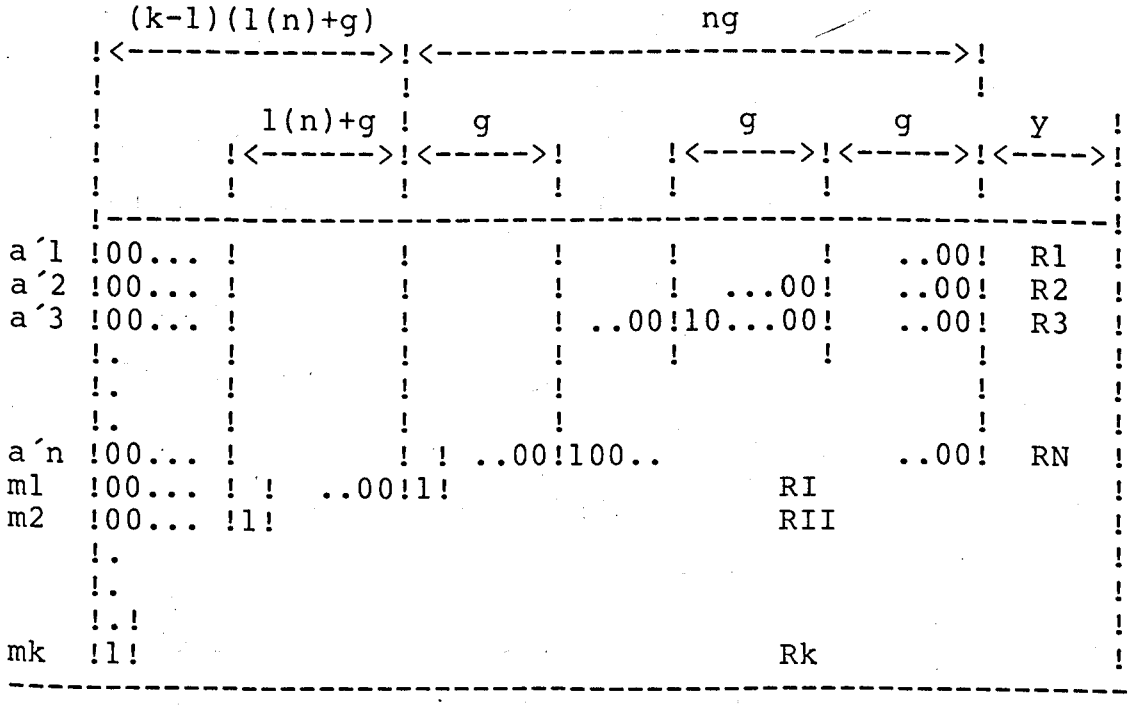


FIGURE 2.1 COMPONENT SELECTION IN A GENERAL TK-PKC

$$z = l\left(m_{k,max} \right)$$

From figure 2.1,

$$z = y + g.n + (k-1).(g + l(n)) \qquad ...2.6$$

From equation 2.5:

$$S_{max} = n * ((2^{z} - 1).(2^{g} - 1))$$

$$= n.(2^{z+g} - 2^{z} - 2^{g} + 1)$$

But $n = 2^{\log n}$. If $z >> g$. then the number of bits transmitted:

$$\dot{t} = z + g + l(n)$$

Substituting for z from equation 4, we obtain:

$$\dot{t} = y + g.n + k.(g + l(n)) \qquad ...2.7$$

But the number of actual message bits transmitted:

$$t' = g.n \qquad ...2.8$$

Therefore the ratio of message bits to ciphertext bits is:

$$E = 1 : 1 + k/n + ((1/n).g).(y + k.l(n)) \quad ...2.9$$

The number of bits in the secret key is given by:

$$Q = k.((\text{no. bits in } w_i) + (\text{no. bits in } m_i))$$

$$= 2.k.z$$

Therefore from equation 2.6:

$$Q = 2.k.(y + g.n + (k - 1).(g + l(n))) \quad ...2.10$$

The number of bits in the public key:

$$P = n.z$$

Therefore from equation 2.6:

$$P = n.(y + g.n + (k - 1).(g + l(n)) \quad ...2.11$$

The above results are summarized in table 1.

VAR.	ORDER OF MAGNITUDE n	g	y	k
E	1/n	1/g	y	k
Q	n	g	y	k^2
P	n^2	g	y	k

TABLE 1 EFFECT OF PARAMETERS ON EFFICIENCY AND KEY SIZE.

2.5 LIMITATIONS ON PARAMETER SELECTION.

The choice of parameters is limited by security

considerations. There are two possible attacks on the general TK-PKC.

2.5.1 Breaking the trapdoor.

To achieve equivalent security to the Merkle-Hellman TK-PKC the superincreasing knapsack components must be well hidden. The two parameters which affect this are k and y. For a given n and g we set the following limitations:

$$y * k > 200 \qquad \ldots 2.12$$

$$k > 1 \qquad \ldots 2.13$$

2.5.2 Breaking the knapsack problem.

By expanding the message into its binary form it is possible to turn a general knapsack into a binary one. That is if:

$$x_i = (x_{i,g-1} * 2^{g-1}) + (x_{i,g-2} * 2^{g-2}) + \ldots + x_{i,0}$$

Then we can write a general knapsack problem as:

$$S = \sum_{i=1}^{n} \sum_{j=0}^{g-1} (a_{i,j}).(x_{i,j})$$

where:

$$a_{i,j} = a_i * 2^j$$

This is a binary knapsack problem. In order to present a large equivalent binary knapsack problem we set:

$$n * g > 200 \qquad \ldots 2.14$$

This limit is not sufficient, since not all message bits are equally well hidden. For example, the least significant bit (lsb) is only determined by the n lsbs of each message component. To overcome this weakness the general knapsack must employ a number of random bits in the lsbs of each message vector. If v is the number of random bits, then we can approximately compensate for the effect of a small n (provided n > 2) with the following limitation:

$$n * \frac{(v+1)^2}{2} > 128 \qquad \ldots 2.15$$

Let E´ be the efficiency with v random bits added to each message component. Then it can be shown that:

$$E' = E + \frac{v}{g} \qquad \ldots 2.16$$

In the past few years there have been rapid advances in solving the knapsack problem. Though these attacks have been on the binary version (g = 1), the techniques can be extended to cover the general knapsack problem.

The technique (Ref.209) are based on forming a lattice of rank n appear particularly attractive. If:

$$\text{Density} = \frac{n}{\log_2 (\max_j a_j)} < \frac{1}{\log_2 n} ,$$

then a binary knapsack problem can be broken. However for the general knapsack problem, with the additional restraint of equation 2.15, finding a suitable lattice is more difficult. In this case a better measure is to say:

$$\text{Density} = \frac{n.(g - v)}{h + 1} < \frac{1}{\log_2 n} \qquad \ldots 2.17$$

2.6 A BINARY TK-PKC.

The Merkle-Hellman method is a general TK-PKC with parameters:

n = 200 , g = 1 , y = 200 , k=2.

With these parameters we obtain (equations 8.12-8.15):

E = 2.09, Q =1628 bits , P = 81400 bits.

2.7 A NON BINARY TK-PKC.

Using the assumptions of section 2.6 we can choose an alternative set of parameters. The additional guides used to choose them are:

a) Only the receiver stores Q, so its size is not critical.

b) All users must either store or be transmitted P, so its size should be minimized.

c) Data expansion is a waste of resources, therefore minimize E.

Combining these observations with those of table 2.1, an alternative set of parameters are chosen:

$n = 7$, $g = 32$, $y = 100$, $k = 2$.

With this new set of parameters we find:

$E = 1.76$, $Q = 1436$ bits , $P = 2513$ bits.

In order to compensate for the effect on security of the small n (see section 2.5.2) we must have approximately 5 random bits (v) in the lsbs of each message (equation 2.15). This reduces the efficiency (equation 2.16) to:

$E' = 1.92$

2.8 CONCLUSION.

In this chapter we have demonstrated a non-binary TK-PKC with equivalent security to the Merkle-Hellman method. It has the advantage that the resulting public directory would be 35 times smaller. It also has less message expansion.

Recent improvements in cracking the knapsack problem (Refs.208-209) have been dramatic. These polynomial time solutions have been successful against the binary TK-PKC. As the general system is no more secure than the binary problem, this puts into question the use of this type of knapsack in cryptography.

A particular weakness with the TK-PKC is the trapdoor information. In the next chapter we shall describe a system not based on superincreasing sequences.

2.9 REFERENCES.

201 P.SCHOBI & J.L.MASSEY.

Fast authentication in a trapdoor-knapsack public key cryptosystem.

IEEE SYMP. ON INFO. THEORY, LES ARCS, FRANCE, JUN.1982.

202 P.S.HENRY.

Fast decryption algorithm for the knapsack cryptographic problem.

BELL SYSTEMS TECH. J., VOL.60, PP.767-773, MAY-JUN.1981.

203 T.HERLESTAM.

Critical remarks on some public key cryptosystems.

BIT, VOL.18, PP.493-496, 1978.

204 A.SHAMIR & R.E.ZIPPEL.

On the security of the Merkle-Hellman cryptographic scheme.

IEEE T. ON INFO THEORY, VOL.26, PP.339-340, MAY 1980.

205 Y.DESMEDT, J.VANDEWALLE & R.GOVAERTS.

A critical analysis of the security of knapsack public key cryptosystems.

IEEE T. ON INFO THEORY, VOL.30, PP.601-611, JUL.1984.

206 I.INGEMARSSON.

A new algorithm for the solution of the knapsack problem.

IEEE SYMP. ON INFO. THEORY, LES ARCS, FRANCE, JUN.1982.

207 E.HOROWITZ & S.SAHNI.

Computing partitions with applications to the knapsack problem.

J. OF ACM, VOL.21, PP.277-292, APR.1974.

208 A.SHAMIR.

A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem.

IEEE T. ON INFO THEORY, VOL.30, PP.699-704, SEP.1984.

209 E.F.BRICKELL.

Solving low density knapsacks in polynomial time.

IEEE SYMP. ON INFO. THEORY, ST.JOVITE, CANADA, SEP.1983.

CHAPTER 3

3.1 INTRODUCTION.

This chapter presents a new trapdoor knapsack public key cryptosystem (TK-PKC). The encryption equation is based on the general modular knapsack equation (see chapter 2), but unlike the Merkle-Hellman scheme the knapsack components are not derived from a superincreasing sequence.

The trapdoor is based on being able to transform between the radix and modular representations of the components, via the Chinese Remainder Theorem (Ref.115). The system bears a resemblance to the Lu-Lee system (Ref.118), but whereas their cryptosystem is linear and has been shown to be insecure (Refs.119-120), ours is based on the general modular knapsack equation, which to date has not been generally broken.

The new TK-PKC has a number of advantages over the original system other than security. Firstly, the public key size is reduced from 80K bits to 14K bits. Secondly, the redundancy is reduced from over 100% to around 25%. Compared with the RSA scheme (see section 1.6.3) its main advantage is speed. Typically, knapsack schemes are capable of throughput speeds which are several orders of magnitude faster than the RSA scheme.

3.2 NOTATION.

a_i = i'th published knapsack component.

a'_i = i'th secret knapsack component.

A = published knapsack vector = $(a_1 , a_2 , \ldots , a_n)$.

A' = secret knapsack vector = $(a'_1, a'_2, \ldots , a'_n)$.

E = ratio of (no. message bits : no. ciphertext bits).

g = number of bits on $x_{i,max}$.

$h+1$ = minimum number of bits in any p_i.

n = number of knapsack components.

P = a set of n distinct primes = $(p_1 , p_2 , \ldots , p_n)$.

p_i = h+1 bit prime number.

$p = \prod_{i=1}^{n} p_i$.

PK = number of bits in the public key.

r = maximum number of bits in $(\sum_{j=1}^{n} a'^{(i)}_j)$.

S = hard knapsack problem: ciphertext.

S' = easy knapsack problem.

v = number of random bits in the lsbs of each x_i.

w = a secret modular multiplier, relatively prime to p.

X = n * g bit message vector = $(x_1 , x_2 , \ldots , x_n)$.

x_i = g-bit message block.

3.3 THE NEW TRAPDOOR.

The general modular knapsack problem is given by:

$$S = \sum_{i=1}^{n} a_i \cdot x_i \bmod p \qquad \ldots 3.1$$

When used for cryptography, the a's are the n published knapsack components, p is a published modulus, and the x's are the message bits. In the binary knapsack the x's are 0 or 1, but in the general knapsack they are g bit numbers. The subset sum S is the cryptogram, which only the legitimate user is able to unwind back to the original x's.

Let $(p_1, p_2, \ldots, p_n)$ be a set of primes whose product:

$$p = \prod_{i=1}^{n} p_i$$

$$\text{If } a_j^{(i)} = a_j \bmod p_i$$

is the residue of the j'th knapsack component modulo the i'th prime. Then, by the Chinese Remainder Theorem:

$$a_j \longleftrightarrow a_j^{(1)}, a_j^{(2)}, \ldots, a_j^{(n)}$$

is a bijective mapping. That is, the transformation is one-to-one for all a's between 1 and p-1. Thus if the factorization of p is kept secret, then only the legitimate user will be able to transform the radix representation of the knapsack components into their modular representation. This forms our secret new

trapdoor. Let us now choose a set of n knapsack components and express them in both radix and modular form:

$$a'_1 \longleftrightarrow a'^{(1)}_1, a'^{(2)}_1, \ldots, a'^{(n)}_1$$

$$a'_2 \longleftrightarrow a'^{(1)}_2, a'^{(2)}_2, \ldots, a'^{(n)}_2$$

$$A' = \quad \vdots \qquad \ldots 3.2$$

$$a'_n \longleftrightarrow a'^{(1)}_n, a'^{(2)}_n, \ldots, a'^{(n)}_n$$

Let us then disguise the trapdoor by forming a new set of knapsack components, via the modular multiplication:

$$a_j = a'_j \cdot w \bmod p \qquad \ldots 3.3$$

where w and p are relatively prime. Under this condition we know there exists an inverse (see section 1.3.1) transformation:

$$a'_j = a_j \cdot w^{-1} \bmod p$$

We now publish p, and the modified knapsack components (A) in radix form. This is the public key. The factorization of p and the integer w are kept secret, and hence so is the modular representation of the components (A').

$$\text{Now let } p_{i,min} > 2^h \qquad \ldots 3.4$$

that is, the primes are at least h+1 bit numbers.

Let $x_{i,max} < 2^{g}$ ...3.5

that is, the message blocks are g bit numbers.

And let $\left(\sum_{j=1}^{n} a'^{(i)}_{j} \right)_{max} < 2^{r}$ ...3.6

that is, the columns of A´ sum to an r bit number.

In order to ensure that the encryption equation has a unique decryption, we must ensure that the message to ciphertext transformation X --> S is injective. To guarantee this we must have:

$h >= r + g$ ...3.7

This also ensures that modular multiplication is equivalent to matrix multiplication:

$$(S'^{(1)}, \ldots, S'^{(n)}) = (x_1, \ldots, x_n) \begin{pmatrix} a'^{(1)}_1 & a'^{(2)}_1 & \ldots & a'^{(n)}_1 \\ \vdots & & & \\ a'^{(1)}_n & a'^{(2)}_n & \ldots & a'^{(n)}_n \end{pmatrix}$$

i.e $S' = X \cdot A'$

If the matrix (A´) is non-singular, then:

$$X = S' \cdot A'^{-1}$$ ...3.8

The cryptosystem then operates as follows. A user wishing to send us a message forms the ciphertext:

$$S = (x_1 .a_1 + x_2 .a_2 + \ldots + x_n .a_n) \bmod p$$

From equation 3.1 we compute:

$$S' = S \cdot w^{-1} \bmod p$$

Then, through our known factorization of p, we can tranform:

$$S' \longleftrightarrow (S'^{(1)}, S'^{(2)}, \ldots, S'^{(n)})$$

We then apply $X = S' \cdot A'^{-1}$ and hence recover the message.

The cryptanalyst must either break the factorization of p, attack the trapdoor in some other way (see section 3.5), or solve the general knapsack problem (see section 2.5.2).

3.4 A SMALL EXAMPLE.

In order to help understand the ideas of the last section, a small example is now presented. The example is of course too small for security.

Let n=3 and define P=(37,41,43), hence p=65231, and h=5 (equation 3.4). Choose g=2, that is, the message components are two bit numbers. This dictates that r=3 via equation 3.7 (h>=3+2). Choose n=3 knapsack components which satisfy eqation 3.6, that is, the columns of A' add to less than eight, and express in both modular and radix form:

$$a'_1 = (3 , 1 , 1) \text{ <----> } 125174$$

$$A' = \quad a'_2 = (1 , 5 , 3) \text{ <----> } 151664$$

$$a'_3 = (2 , 1 , 2) \text{ <----> } 122509$$

Now choose w=6553 which is relatively prime to p=65231. Perform the modular multiplication of equation 3.3, and publish the resulting knapsack components:

$$a_1 = 50628$$

$$a_2 = 59907$$

$$a_3 = 3560$$

and the modulus p = 65231.

Compute the invese $w^{-1} = 6553$ (see section 1.3.1), and invert A':

$$A'^{-1} = (1/16) \begin{pmatrix} +7 & -1 & -2 \\ +4 & +4 & -8 \\ -9 & -1 & +14 \end{pmatrix}$$

To transmit a six bit message X=(1,2,3) a user computes the ciphertext:

$$S = (1.50628) + (2.59907) + (3.3560)$$

$$= 181122$$

$$= 50660 \bmod 65231$$

Using the secret inverse w^{-1} the receiver computes:

$$S' = 50660.2618 \bmod 65231$$

$$= 13257 \bmod 65231$$

Using the secret factors of p, the receiver transforms this into modular form:

S´ = (11 , 14 , 13) <----> 13257

From equation 3.8, the receiver computes:

$$16.X = \begin{pmatrix} 11 & 14 & 13 \end{pmatrix} \begin{pmatrix} +7 & -1 & -2 \\ +4 & +4 & -8 \\ -9 & -1 & +14 \end{pmatrix}$$

giving X = (1 , 2 , 3) as transmitted.

3.5 PRACTICAL CONSTRAINTS.

The choice of parameters (n, r, g and h) is limited by security considerations. Their are two possible attacks on security that we shall consider.

3.5.1 Breaking the general knapsack problem.

The general knapsack problem is identical with that of the last chapter. Therefore we must have the same limits (see section 2.5.2). If v is the number of random bits in the lsbs of each x, then:

$$n * g > 200 \qquad \ldots 3.9$$

$$n * (v+1)^2 > 256 \qquad \ldots 3.10$$

3.5.2 Breaking the trapdoor.

From equation 3.3 we can write:

$$a_j - a'_j.w = 0 \bmod p \qquad \ldots 3.11$$

$$a_l - a'_l.w = 0 \bmod p \qquad \ldots 3.12$$

We can now calculate (eqn.3.11 and eqn.3.12):

$$a_j.a'_l - a_l.a'_j = 0 \bmod p$$

writing this in modular form we get:

$$a_j.a'^{(i)}_l - a_l.a'^{(i)}_j = 0 \bmod p_i \qquad \ldots 3.13$$

If the number of combinations of $a'^{(i)}_j$ were too small eqn. 3.13 could be used to break the new TK-PKC. Therefore we set a limit:

$$r >= 63 \qquad \ldots 3.14$$

In order to stop a factorization attack, we must set:

$$h >= 255 \qquad \ldots 3.15$$

3.6 A SECURE TK-PKC.

The efficiency of the cryptosystem is given by:

$$E = 1 : \frac{h + 1 + v}{g}$$

if we assume all primes are exactly h + 1 bit numbers.

From equation 3.7 we get:

$$E = 1 : \frac{g + r + 1 + v}{g} \qquad \ldots 3.16$$

Therefore to minimize E, r and v should be kept small.

The size of the public key (PK) is given by:

$$PK = n.(n + 1).(h + 1) \qquad ...3.17$$

In order to reduce this, n should be kept small.

From the above arguments and the security restraints of section 3.5, we end up with the following parameters:

$n = 7$, $r = 63$, $h = 255$, $g = 192$, $v = 5$.

Which from equations 3.16 and 3.17 give:

$E = 1 : 1.36$, $PK = 14336$ bits.

3.7 CONCLUSION.

In this chapter we have presented a new public key cryptosystem based on the general modular knapsack problem. Its security is not based on disguising a superincreasing sequence, but on the difficulty of factoring a number with seven 256 bit prime factors, and on a knapsack problem with a typical efficiency of 1:1.36 and block size of 1736 bits.

The knapsack nature of the system ensures that fast encryption and decryption are possible. In addition the size of the public key, which is typically 14K bits, is not excessive.

It may be possible to attack the trapdoor information more directly, but we can see no productive method of doing this. The only successful attacks on dense trapdoor-knapsacks to date have been on the security of the superincreasing sequence. Our method does not require this. However, it may turn out that all injective trapdoor knapsacks are solvable in polynomial time, in which case all such schemes are useless for cryptography.

CHAPTER 4

4.1 INTRODUCTION.

Using a PKC any two users of a network can communicate securely, without the problem of key distribution (see section 1.4). This overcomes one of the big drawbacks to the use of cryptography. However, there are many cases where the same message needs to be sent to a group of users. The application of cryptography would then severely degrade the network.

In this paper we consider the problem of how to securely set up a broadcast transmission in an electronic network environment. Such networks include broadcast satellite and packet switched digital data services. We assume that any user may take the role of broadcaster, and that the broadcaster wishes to send an identical message to any subset of other users.

4.2 MULTIDESTINATION ADDRESSING.

Through his paper in 1978 (Ref.401), J.McQuillan intoduced three enhanced addressing modes:

1. Logical addressing, where a permanently assigned address can denote one or more physical address.
2. Broadcasting, where the message is addressed to all subscribers.
3. Group addressing, in which the message carries a list of addresses.

These three modes were shown to provide more flexibility, greater

reliability and a reduction in network traffic.

The application of these enhanced addressing modes to cryptogaphic networks is far from straight forward. The problem can be overcome by distributing a session key. But this does not necessarily reduce the order of magnitude of the problem. It would be particularly inefficient for a large number of receivers or a small message.

An alternative solution is to trade complexity in terms of the number of keys in the system rather than the distribution time. In this case however there is an explosive growth in the number of keys required.

For a network with n users, a message intended for a subset k of these would require either:

a) O(k) transmissions.

b) O(n!) keys per user.

If the application of cryptography is not to severely degrade system performance, a more efficient solution must be found.

4.3 EXPLOITING NETWORK STRUCTURE.

If the network has the form of a ring, such as in a local area network, we can consider the following distribution method. The broadcaster enciphers the session key and a list of station addresses under the public key of the first receiver, in

say a clockwise direction. This receiver deciphers the information, and then re-enciphers under the public key of the next receiver on the list.

The information is thus passed from user to user round the ring, with only the intended subset of users able to decipher the session key and addresses. Additionally, each of the intended receivers can 'sign' the re-enciphered information, using the signature property of a PKC, before passing the packet on. After the packet has been round the ring, the broadcaster can check to ensure no user has missed the message: either intentionally or because of error.

4.4 EXPLOITING SECURITY STRUCTURE.

An alternative method of exploiting structure exists if the network of users have a fixed security structure. For example, if the users form a layered hierarchical structure then security can be arranged in layers, with a pair of keys for each level in the hierarchy. Thus the level 1 key would only allow level 1 users to decipher a message. The level 2 key would allow both level 2 and level 1 users to decipher, and so forth. Any particular level therefore possesses the keys to its own level and those below it.

With the above arrangement the maximum number of keys would be O(n). If this is too large, a trade-off of the number of

keys for the number of transmissions is possible. For example with O(logn) keys only O(logn) transmissions are needed.

4.5 A BROADCAST CRYPTOSYSTEM.

Each user's public key defines a one to one mapping between a particular message and ciphertext. The secret decryption key defines the inverse mapping (see section 1.3). It is possible that two keys might define the same mapping, for a particular ciphertext. That is, the same ciphertext will produce the same message under a different key. This would allow the broadcast of a session key in a single transmission.

There are, however, a number of reasons why the above scheme is not practical as it stands. Firstly, no such session key may exist between a group of users. Secondly, if such a key exists, there might be too few for security. Finally, there is the problem of how to find these session keys.

4.5.1 Use of message redundancy.

A given ciphertext does not have to decipher to exactly the same message for each intended receiver, but rather the subset of bits that forms the session key does. As an small, example consider the problem of transmitting a 2-bit session key to two of three receivers.

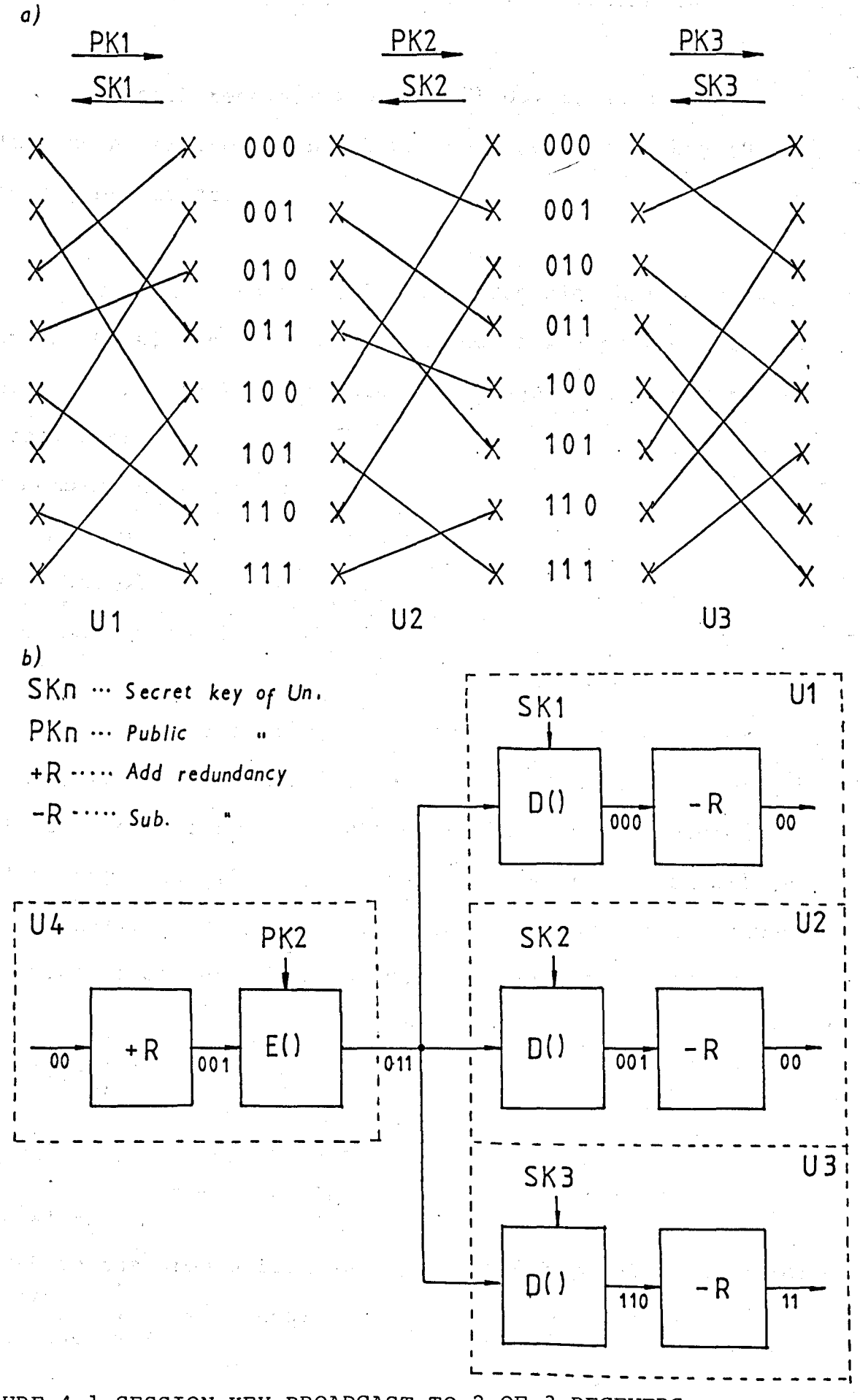


FIGURE 4.1 SESSION KEY BROADCAST TO 2 OF 3 RECEVERS.

If each receiver mapping (U1-U3) is as shown in figure 4.1a, then figure 4.1b shows how the session key (00) can be securely broadcast.

Though it is intuitively reasonable that there are more session keys, it is important to know approximately how many there are. This will allow us to gauge the security of the system. Let:

m = number of bits in message.

c = number of bits in ciphertext.

r = number of redundant bits.

k = number of receivers.

When enciphering an (m-r) bit session key, there will be 2^r different ciphertexts for a given public key. The probability that one of these ciphertexts would be produced under another public key is: 2^(r-c). For k users the probability is: 2^(r-c)(k-1). The probability that none of the 2^r ciphertexts will yield a suitable session key, P(fail), is given by:

$$P(fail) = \left(1 - 2^{(r-c).(k-1)}\right)^{2^r}$$

As a first order approximation:

$$P(fail) \doteq 1 - 2^{r + (r-c).(k-1)}$$

Therefore the probability of one session key being suitable:

$$P(success) \doteq 2^{c + k.(r-c)}$$

Of the 2^(m-r) possible session keys, let Q be the

number that are correctly deciphered by all k receivers. It follows that:

$$Q = 2^{(m-r)} * P(\text{success})$$

$$Q = 2^{m + c.(1-k) + r.(k-1)}$$

Let $Q = 2^b$, then:

$$r = \frac{b - m + c.(k-1)}{k-1}$$

If we assume a bijective mapping (m = c) then:

$$r = \frac{b + m.(k-2)}{k-1} \qquad ...4.1$$

For security it is important to have a sufficiently large session key and choice of session keys. If we say both must have a selection of at least 2^v, then:

$$b = m - r = v \qquad ...4.2$$

Substituting into equation 4.1 we obtain:

$$(m - v).(k-1) = v + m.(k-2)$$

Therefore in order to have a practical cryptosystem:

$$m = v.k \qquad ...4.3$$

Equation 4.3 tells us that for a given message size (m), the number of receivers (k) who can obtain a session key in one broadcast is inversely proportional to the security (v). That is we have a degradeable cryptosystem, where it is possible to trade security with the number of receivers.

4.5.2 A practical broadcast cryptosystem.

Consider how the ideas of the last section can be applied to a trapdoor knapsack PKC (see section 1.6.2). Let C1 and C2 be the ciphertexts formed by two users keys:

$$C1 = \sum_{j=1}^{m} a_{j,1} * x_{j,1}$$

$$C2 = \sum_{j=1}^{m} a_{j,2} * x_{j,2}$$

where $x_{j,k}$ = bit j of message to user k,

$a_{j,k}$ = j'th knapsack component of user k.

We require C1 = C2, and the first (m-r) bits of each message to be the same. Therefore:

$$\sum_{j=1}^{m-r} (a_{j,1} - a_{j,2}).x_{j,1} + \sum_{j=m-r+1}^{m} a_{j,1}.x_{j,1} + \sum_{j=m-r+1}^{m} a_{j,2}.x_{j,2} = 0$$

or

$$\sum_{j=1}^{2m-r} a_j.x_j = 0 \qquad \ldots 4.4$$

where $a_j = a_{j,1} - a_{j,2}$ for $j = 1$ to $j = m-r$

$= a_{j,1}$ for $j = m-r+1$ to $j = m$

$= a_{j-r,1}$ for $j = m+1$ to $j = 2m-r$

and $x_j = x_{j,1}$ for $j = 1$ to $j = m$

$= x_{j-r,1}$ for $j = m+1$ to $j = 2m-r$

To find a solution to equation 4.4 does not seem practical, since it would be even more difficult than breaking the knapsack cryptosystem. However it is possible to exploit the fact that there is not just one solution, but approximately 2^b.

Consider what happens if we combine two vectors. That is, we force two message components to be the same: either both 1 or both 0. When we do this the number of knapsack components is reduced by one, but the number of probable solutions is reduced by half. So after f combinations, there will be 2m-r-f components and approximately 2^(b/f) solutions.

To help explain the principle, consider a small example. We wish to distribute a 3-bit key to 2 users whose public keys consist of 8 knapsack vectors:

a = (123, 92, 233, 61, 11, 188, 103, 134)

b = (132, 210, 177, 70, 201, 107, 88, 54)

We want the first three message bits to be the same. This forces us to combine the first three components of the two receivers. Combining an extra five terms we get 8 new knapsack components:

$$c_1 = a_1 - b_1 = -09$$

$$c_2 = a_2 - b_2 + a_8 = +16$$

$$c_3 = a_3 - b_3 - b_8 = +02$$

$$c_4 = a_4 = +61$$

$c_5 = a_5 + a_6 - b_5 = -02$

$c_6 = -b_7 = -88$

$c_7 = -b_4 = -50$

$c_8 = a_7 - b_6 = -04$

It is now possible to find components that satisfy equation 4.4:

61 + 02 -50 -09 -04 = 0

Therefore all terms in new components 1, 3, 4, 7, 8 must be 1.

That is:

ax = (1,0,1,1,0,0,1,0)

bx = (1,0,1,1,0,1,0,1)

The ciphertext which results when either of these vectors is enciphered is 520. Both receivers will obtain the session key (1,0,1).

In a practical system we would choose f=b, giving a 50% chance of solution. Clearly the algorithm used to select the combinations should be random. If it is not the process could be duplicated by a cryptanalyst.

4.6 CONCLUSION.

It has been shown that in a packet switched network, considerable savings in transmission time, that is , packet hops, can be achieved using multi-addressed packets. In this chapter we have considered the problem of secure communication of such a

broadcast.

Three possible solutions were put forward, which are not mutually exclusive. Taking advantage of the network structure and security hierarchy were two of the schemes. However, these can only be applied in certain cases.

A more general solution to the broadcast problem was also put forward. It uses redundancy between the messagetext and ciphertext. The method effectively trades security with redundancy, number of users, and transmission time.

Finally we present a practical broadcast cryptosysem based on the knapsack PKC. It is applicable to any knapsack cryptosystem, including our own given in chapter 3.

4.7 REFERENCE.

401 J.M.MCQUILLAN.

Enhanced message addressing capabilities for computer networks.

PROC. IEEE, VOL.66, NO.11, PP.1517-1527, NOV.1978.

CHAPTER 5

5.1 INTRODUCTION.

Books, films, television, records and computer software can all be stored and distributed in digital form. Electronic publishing, where information is stored in data banks and distributed over a convenient communication channel, offers many advantages over traditional methods of distribution. However if it is to replace other means of publishing, it is necessary to provide a means of rewarding the Author, Producer, Musician or Programmer.

Copyright is traditionally used to protect the publisher from the pirate. In this chapter methods of enforcing the copyright law will be considered. In particular the application of a PKC to the problem.

5.2 ELECTRONIC PUBLISHING.

Writing programs for personal computers is now a big business, approaching the level of more traditional published information: such as books, television and films. In the next ten years an increasing proportion of this information will be distributed in digital form. This offers many advantages:

1. Ease of distribution - e.g. over telephone lines.
2. Less redundancy - The size of a market for books is often unpredictable. But since copies can be quickly and cheaply made, unnecessary copies need not be produced.

3. Fast update - The lag between supply and demand can be broken.

Electronic publishing does have some disadvantages. Some people might like the feel of vinyl or not like reading from a screen. But though many of these ergonomic problems can be overcome, one major problem remains: how to stop people making illegal copies.

In order to make publishing worthwhile, it must be possible to reward those responsible for developing the product. However information stored in digital form is very much easier to copy, making copying more financially rewarding.

5.3 COPYRIGHT AND CRYPTOGRAPHY.

Copyright is used to give the sole legal right to print, publish, perform, film or record a literary, artistic or musical work (Ref.501). This legal deterrent could easily be extended to cover the newer forms of information, such as computer software. However the ease and cheapness with which electronic information can be copied, suggest that better methods of circumventing the pirate are needed. In the field of computer software, some estimates say as few as 1 in 10 copies may be legally obtained.

Cryptography has been used for centuries to secure communications over a public link. But because the receiver cannot be relied upon to keep the information secure, this

technique cannot be directly applied to copyright protection.

There have been a number of papers on how cryptography can be applied to protect software (Refs.502-504). However, these methods are not generally applicable to the wider field of copyright protection.

5.4 SOFTWARE PROTECTION.

There are at present a wide variety of ways of protecting software, none of which offer 100% protection. For disc based software these include: non-standard formatting, corrupting the operating system, nibble counting, and using unique timing charcteristics of discs. However, none of these methods have proven sufficient deterrent to the skilled pirate.

Though the methods described above do undoubtedly stop some illegal copying, they have some serious disadvantages. These include being uneducational, and not allowing modifications to a program if it does not work. However, the most important drawback is that the legitimate user is unable to make backup copies.

An improved scheme is based on the use of a dongle. The dongle is a piece of hardware, varying from a linear feedback shift register to a dedicated microcomputer (Refs.503-504), that is repeatedly interrogated by the software. The software is written so that it will not work if the dongle is not present.

Using Tamper Resistant Module (TRM) technology (Ref.505), the dongle is not difficult to protect. However, it is possible to find and delete those parts of the program which inspect the dongle. Additional disadvantages are the cost of the dongle and the difficulty of converting a program to read a dongle.

A recent improvement on the dongle concept is the Intel 27916 KEPROM (Ref.502). It is designed to prevent the use of a PROM programmer copying EPROM based software. Before allowing the contents to be read the KEPROM requires the presence of: a key stored in write only memory, and another similar chip. These conditions prevent direct use of a PROM programmer. Though more sophisticated attacks would succeed, these would be considerably more difficult.

5.4.1 Tamper resistant microcomputer.

The Tamper Resistant Computer (TRC) proposed here is a mix of three technologies: a single chip microcomputer, a tamper resistant chip and a PKC. It is basically an extension of the KEPROM idea to encompass a whole computer. Figure 5.1 shows the function of the TRC.

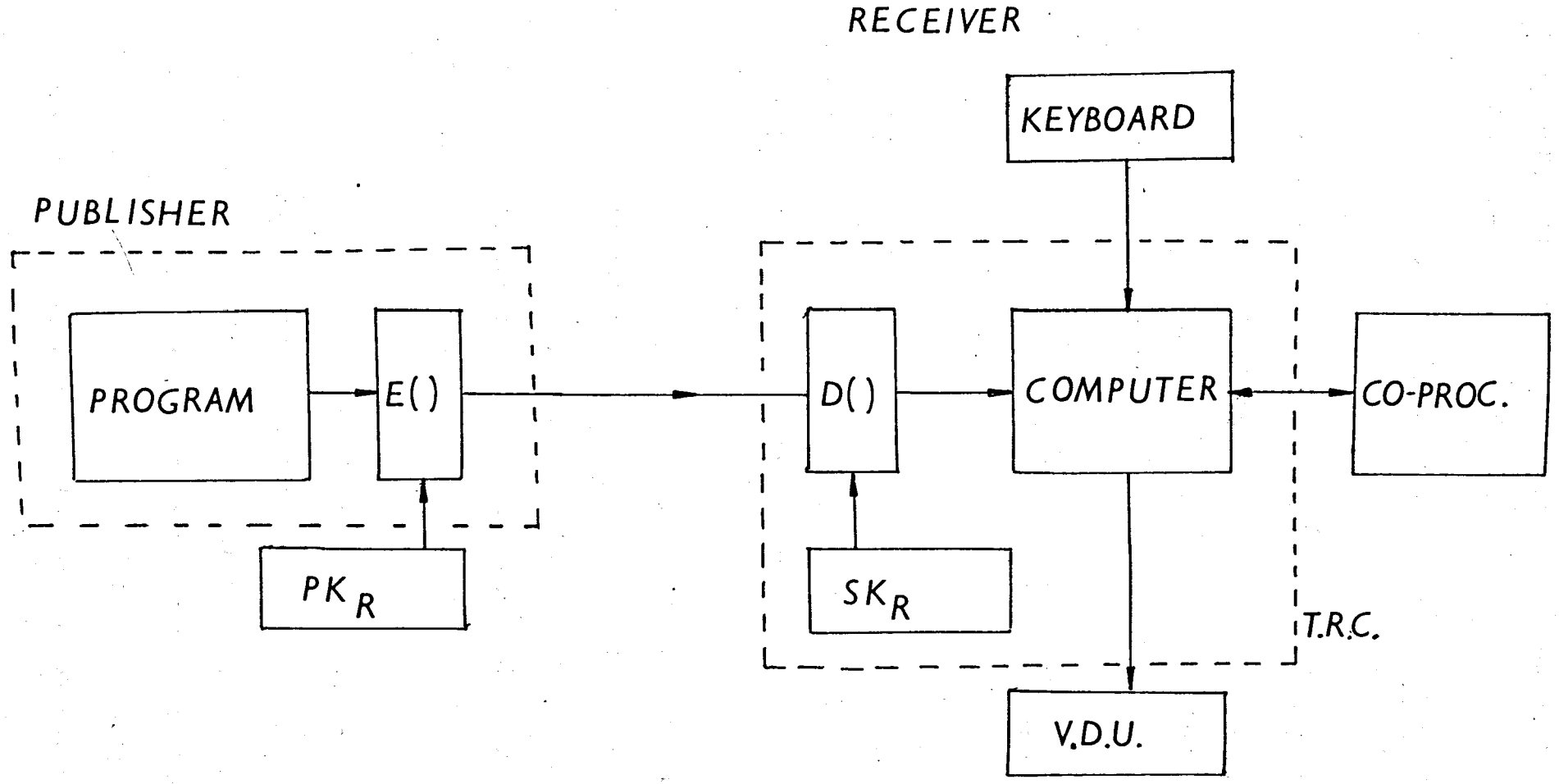


FIGURE 5.1 TAMPER RESISTANT COMPUTER.

A trusted manufacturer could produce the TRC with their built in decryption key. When sold, the associated public key is released and the secret key is kept safe or even destroyed. The software publisher could then encipher the program under the recievers TRC public key. Only the owner of that particular TRC could then run the encrypted program.

If the output is a complex function of the input, it would be impossible to attack the security of this system: assuming the security of the PKC and TRC. Backup copies are also simple to produce. If the TRC malfunctions, the manufacturer could verify and replace it.

The main problem with the TRC is that it would be technologically demanding and expensive. However, in ten years time it is likely that this scheme would be economic.

5.5 GENERAL INFORMATION SECURITY.

The software security methods described in section 5.4 cannot be applied to books or music. This is because the final output is all that is desired, making copy prevention impossible. However, it is possible to help copyright enforcement.

5.5.1 Fingerprinting.

With books, records or video sold today it is almost impossible to make a perfect copy. This provides a means of

detecting and convicting those that sell the duplicates. The 'problem' with digital information is that a perfect copy can be obtained. However, the equivalent detection capability can be provided, using the signature property of a PKC.

Figure 5.2 shows a typical 'fingerprinting' system. Each publisher has a secret key SK, which they use to sign any published information. To do so they take a known hash function of the program and receiver's name. They can then append this 'deciphered' function as an signature or fingerprint.

If any dispute arose as to whether the information was legally obtained, the fingerprint could be used. By 'enciphering' the signature the hash function can be recovered and compared with the recalculated hash function. Since it would be impossible for anyone to produce the 'deciphered' funtion, without the publisher's secret key, the pirate can be detected.

With the above scenario it is probable that the pirate will delete the signature: substituting one of his own. However, this is open to detection. The legitimate publisher could show the similarity of information, and that his was first published. A better alternative might be to hide the signature in the information.

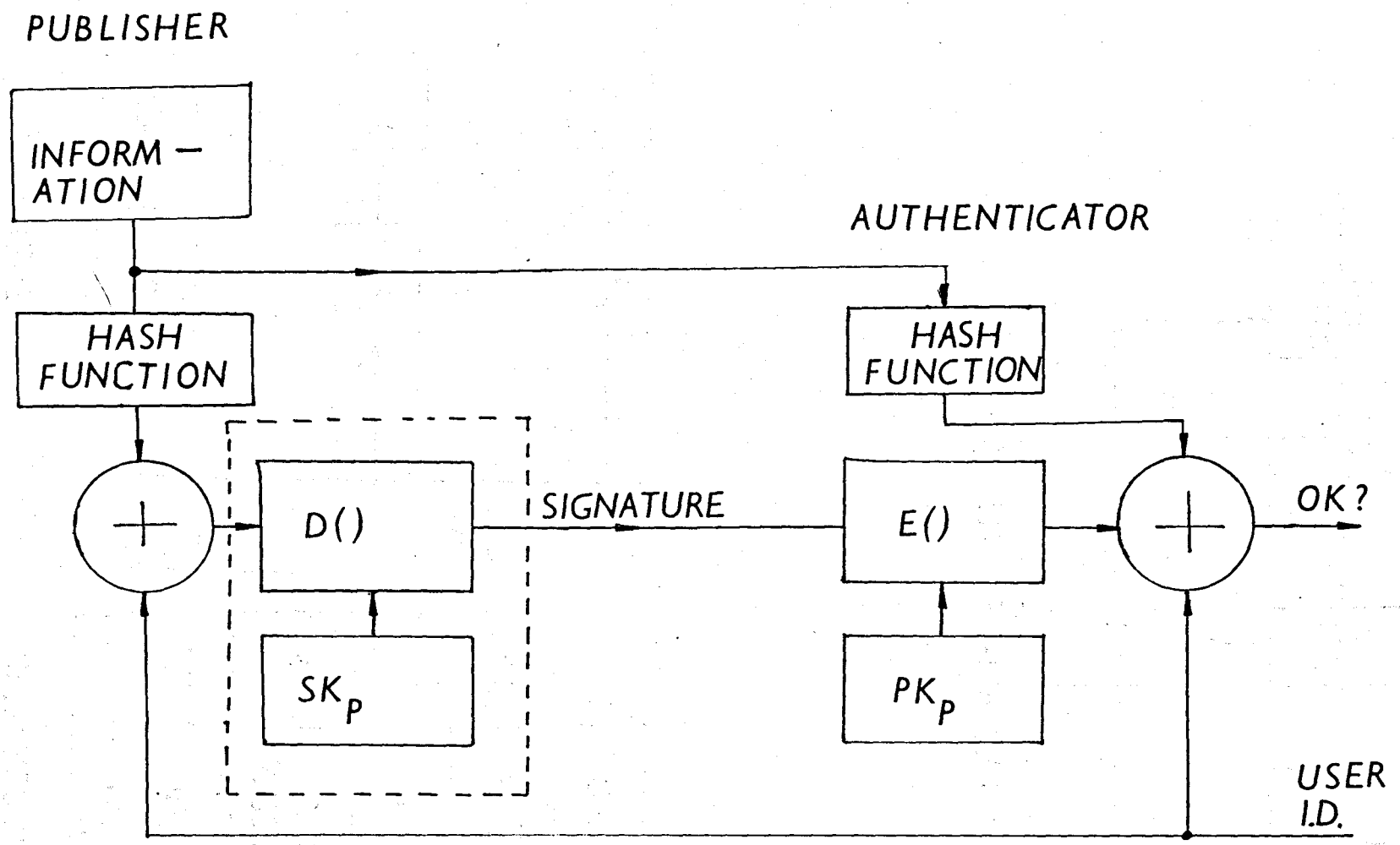
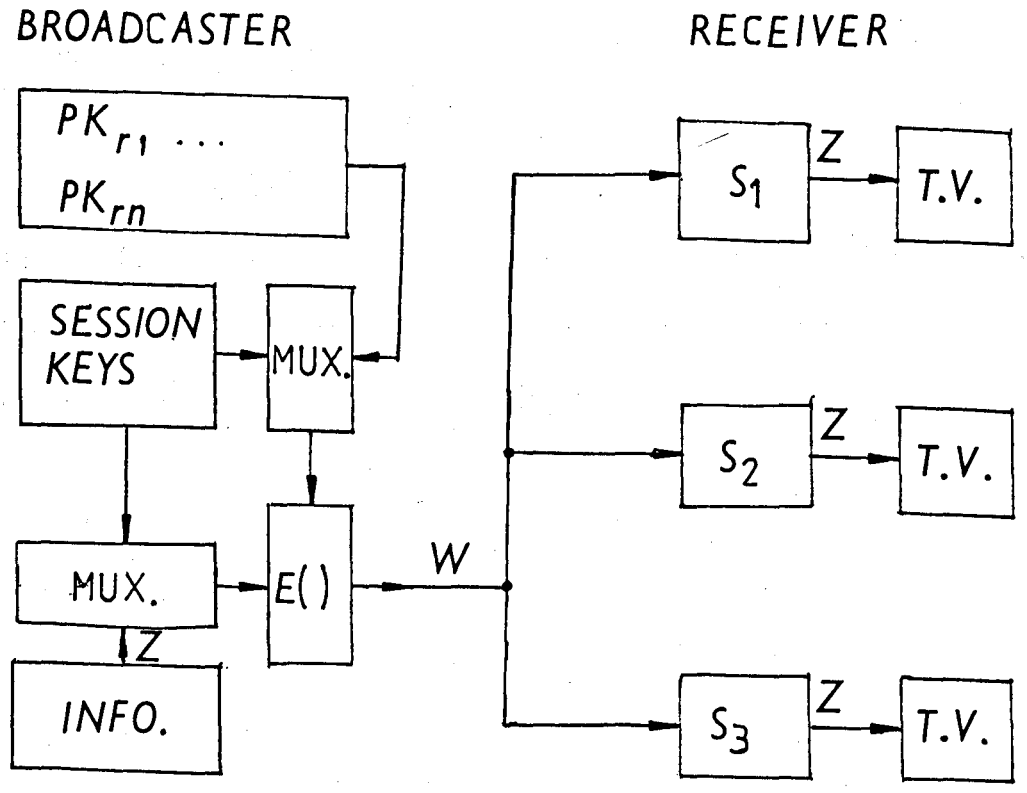


FIGURE 5.2 FINGERPRINTING PUBLISHED INFORMATION.



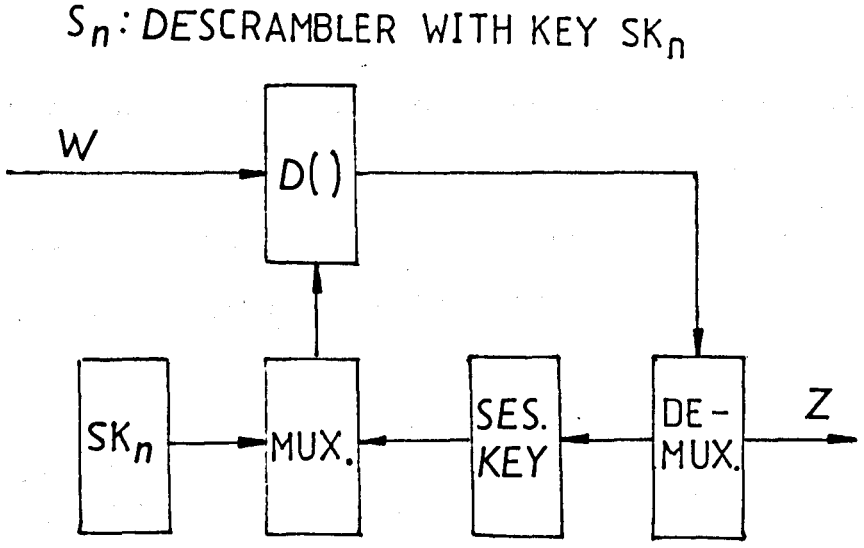


FIGURE 5.3 BROADCAST RECEPTION.

5.5.2. Broadcast reception.

In certain applications, such as Direct Broadcast by Satellite (Ref.506), only immediate copying needs to be prevented. In this case it is possible to use a more conventional encryption technique, as shown in figure 5.3.

Take as an example the broadcast of a live football match. Each receiver who has paid a fee receives the session key, encrypted under their secret key. Then when the match is broadcast it is enciphered under this session key. Since the session key is stored in a TRM there is no way of 'cloning' the session key.

There are a number of possible problems with this scheme: preventing retransmission, stopping recording, and the more practical problem of transmitting a large number of keys. However these problems are not as bad as they appear: retransmission is a more active and detectable copyright infringement; a recording is often of little value after it is transmitted; and the time to send the keys can be reduced using the techniques given in chapter 4.

5.6 CONCLUSION.

In this chapter we have looked at what is probably the hardest application of cryptography: where the receiver cannot be

trusted. Software is one example considered where cryptography can help provide a solution. The concept of the tamper resistant computer was described, and shown to offer an ideal solution. One drawback is that it will not be practical for a number of years.

This chapter has also looked at the wider field of copyright protection, where software techniques cannot be applied. Though absolute security cannot be given, two aids to detection were described. These deterrents should help enable the benefits of electronic publishing to be realized.

5.7 REFERENCES.

501 J.M.HAWKINS.
Oxford Paperback dictionary.
OXFORD UNIVERSITY PRESS, 1979.

502 A.FOLMSBEE, D.HOFF & L.LETHAM.
KEPROM approach locks out software pirates.
NEW ELECTRONICS, PP.78-83, 19-MAR.1985.

503 L.JONES.
The intelligent dongle.
COLL. ON SOFTWARE PROTECTION- THEORY AND PRACTICE, DEC.1984.

504 I.SCHAUMUELLER-BICHL & E.PILLER.

A method of software protection based on the use of smart cards and cryptographic techniques.

EUROCRYPT '84, LA SORBONNE, PARIS, FRANCE, APR.1984.

505 R.G.F.AITCHISON.

A cryptographic approach to software.

ELECTRONIC PRIVACY AND AUTHENTICATION TECHNOLOGY WORKSHOP, HATFIELD POLYTECHNIC, ENGLAND, JULY 1982.

506 S.M.EDWARDSON.

Scrambling and encryption for direct broadcasting by satellite.

INTERNATIONAL BROADCASTING CONVENTION, SEP.1984.

CHAPTER 6

LOW COMPLEXITY CUSTOM VLSI DESIGN.

6.1 INTRODUCTION.

The evolution towards increasingly fast and dense integrated circuits has been well prophesied (Ref.601). However changes in design techniques have been more revolutionary. As the scale of a system is increased by orders of magnitude, different issues become relevant. The cost of defining one million transistors using traditional ad hoc methods is prohibitive, for all but a few applications. This complexity barrier (Ref.902) provides a more realistic obstruction to system growth than minimum speed of light propagation.

Many diverse design methodologies are possible. Today most semiconductor producers employ some combination of gate arrays, standard cells and custom design. Gate array methods use a matrix of predefined transistors whose metal interconnections define a particular boolean logic function. A rapid turn-around is possible, since a manufacturer can have the silicon processed awaiting metalization. Standard cell techniques employ a large library of cells with known electrical and topological characteristics. This is usually more efficient than gate arrays; but slower since all processing steps are needed. Both these semi-custom approaches are excellent as replacements for breadboarded TTL breadboard systems. However, designs based entirely on semicustom ideas do not accrue the full potential of VLSI.

Custom design involves optimizing every transistor

within a circuit. Design costs and development time preclude this approach for most applications. But if a few key cells can be made to account for most of the area, then this approach becomes economical. The remaining random circuitry can then be inefficiently produced. A figure of merit for this regularity is:

$$E = \frac{\text{Total number of transistors}}{\text{Number of drawn transistors}}$$

The diversity of possible applications means that it is not possible to describe a universal design methodology. In this thesis we are concerned with chips having a very high value for E. However before describing their architectures, the overall process of chip design is described.

6.2 TOP DOWN DESIGN.

A structured design methodology is needed to design a VLSI integrated circuit. All such techniques use a similar top down approach. This is a technique of designing a system at a high level and gradually introducing more detail.

It is not a good idea to stick too rigidly to a top down approach. A flexible moving up and down is desirable. This allows consideration of what is efficient at low levels.

Design, simulation and documentation are integral requirements at each stage in a design.

6.2.1 Simulation.

Until the advent of VLSI, gate and transistor simulators were sufficient to design a chip. Today, however, it is necessary to have a modelling tool as a companion to a top down design. It should be capable of describing a digital system at the various levels of hierarchy.

HILO-2 (Ref.603) is a system description language that can be used to describe and simulate a chip. It is a capable of modelling low level primitives, tristate wires and has vector notation. But the most important property of HILO is its ability to straddle the architectural and lower levels of a design.

6.2.2 Documentation.

It is difficult to overstate the importance of documentation. Though it is now an accepted practice in software design, it is still often an afterthought in chip design.

Documentaion should allow someone to understand the design process at each level of hierarchy. To try and comprehend a chip without it, is equivalent to trying to read a program without comments.

6.3 SYSTEM DESIGN.

The first stage in the design of a chip is to choose its basic function and interface. To do this job requires not only careful design, documentation and simulation; but close liaison with potential users. Concerns at this time include: minimizing pin count (Ref.604) and partitioning to reduce communication across boundaries.

Behavioural simulators, such as ISPS (Ref. 605), are used to verify algorithms. These are Pascal like languages, which can be written quickly. No mapping into hardware is required at this stage in the design process.

At the end a provisional data sheet can be produced. This serves the dual purpose of fixing the design and allowing potential users to better understand the capabilities of the device.

6.4 ARCHITECTURE DESIGN.

This intermediate stage, between system and primitive implementation, has no correspondence in a TTL breadboard design. The reason for this stage is to partition the chip into a network of blocks with defined interfaces. These may in turn be subdivided into smaller units: though no more than is sufficient to describe the function of the chip.

At this stage a high level network description language is required. An example of such a simulator is HILO-2 (see section 6.2.1). Typical primitives at this stage might be a gated full adder or Booth's decoder. These would be described in terms of registers and Pascal like 'case' statements. These primitives are then combined into macrocells, which are themselves combined until the whole chip is described. In order to later expand these primitives, it is important to have all signals that pass the interface included. For example, 'control' and 'control bar', if the complement is generated external to the cell.

The high level description is useful for a number of reasons. But one perhaps unexpected benefit is as a guide to hidden complexity. It is tempting to postpone consideration of bits of random logic in an otherwise regular architecture. However, it is usually these areas that cause the most problems. The need to correctly simulate the chip will make this complexity explicit. I have found that if something is difficult to describe at this level of hierarchy, it is a good indication of trouble later on.

Care should be taken to ensure all nodes are easily controllable and observable (Ref.606). This would typically involve a judicious combination of scanpath latches and other test structures.

Complementary to the high level description is a floor

plan. This is a mapping of the network description into a physical placement. The diagram should include all routing, with particular care being taken over the power supply and clock distribution.

This is perhaps the critical stage of a custom chip design, since it is here that the greatest savings in complexity can be made. Chapter 7 will cover this area in greater depth.

6.5 CIRCUIT DESIGN.

At this point in the design the actual implementation technology need be considered. This level in the hierarchy is usually called gate level, but this is somewhat deceptive. Different technologies have a wide variety of properties: differences in relative size of cells and their ability to drive loads. This makes the choice of primitive technology dependent.

In order to make a decision on technologies it is important to consider the application. Past leaders for LSI; NMOS and Bipolar are now being superceeded by CMOS (Ref.607). I beleive that a dominant part of digital VLSI devices will be realized in CMOS because of:

1. Low power consumption.
2. Active switching between the power rails.
3. High noise immunity.
4. Tolerance of power supply variations.

5. Provides a designer´s paradise.

The last point deserves more explanation. Design in Bipolar requires detailed knowledge of transistor operation, while NMOS requires careful consideration of threshold drops in pass transistors. CMOS by comparison is both simple and effective.

CMOS is easy to design in (Ref.608), but to fully take advantage the correct primitives must be employed. Figure 6.1 shows how some useful functions can be implemented more efficiently without using standard logic gates. Figure 6.2 shows an elegant full adder using just 16 transistors, a third less than a logic gate implementation.

The primitives of the architectural level description are now to be expanded. The final primitives, called leaf cells, are normally of eight transistors or less. The circuits shown in figure 6.1 are typical leaf cells. More accurate timing simulations are now included, using parameters gained from a circuit analysis simulator. SPICE (Ref.609) is the most widely used simulator for this purpose. However, it is best used as an educational tool, and only sparingly on critical elements in a design. Their are a number of reasons for this: the amount of computing resources required; an experienced designers ´rule of thumb´ is often sufficiently accurate; and most importantly, the amount of a designer´s time needed to overcome numerical limitations (such as the dreaded ´internal timestep´).

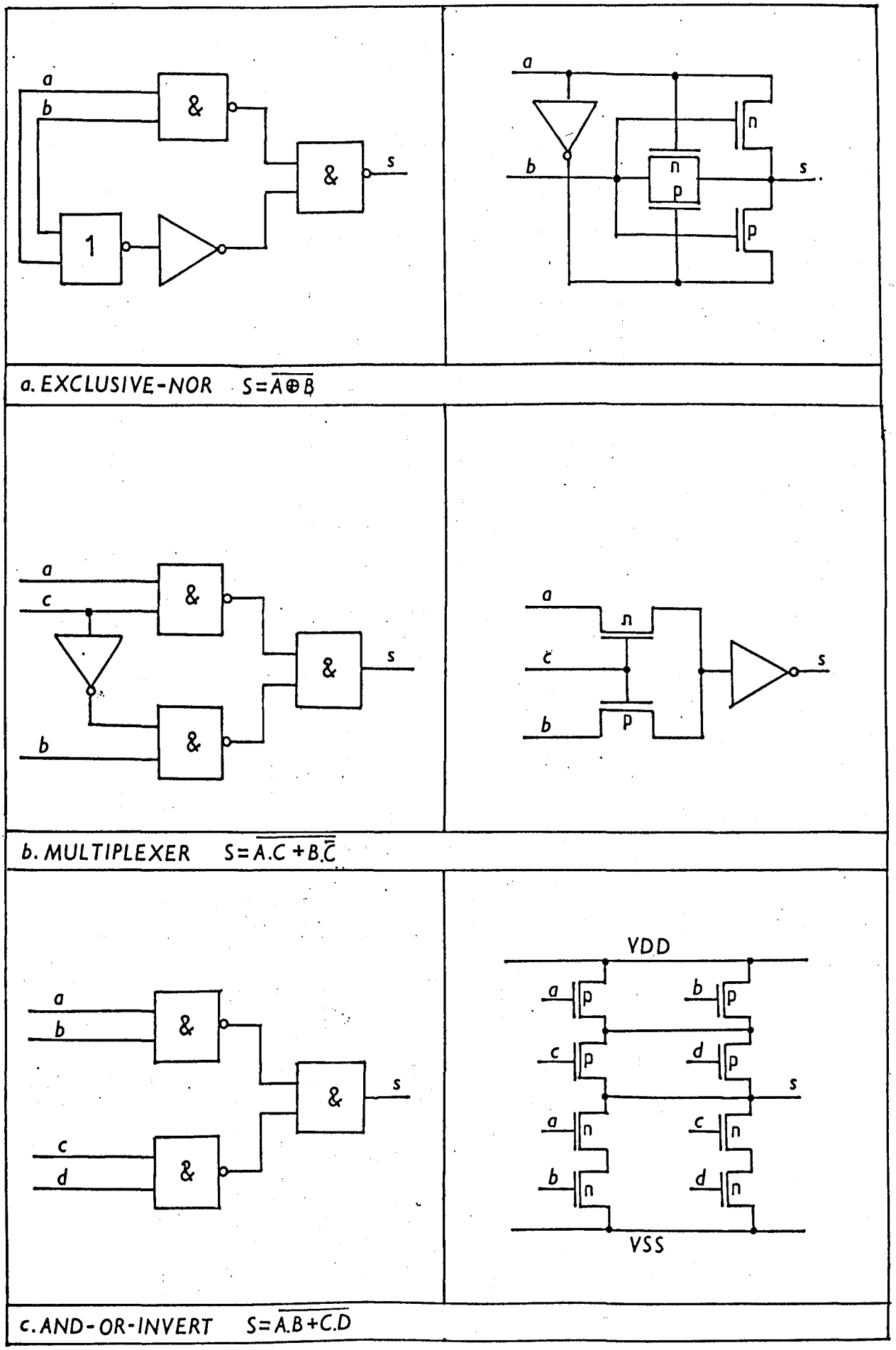


FIGURE 6.1 TRANSISTOR AND GATE LEVEL PRIMITIVES.

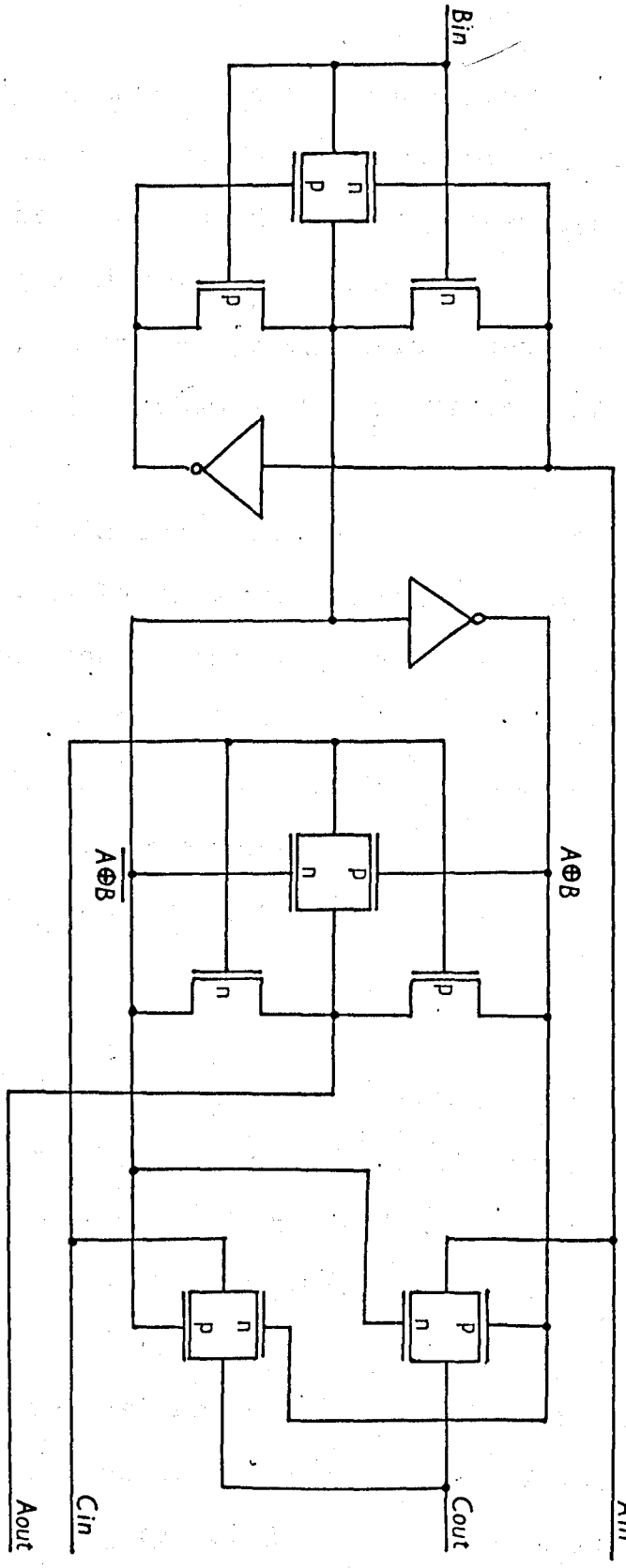


FIGURE 6.2 FULL ADDER WITH 16 CMOS TRANSISTORS.

6.6 LAYOUT DESIGN.

The interface between the designer and chip manufacturer is a set of design rules. The degree of understanding of how these rules are chosen depend on the type of circuit being designed. In general some understanding is desirable: though for some elements, such as pads, it is essential. The design rules used for the ACP (see chapter 9) are C2.5A (Ref.610).

The translation from a transistor diagram into mask shapes is potentially very time consuming. For double level metal CMOS, thirteen mask levels need to be defined. Each level is put down in ascending order, from the active area (level 1) to the passivation (level 13). An unusual feature of C2.5A is the coincident vias and contacts, made possible by the use of polymide insulator. A schematic of a typical inverter is shown in figure 6.3.

To design a leaf cell a number of design methodologies are possible. With full custom design all possible topologies are considered. However the astronomical number of potential permutations results in very slow designs. A far more efficient method is to limit the degrees of freedom. The structured techniqe I adopted has second level metal running vertically with the active area running horizontally. This technique allows packing densities very close to a handcrafted call, while speeding up design time by several orders of magnitude.

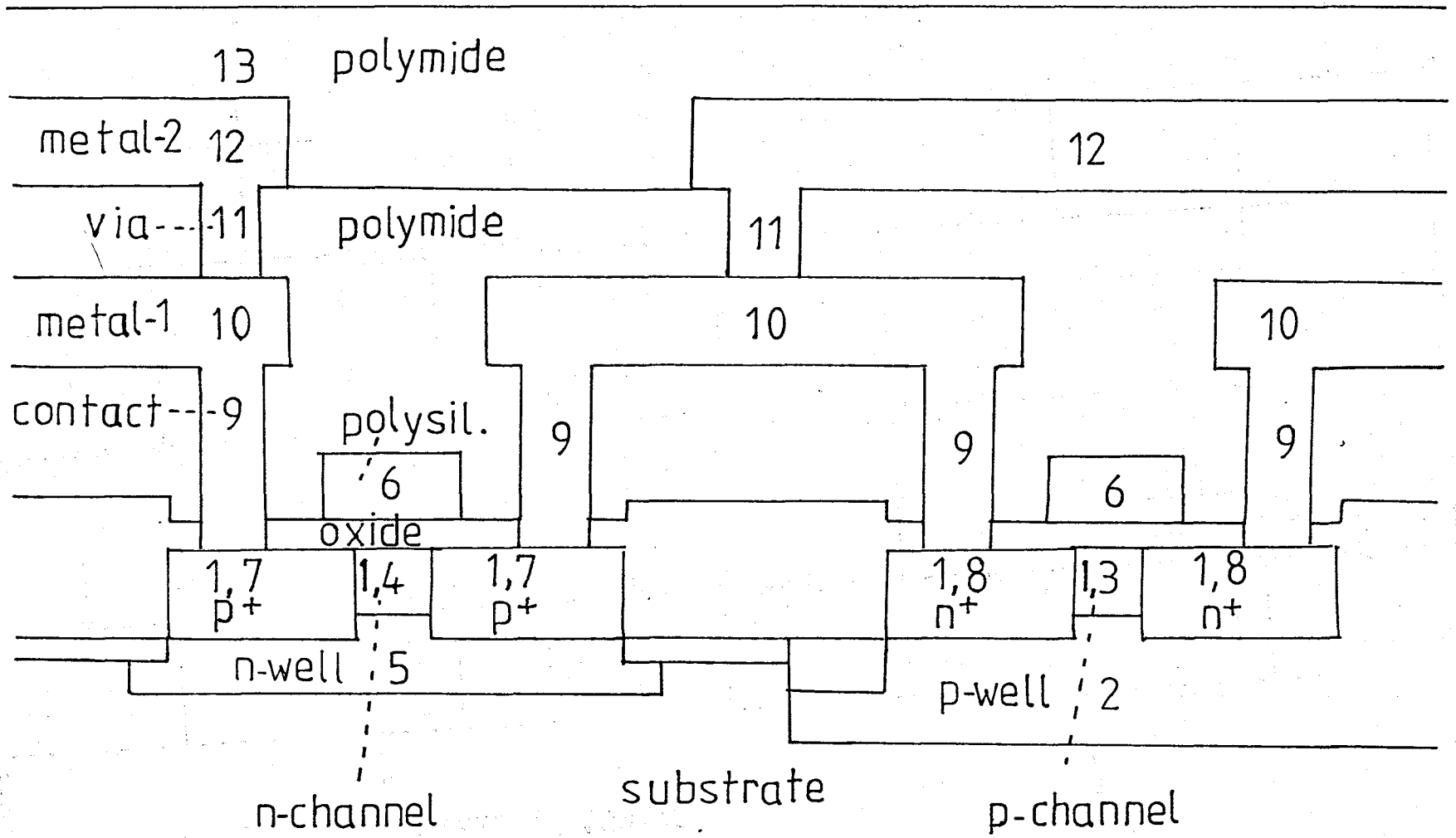


FIGURE 6.3 DOUBLE LEVEL METAL BULK CMOS PROCESSING LEVELS.

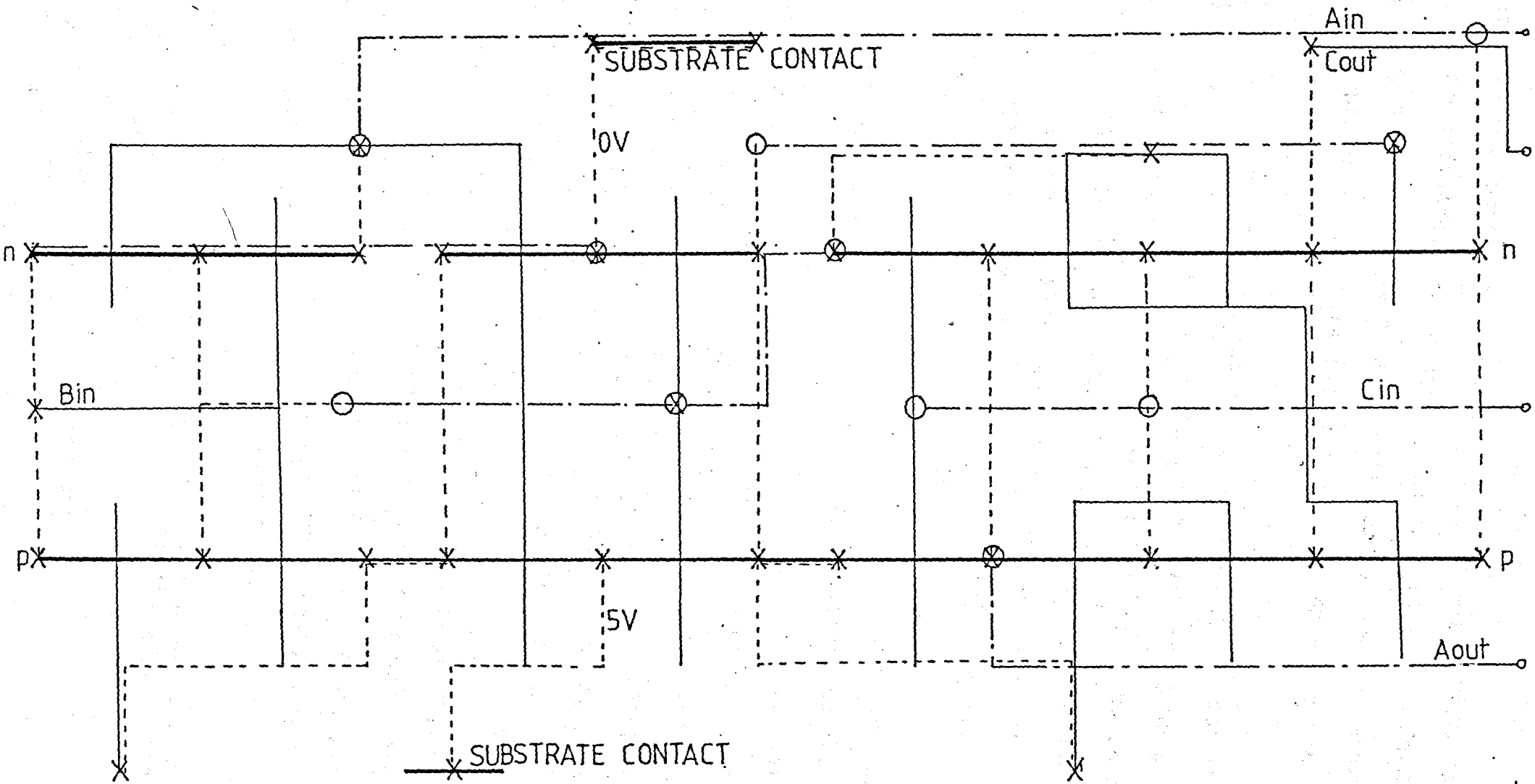


FIGURE 6.4 FULL ADDER STICK DIAGRAM.

6.6.1 Stick diagram.

In order to simplify the layout process an intermediate stick diagram is used. This is essentially a skeleton of the final layout. The 'fleshing out' can be done quickly on a graphics machine, such as the Applicon (Ref.611). As an example, the design of the adder illustated in figure 6.2 is given below.

Figure 6.4 shows a stick diagram of the adder. In order to simplify the design just the four key levels are used: the other levels can easily be added when 'fleshing out'. The second level metal (long dashes) is the first to be put down. This contains all the global signals, such as the clock and power supplies. For the adder, where there are no global signals, it is also used for interconnections. However, normally it only runs in vertical stripes.

The transistors are then defined by polysilicon (solid lines) crossing active area (thick solid horizontal lines). All p-type transistors are placed on one horizontal line and all n-types on another. Long interconnections are done in first level metal (dotted lines). Contacts are shown as crosses and 'vias' (contacts between the two metal layers) as circles.

The stick diagram is a useful aid, but it cannot be done without knowing what is practical. For example, to allow for the larger p-type transistors, the length of the polysilicon must be increased. In practice the stick diagram is done with colour

pencils.

6.6.2 Checking.

Before maskmaking and fabrication the layout must be checked. Firstly, to ensure the design rules are obeyed. The Design Rule Checker (DRC) program finds errors such as metal separation and overlaps around contacts. Secondly, an Elecrical Rule Checker (ERC) checks for the electrical connections. It locates the electrical short and open circuits of a labelled diagram.

The final check is to verify the extracted layout is the same as the one simulated. This is done by converting the description into one that can be used by a Network Consistency Checker (NCC). To do this, it is necessary to first describe the primitives in the language of the NCC. It is then possible to convert the remaining description automatically.

It is best to do checks at the macro cell level, before the circuit is put together. This greatly improves the ability to check errors at the final layout stage. An example of a suite of such checkers are those provided by NCA (Ref.612).

6.7 TESTING AND REDESIGN.

After pattern generation, maskmaking and fabrication it is necessary for the designer to check the finished wafer. If the

design has been done well this should be a simple and easily automated. For simple redesigns, it might be possible to change just the metal and contact masks.

6.8 CONCLUSION.

It is clear that VLSI design will be vital to the next generation of systems. This chapter has described what, in the author's opinion, constitutes a typical low complexity design. Such a design would typically take 6 months for two people working together. However, to achieve this many pitfalls need to be avoided.

A CAD system built around the hierarchical design style would be helpful, but certainly no guarantee of a more cost efficient design. There should be no split of documentation and testing from design. The need for a powerful but flexible description language cannot be overemphasized. At the circuit level the savings from not using logic primitives was shown. However, the most important concept is how to split the design.

VLSI system design is concerned with all aspects af IC design. The task of designing a chip has been split into levels, but too great a partition will result in inefficient design. Each level is influenced by those to follow. System and architectures can only benefit from understanding and experience of the total design process.

It is impossible to make effective trade-offs against a rigid specification. It is at the architectural level that this is most apparent.

6.9 REFERENCES.

601 G.E.MOORE.
Progress in digital integrated electronics.
IEEE INT. ELEC. DEV. MEETING, 1975.

602 C.H.SEQUIN.
Managing VLSI complexity: an outlook.
PROC. OF THE IEEE, VOL.71, NO.1, PP.149-166, JAN. 1983.

603 R.HARRIS S.DAVIDMAN & G.MUSGRAVE.
HILO-2, a system to build upon.
SILICON DESIGN, PP.6-11, FEB.1985.

604 J.W.BALDE.
Alternatives in VLSI packaging.
VLSI DESIGN, PP.23-29, DEC.1983.

605 M.R.BARBACCI.
An introduction to ISPS.
DEPT. COMPUTER SCIENCE, CARNEGIE-MELLON UNIV., 1981.

606 J.A ABRAHAM.

Design for testability.

IEEE SOLID STATE CIRCUITS CONF. CALIF., 1983.

607 J.FIEBIGER.

CMOS - A designer´s dream come true.

ELECTRONICS, 5TH APRIL 1984.

608 P.B.COHEN.

An introduction to CMOS design styles.

VLSI DESIGN, PP.88-96, SEP.1984.

609 A.VLADIMIRESCU, ...

SPICE version 2G user´s guide.

DEPT. ELEC. ENGIN. & COMP. SCIENCE, UNIVERSITY OF CALIFORNIA, BERKELEY, 1981.

610 THE GENERAL ELECTRICAL COMPANY.

Small geometry process C2.5a, lambda layout rules.

GEC HIRST RESEARCH CENTRE, ISSUE 1, DEC.1982.

611 APPLICON.

AGS/860 user´s guide.

APPLICON INC., 32 SECOND AV., BURLINGTON, MA.01803, USA, 1981.

612 NCA.

DRC/ERC/NCC user´s guide.

NCA CORPORATION, SANTA CLARA, CA.95054, USA, 1983.

CHAPTER 7

7.1 INTRODUCTION.

IC technology has reached the point where it is available to all engineers, not just a few circuit designers. However, VLSI design is an immature discipline. What constitutes a 'Good VLSI Architecture' (GVA) is still a moot point.

Mead and Conway popularized new VLSI design techniques in their book 'VLSI System Design' (Ref.701). Some of their conclusions are controversial, and little used. But one point has been widely accepted: the need to re-evaluate architectures in the context of developments in circuit technology.

In the past five years there have been many papers on new architectures. Each new application seems to require a new scheme. But it is not the diversity of the new architectures that is striking, but their similarity.

The most promising genre of new architectures is the systolic array (Ref.702). H.T Kung proposed this class of architectures as one ideally suited to future signal processing chips. What makes these architectures better? In this chapter we shall take a detailed look at a model for future CMOS VLSI devices. The features of a GVA are derived from my experience of circuit design and layout.

7.2 VLSI MODELS.

Designers base their architectures on the hardware available to implement them. In the past twenty years this has resulted in architectures optimized for TTL breadboard implementations. These designs concentrated on the trade-off between the number of gates and the speed of execution. This model is unrealistic for the custom VLSI chips of today.

The TTL designer must learn the 'data book' of logic structures used in custom design. But more important he must be able to compare their cost and performance.

7.3 PERFORMANCE.

To measure the performance of a chip it is possible to take power dissipation, size or reliability. However, for high performance chips the normal measure is the worst case delay. That is, for a given operation, which chip can go fastest.

Standard models of delay stress the importance of the number of gate stages. In VLSI it is as important to consider also, the loading of each gate and the distance a signal has to travel.

All figures will be based on the worst case delay. The important class of self-timed designs, where it is the average

delay that is important, are not discussed.

7.4 DESIGN COSTS.

The most important factor in design cost is labour. This is expressed by number of man-months design time and is a function of the chip complexity.

In breadboard design the complexity is proporional to the number of TTL parts. However, for VLSI regularity is the prime factor.

To reduce design costs it is important to have regularity in structure and communication. This means using just a few simple building blocks, placing them in a regular manner and have a repetitive routing scheme. In other words, it must be easy to describe the chip layout.

7.5 MANUFACTURING COSTS.

In TTL breadboards the manufacturing cost is proportional to the number of devices used. For chip design, the traditional measure for this cost has been area, with three contributing factors: I/O ports, gates and wires. However, this is only half the story, a better measure is yield.

For chips with fault tolerance, the yield is no longer directly connected with area. For regular circuits, it is often

possible to add fault tolerance, without adversely affecting design costs.

Manufacturing costs are also affected by the number of pins and the ease of testing. If testing is not considered early in the design, the test cost can dominate. An attractive flip side of this is to include self-testing. This would allow a chip to test itself without the aid of external equipment or software (Ref.703).

7.6 CHARACTERISTICS OF VLSI ARCHITECTURES.

Architectures can be evaluated using a variety of criteria. To give a single function would require technological dependence. In order to avoid this built in obsolescence, we shall not give a single cost or performance figure. Rather we shall use a table with the following ten parameters:

a) NUMBER OF LOGIC CELLS.

The complexity of the logic, measured by the number of different leaf cells required.

b) NUMBER OF WIRING CELLS.

It might seem strange to use wiring cells. However, for regular circuits this is often how wiring is done. The number of leaf wiring cells can therefore be used as a measure of design cost.

c) LOGIC AREA.

This figure is equivalent to the traditional cost

measure: the number of integrated circuits. The use of area, as opposed to yield, is intended to simplify notation.

d) WIRING AREA.

Until recently neglected, this is increasingly becoming an important measure of manufacturing costs.

e) I/O PORTS.

The number of wires going in and out of the system.

f) TESTING TIME.

The number of input combinations needed to fully test a design.

g) TESTING AREA.

The area of test logic, such as scanpath latches.

h) GATE DELAY.

The time between presenting the operands and reading the result. This figure assumes no pipelining, no delay through wires and no extra time needed to drive larger loads.

i) WIRING DELAY.

This is a measure of the absolute delay through wires and accounts for the time needed to drive larger loads.

j) PIPELINE AREA.

By introducing latches into the data path it is possible to improve the throughput to the delay of a single cell. If possible, this parameter shows what area is required by the pipeline latches.

The above parameters will be measured using order statistics (see section 1.4). The use of order statistics ignores the important constant factors. For small operands, this might

give inaccurate comparisons. For example, a design with a higher order of magnitude delay, could in fact be faster for small operands. However, it is the asymptotic behaviour that ultimately determines whether an architecture is suitable. For the larger arithmetic units of the future this simplification is acceptable.

The importance of different parameters will vary with technology and requirements. But in general the first two, which reflect design costs, will be the most important.

7.7 CONCLUSION.

This brief chapter is a vital prerequisite to Chapter 8. It has modelled the characteristics needed to design GVA. In order not to be dependant on just "today's" technology, the model has included parameters which are not yet considered important. However like any abstractions, they are only meant to reduce the design time. True comparison can only be achieved by practical realizations.

Speed, complexity and yield were explained in their VLSI context. Section 7.6 summarizes the characteristics which will be used in chapter 8.

It is often necessary to sacrifice speed and yield in order to design efficiently. If the complexity cannot be reduced, it is probably more cost effective to go for some semi-automatic

technique.

7.8 REFERENCES.

701 C.MEAD & L.CONWAY.
Introduction to VLSI systems.
ADDISON-WESLEY, 1980.

702 H.T.KUNG.
Let´s design algorithms for VLSI systems.
DEPT. COMPUTER SCIENCE, CARNEGIE-MELLON UNIV., APR.1983.

703 D.KOMONYTSKY.
Synthesis of techniques creates complete system self-test.
ELECTRONICS, PP.110-119, 10TH MARCH 1983.

CHAPTER 8

ARITHMETIC ARCHITECTURES.

8.1 INTRODUCTION.

High performance Digital Signal Processing (DSP) systems are required in a wide variety of applications (Ref.801). These range from processing signals, image and speech, to control, telecommunication and instrumentation. Though the final applications are varied, the nature of the task performed internally is similar. Indeed all these machines are essentialy 'Big Number Crunchers' (BNC), performing some simple arithmetic operation.

All arithmetic architectures can be related to the basic arithmetic operations. This chapter will give a comparative description of architectures for performing them. In particular those suited to custom CMOS chips.

The overriding desire was to create designs that were of very low complexity. Within this limit, the architectures offering the highest asymptotic performance are described. Conclusions are be based on the work of the last two chapters.

8.2 ARITHMETIC ON SILICON.

Research into arithmetic operations is as old as the computer. However, the efficiency of arithmetic architectures cannot be divorced from their implementation technology. As a result of VLSI many new architectures are coming into favour.

Competitive pressures require DSP chips to be both high performance and cheap. In recent years a number of useful DSP 'building block' ECL chips have been developed. These include the AM29500 family (Ref.802) and the MC109XX (Ref.803). However, when single chip implementation is needed, MOS technology is preferred (Ref.804).

In the future DSP will increasingly use dedicated chips. When mapping BNC onto silicon, the architecture is the most important consideration. To do this efficiently a designer will require a vocabulary of arithmetic architectures.

In the rest of this chapter designs of the basic arithmetic architectures are given. These, together with those of Kung (Ref.702), give a good basis for synthesising any required function. Chapter 9 will give an example of how these architectures can be combined to perform modular exponentiation.

8.3 ADDITION AND SUBTRACTION.

Many high performance VLSI devices require a fast adder. The general equation for addition of two conventional radix numbers is given by:

Sum: $Sk = Ak \text{ xor } Bk \text{ xor } Ck\text{-}1$ ...8.1

Carry: $Ck = (Ak + Bk).Ck\text{-}1 + Ak.Bk$ ...8.2

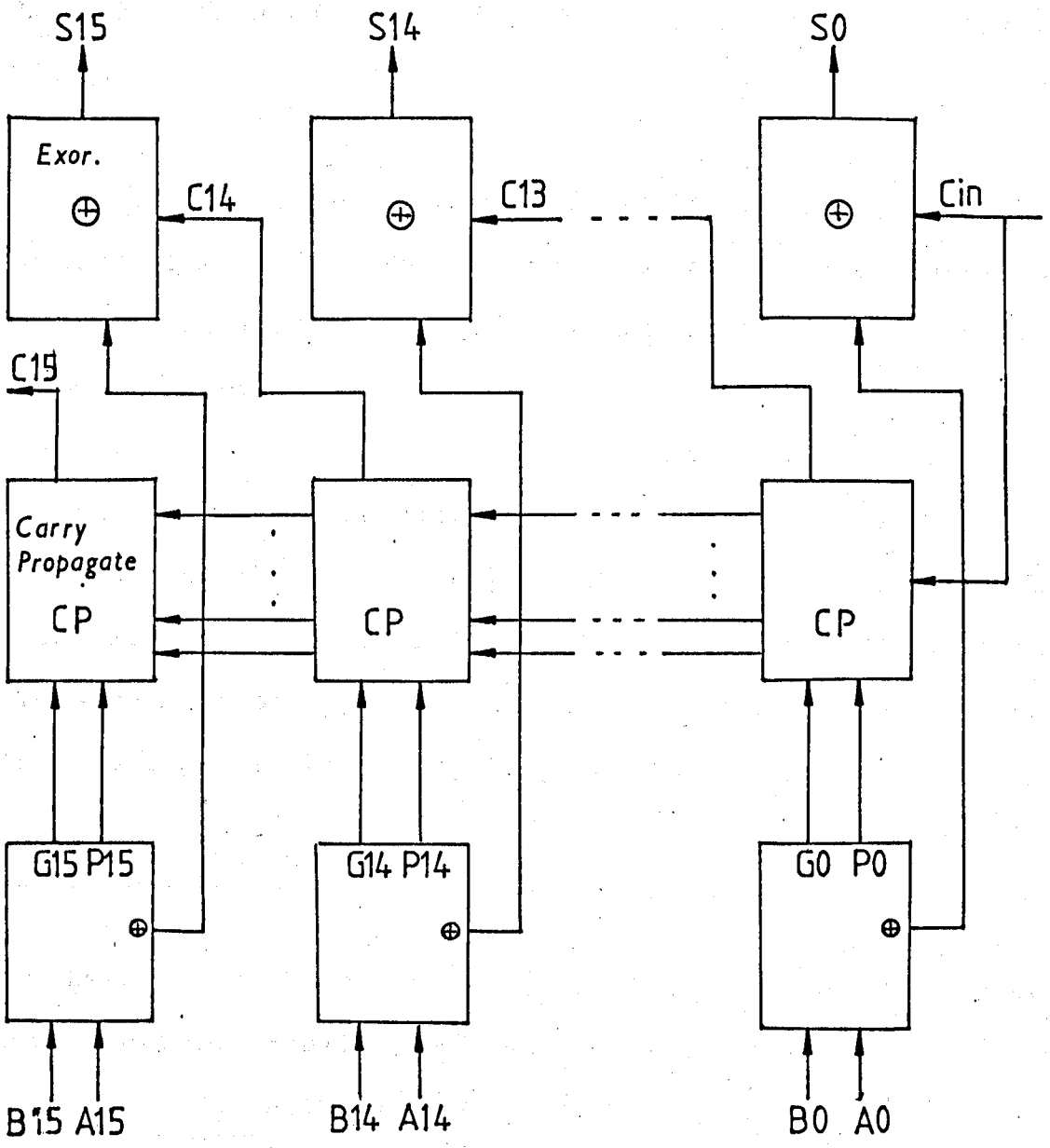


FIGURE 8.1 FULL ADDER ARCHITECTURE.

Where Ak and Bk are the k'th bits of the addend and augend respectively. The recursive equation, 8.2, can be expanded in terms of a carry generate (Gk) and carry propagate (Pk) terms:

$$Ck = Gk + Pk.Gk\text{-}1 + Pk.Pk\text{-}1.Gk\text{-}2 + \ldots + Pk.Pk\text{-}1\ldots P1.Cin \quad ..8.3$$

Where Pk = Ak + Bk and Gk = Ak.Bk. Figure 8.1 shows a typical n-bit adder.

It is possible to have a dedicated subtractor, with a small change in logic. But using 2's complement notation, subtraction can be implemented using an adder. Therefore, it will not be discussed separately.

The speed of addition is limited, because the most significant bit of the sum is dependant on every bit of the addend and augend. The information between stages is held in the carry. Thus methods of improving addition times have concentrated on efforts to improve this carry propagation.

8.3.1 Fast Carry Propagation Schemes.

There have been many schemes to speed the carry propagation (Refs.805-807). The best known are:

Manchester Carry Chain (MCC),

Carry Look-ahead Adder (CLA),

Carry Bypass Adder (CBA),

Conditional Sum Adder (CSA).

The traditional understanding of these schemes is described below.

The MCC and CLA are hardware implementations of equations 8.2 and 8.3 respectively. The CBA bypasses a group of consecutive carry propagate stages. This involves 'ANDing' together the Pk terms of consecutive stages. Finally, the CSA requires the formation of two conditional carry terms for each stage: first assuming no carry-in (Gk), then assuming a carry-in (Pk). Multiplexers are then used to select which carry is chosen, depending on the carry of previous stages.

The four schemes described above represent the traditional choices open to a logic designer. However, for VLSI these categorizations are better understood in terms of the type of tree. Indeed all the fast implementations use the same leaf cells: only their placement and routing vary.

8.3.2 Manchester Carry Chain.

The simplest method to improve carry propagation (CP) is to just use the carry as the signal between stages, while optimizing the CP cells of figure 8.1. Figure 8.2 shows the cells for reducing the worst case delay. Because the transistors connected to Gk and Pk are precharged, the worst case delay is equivalent to n inverters.

The use of alternating logic cells is a technique that is used frequently. It arises because the basic gates in CMOS are inverting. Their use does make the circuitry slightly more complex, but this is more than offset by the increase in speed.

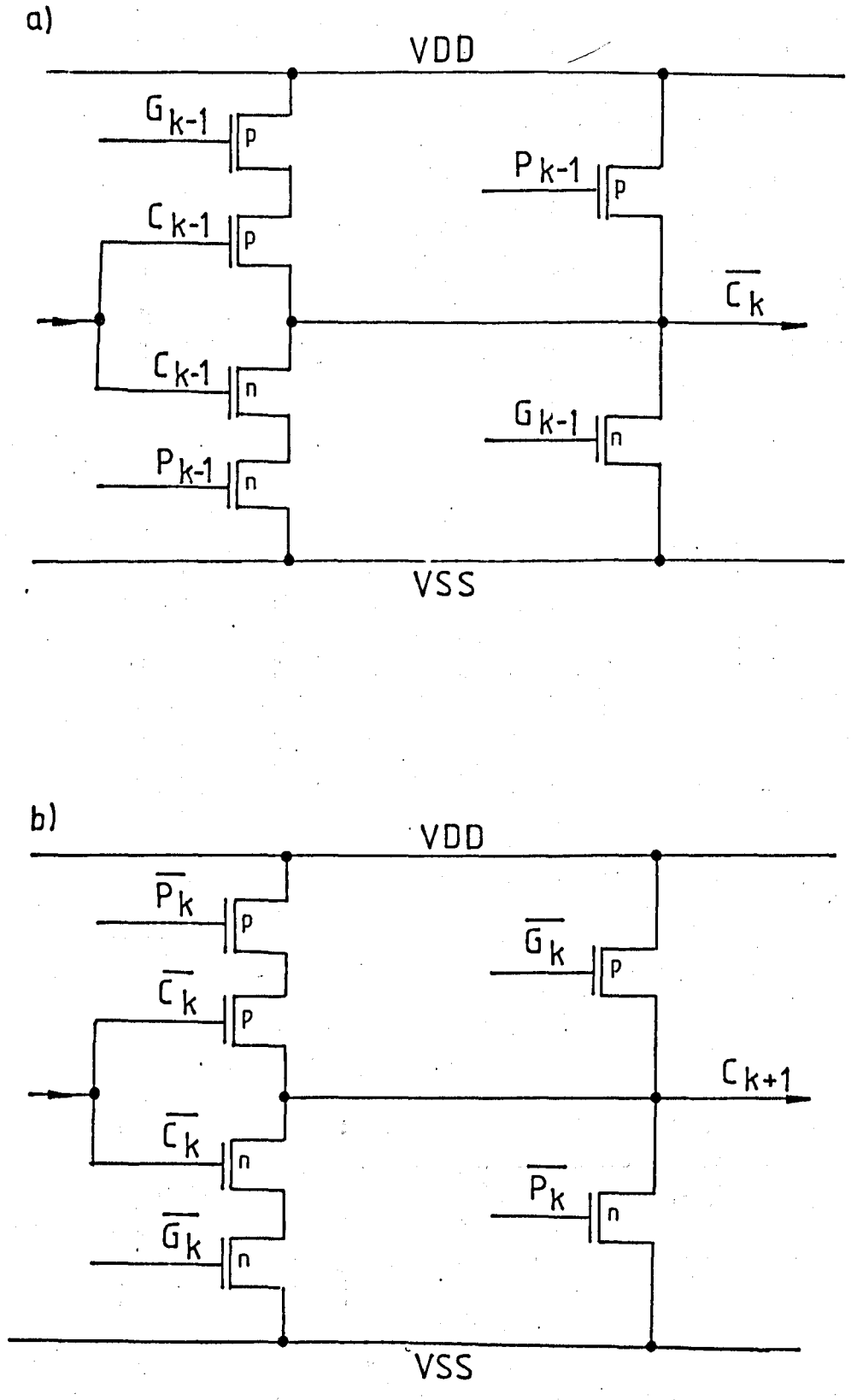


FIGURE 8.2 MANCHESTER CARRY CHAIN CELLS IN CMOS TRANSISTORS.

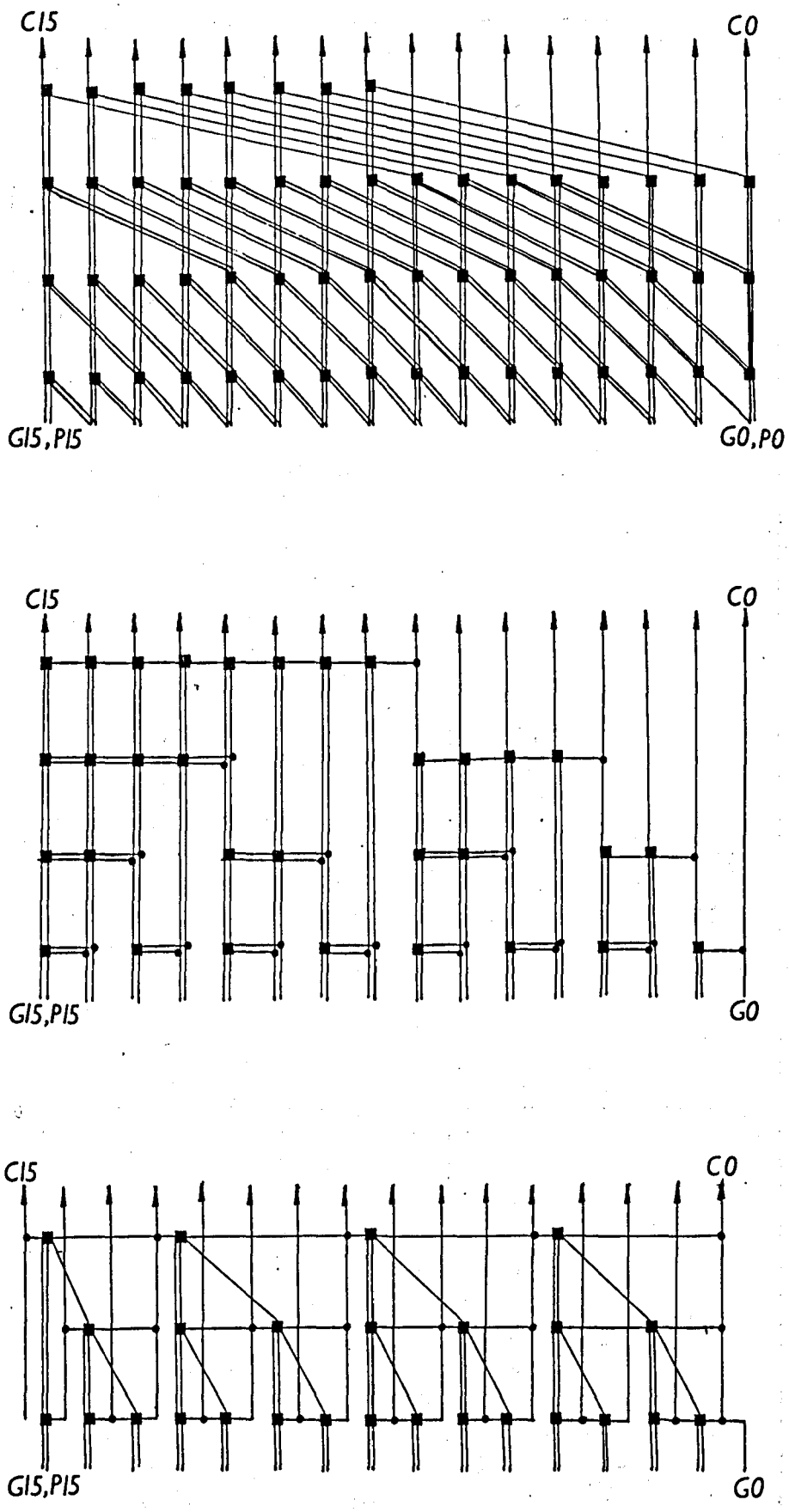


FIGURE 8.3 FAST CARRY PROPAGATE TREE ARCHITECTURES.

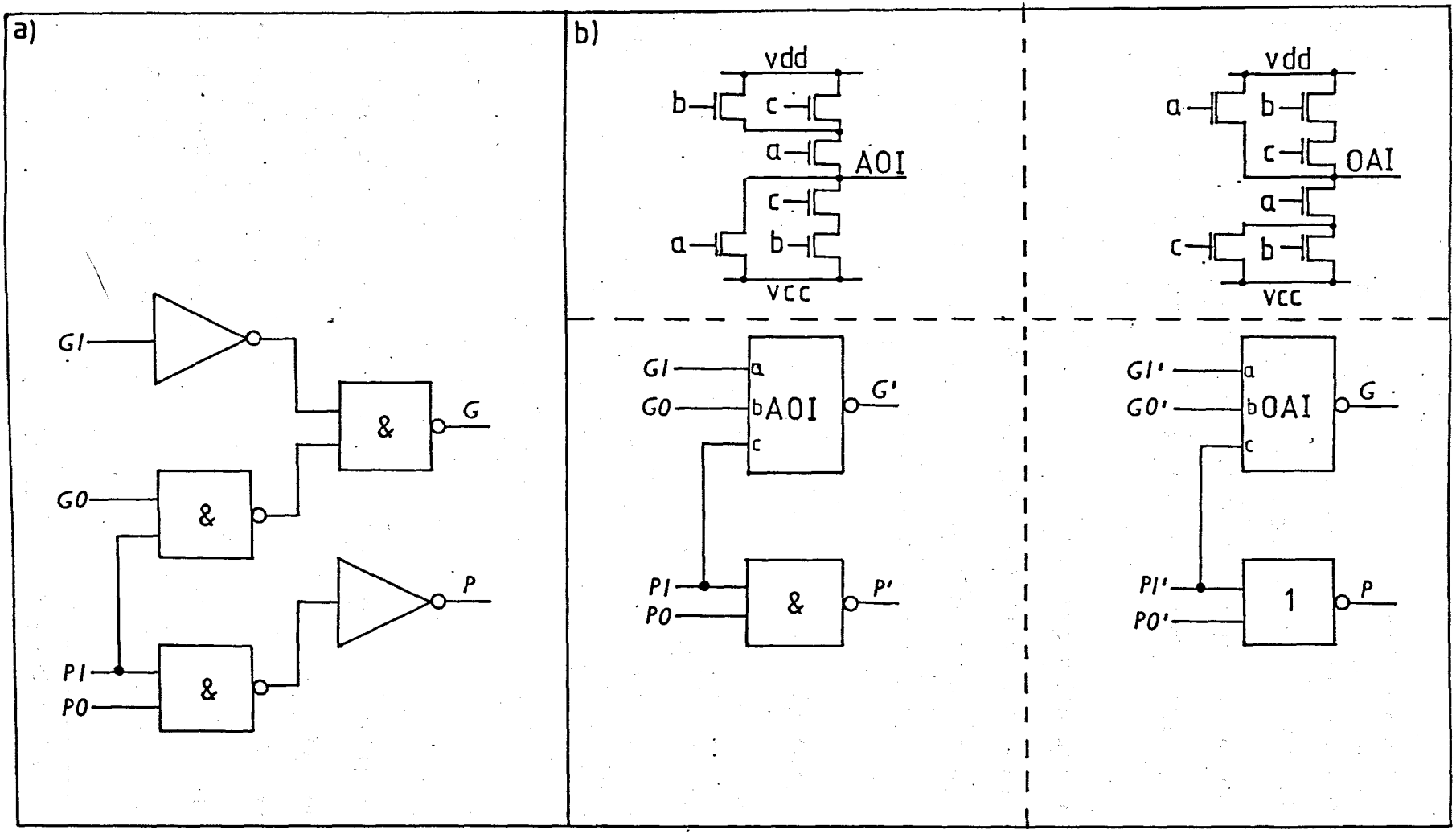


FIGURE 8.4 FAST CARRY CELLS IN LOGIC GATES AND CMOS TRANSISTORS.

8.3.3 Carry Look-ahead Adder.

The CLA is perhaps the most widely used carry acceleration technique. In this scheme the Pk and Gk signals are combined to produce the Ck output (see equation 8.3) of each stage. Traditional methods for generating the carries have used multi-input gates. However, the use of k-input gates is not only irregular, but slow for large k.

A Good VLSI Architecture (GVA) for the CLA was described by Brent and Kung (Ref.808). A modified version of their tree structure is shown in figure 8.3a. The black processing elements combine two P,G terms from a lower level into a new P,G term. The logical contents of the black dots are shown in figure 8.4a. In practice, alternating logic of figure 8.4b would be employed.

This original tree architecture has both less logic area and less gate delays than the Brent-Kung scheme. The new architecture is in fact the optimum, in terms of the number of two input gate delays.

The proof that the final carries agree with equation 8.3 can be done using the Brent-Kung method. A less formal way is to follow the G,P terms as they progress up the tree: verifying that the final generate terms (Gk), do indeed equal the carry (Ck) from equation 8.3.

8.3.4 Conditional Sum Adder.

Figure 8.3b shows a GVA for the CSA. The tree structure is similar to that for the CLA, and makes use of the same processing elements (see figure 8.4). The CBA is better than the CLA, in that it has less routing area. However, this is balanced by the increased delay caused by the loading on the control lines. These points arise from using the same signals for a group of stages. For example, the top eight cells are all driven by the same signal.

8.3.5 Carry Bypass Adder.

A structured version of the CBA is shown in figure 8.3c. This GVA again uses the same processing elements (see figure 8.4) as the CLA , with a modified tree structure. Like the CSA, it has eliminated the routing problem. But this time the loading problem, though worse than the CLA, is not as bad as in the CSA.

8.3.6 Testing an adder.

To test an adder it is normal to put scan-path latches on the input and output. However, in order to quickly test an adder it is important to also have control of the carry-in. This makes it possible to fully test the adder using just the eight test inputs. Table 8.1 shows an example of the test sequence (in hexadecimal)for an eight bit adder.

CIN	ADDEND	AUGEND	SUM
0	$00	$00	$00
0	$FF	$00	$FF
1	$FF	$00	$00
1	$FF	$FF	$FF
1	$00	$FF	$00
0	$00	$FF	$FF
0	$55	$55	$AA
1	$AA	$AA	$55

TABLE 8.1 ADDER TESTS.

8.3.7 Performance Comparison: Addition.

Four types of adder architecture have been discussed. Table 8.2 compares their performances and costs using the parameters of section 7.6.

PARAMETER	MCC	CLA	CBA	CSA
NO. LOGIC CELLS	c	c	c	c
NO. WIRE CELLS	c	log(n)	c	c
LOGIC AREA	n	n*log(n)	n*log(n)	n*log(n)
WIRING AREA	n	n^2	n*log(n)	n*log(n)
I/O PORTS	n	n	n	n
TESTING TIME	c	c	c	c
TESTING AREA	n	n	n	n
GATE DELAY	n	log(n)	log(n)	log(n)
WIRING DELAY	c	n	n	n
LOADING	c	c	log(n)	n
PIPELINE AREA	-	n*log(n)	n*log(n)	n*log(n)

TABLE 8.2 ARCHITECTURE COMPARISON TABLE

Table 8.2 compares the performance of the various tree structures. For area and complexity, the MCC is the clear winner. However, for applications requiring higher performance there is a choice of three. The loading problem of the CSA mean it would only be useful for small operands. For absolute speed, the CLA is preferred. But if the area becomes a problem, the CBA offers a good compromise.

8.4 MULTIPLICATION.

The paper and pencil method of multiplication is to shift and add. An n bit multiplier (R) and an n bit multiplicand (S) produce a 2n bit product (P). This requires the formation of n partial products and their subsequent reduction to a single 2n-bit result. For 3 bit operands, this operation is done as follows:

```
                    s2    s1    s0
                    r2    r1    r0
-------------------------------------
                 r0.s2 r0.s1 r0.s0
           r1.s2 r1.s1 r1.s0
     r2.s2 r2.s1 r2.s0
-------------------------------------
  p5    p4    p3    p2    p1    p0
```

High performance multiplication is an indispensible element in modern DSP. Recently the trend has been towards parallel multipliers. But the serial-parallel method is useful for larger operands.

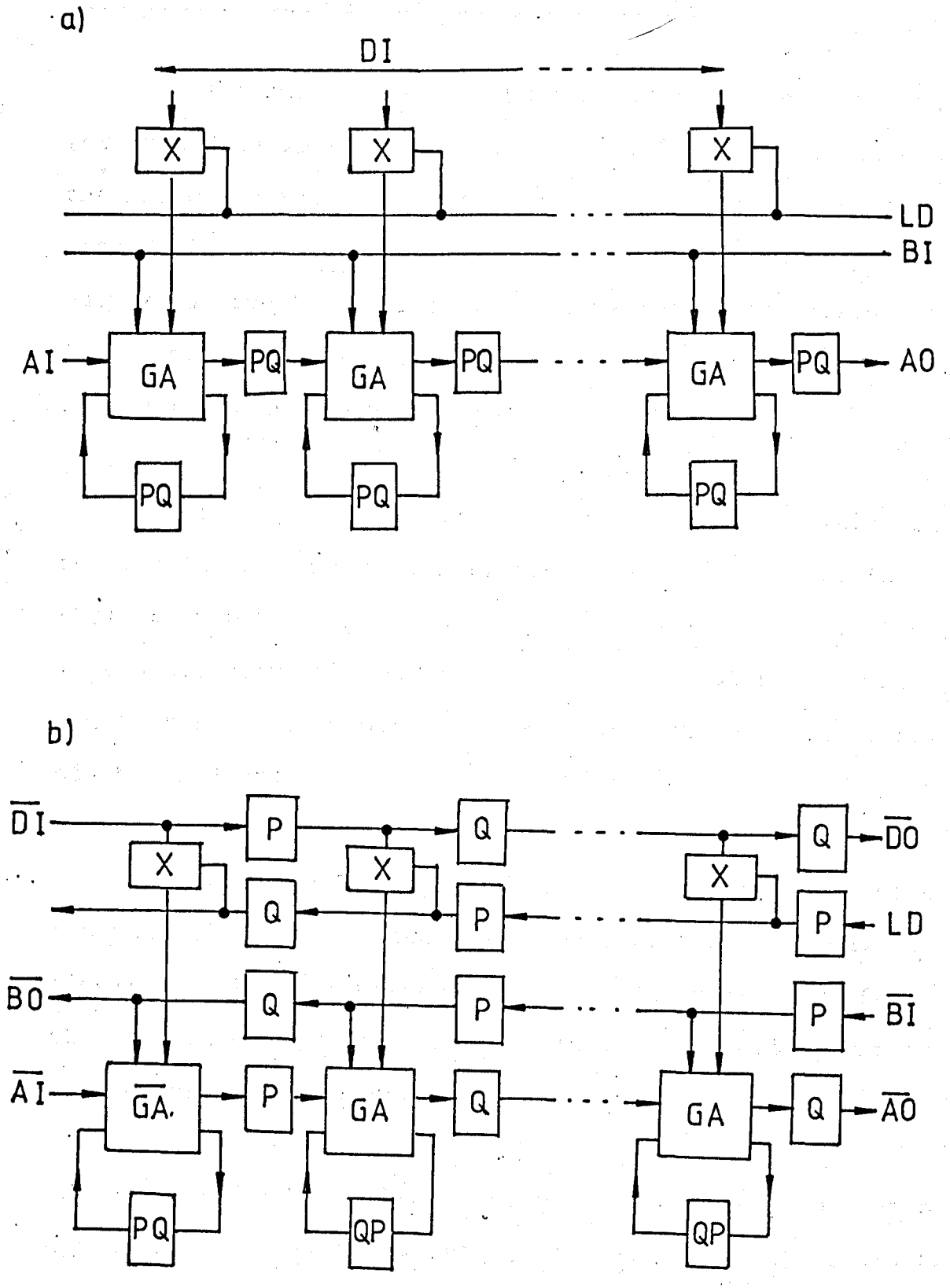


FIGURE 8.5 SERIAL-PARALLEL MULTIPLIERS.

8.4.1 Conventional serial-parallel multiplier.

The serial-parallel multiplier (SPM) is an old technique for speeding up multiplication (Refs.809-810). Figure 8.5a shows the basic architecture of this multiplier. The partial product, stored in the carry save register (the latches (PQ) on the outputs of the full adder (FA)), is first initialized to zero. Then for n clock cycles it accepts S in parallel (DI), R in series (BI), and outputs P in series (AO).

The multiplier bits are ANDed with the multiplicand, and the result is added to the previous partial product. The latches (PQ) are clocked in and out on the two phases of the clock, P and Q respectively. The least significant bit of the partial product register is shifted out to become the next least significant bit of the product.

After n clock cycles the least significant n bits of the product will have been shifted out. If desired, the most significant half of the result, stored in carry save form, can be shifted out during the next n cycles. This final operation can be speeded up using a fast adder.

The above scheme could be modified to accept both operands in series, using a serial to parallel converter. A recent scheme (Refs.811-812) shows how both operands can be simultaneously presented serially. However, one major limitation of all these schemes is the broadcast nature of the multiplier.

For large operands the loading on this line would degrade the performance. An alternative architecture proposed here has no broadcast lines and accepts both operands in serial form.

8.4.2 New serial-parallel multiplier.

The New Serial Parallel Multiplier (NSPM) structure of figure 8.5b employs only a little more hardware than the original scheme. It uses a simple full adder (FA), and its dual (which expects the inputs inverted), not the more complicated (5,3) counter. The half-latches (P and Q) in the multiplier, multiplicand, load and partial product lines are the master or slave of a master-slave latch. They pass data during opposite halves of the clock cycle.

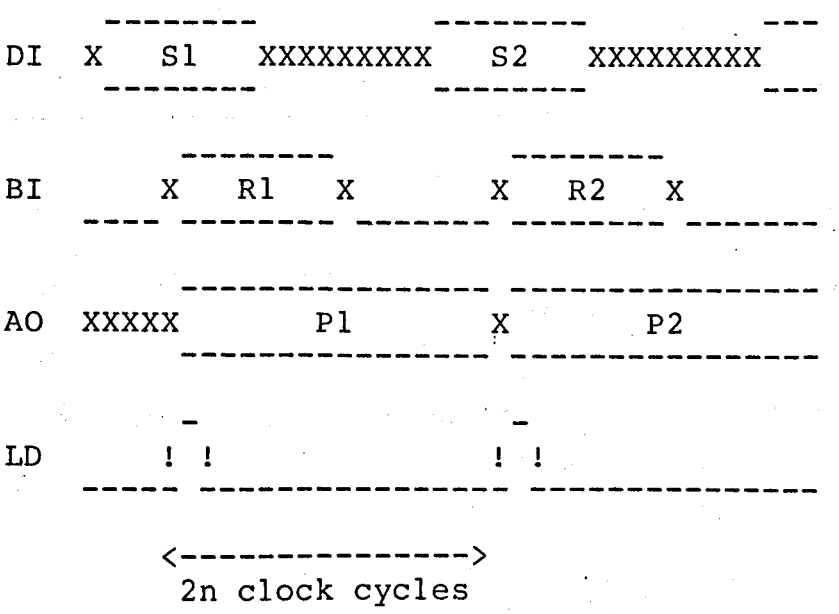


FIGURE 8.6 TIMING OF THE NSPM.

Figure 8.6 shows the timing diagram for the NSPM. For the first n clock cycles S is loaded bit serially into the

holding register (DI). After n/2 clock cycles the load line (LD) is held high for one clock cycle. This loads S into the multiplicand register (X). During this and subsequent clock cycles, R is introduced least significant bit first (into BI). The AI input, normally set to 1, can be used to add a constant.

8.4.3 Testing the SPM.

The lack of control over internal nodes of the SPM presents problems for testing. However, it is possible to test all combinations of the full adders, by setting all the bits of S to 1 and R to:

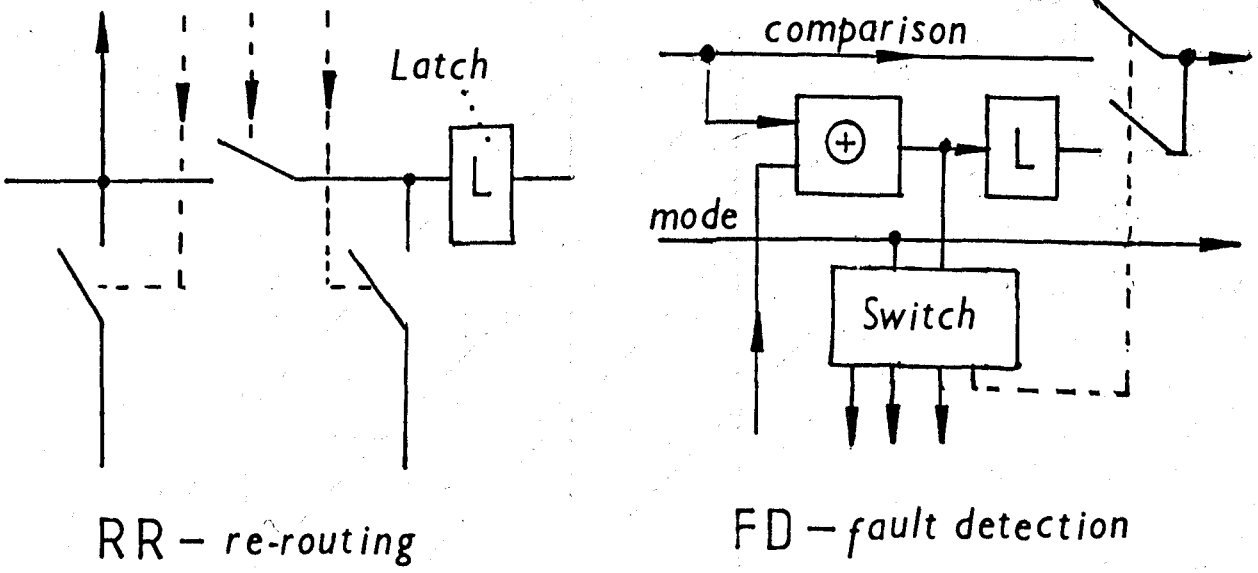
R = 0 ... 00011110

To test the AND gates we set S = 0 and R to all 1's. The other registers can be tested by observing their outputs.

8.4.4 Fault-tolerance with the SPM.

The inclusion of fault-tolerance removes the traditional link between chip area and yield. However, the adverse effect on complexity has meant there have been few practical applications. The simple regular nature of the SPM makes it ideally suited to fault-tolerance.

A number of schemes for including fault tolerance in a serial pipelines have been proposed (Refs.812-814). Only the Hsia scheme (Ref.815) will be considered here. It not only involves least complexity overhead, but is capable of using all working cells.



RR – *re-routing*

FD – *fault detection*

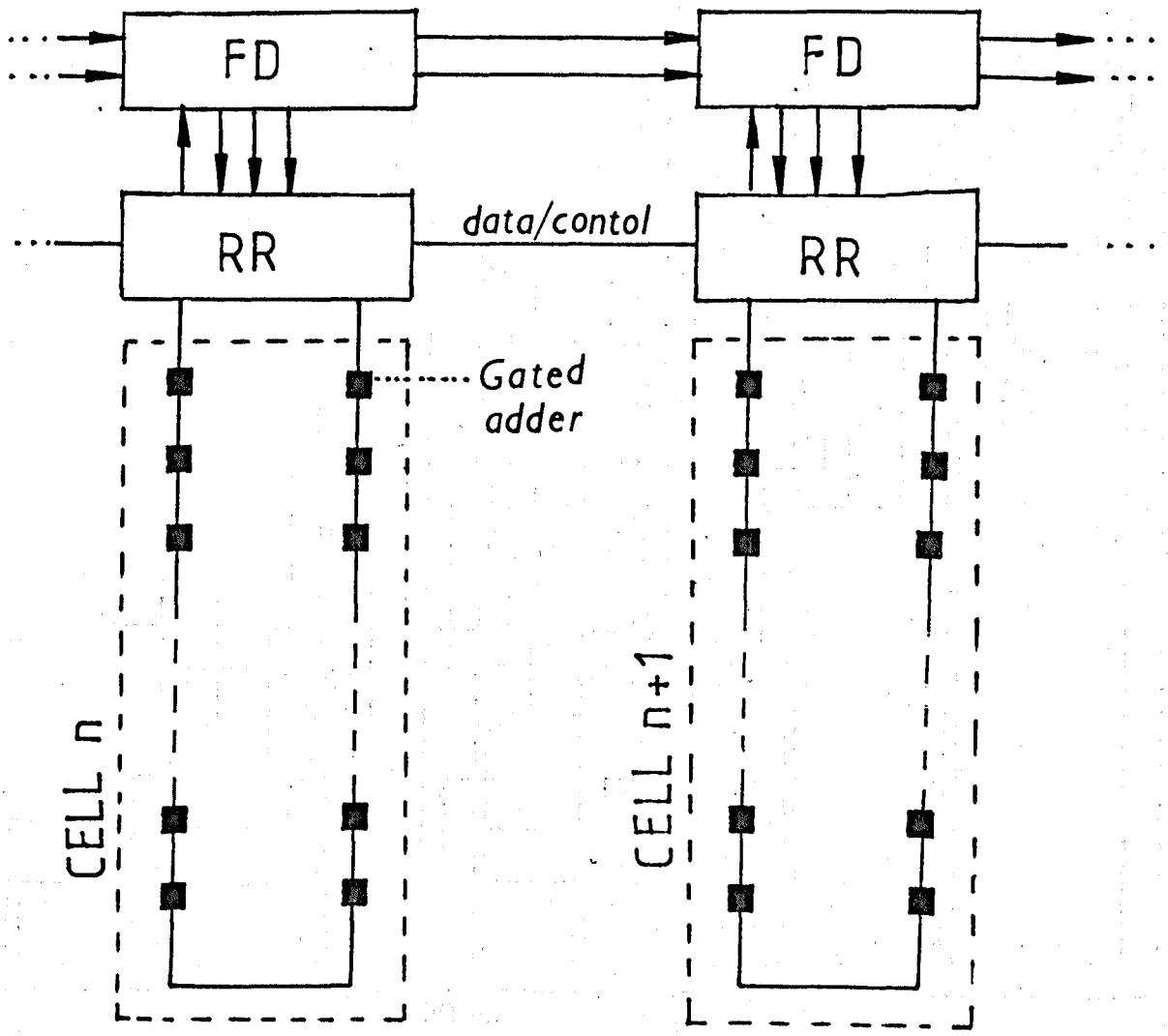


FIGURE 8.7 FAULT TOLERANT SPM.

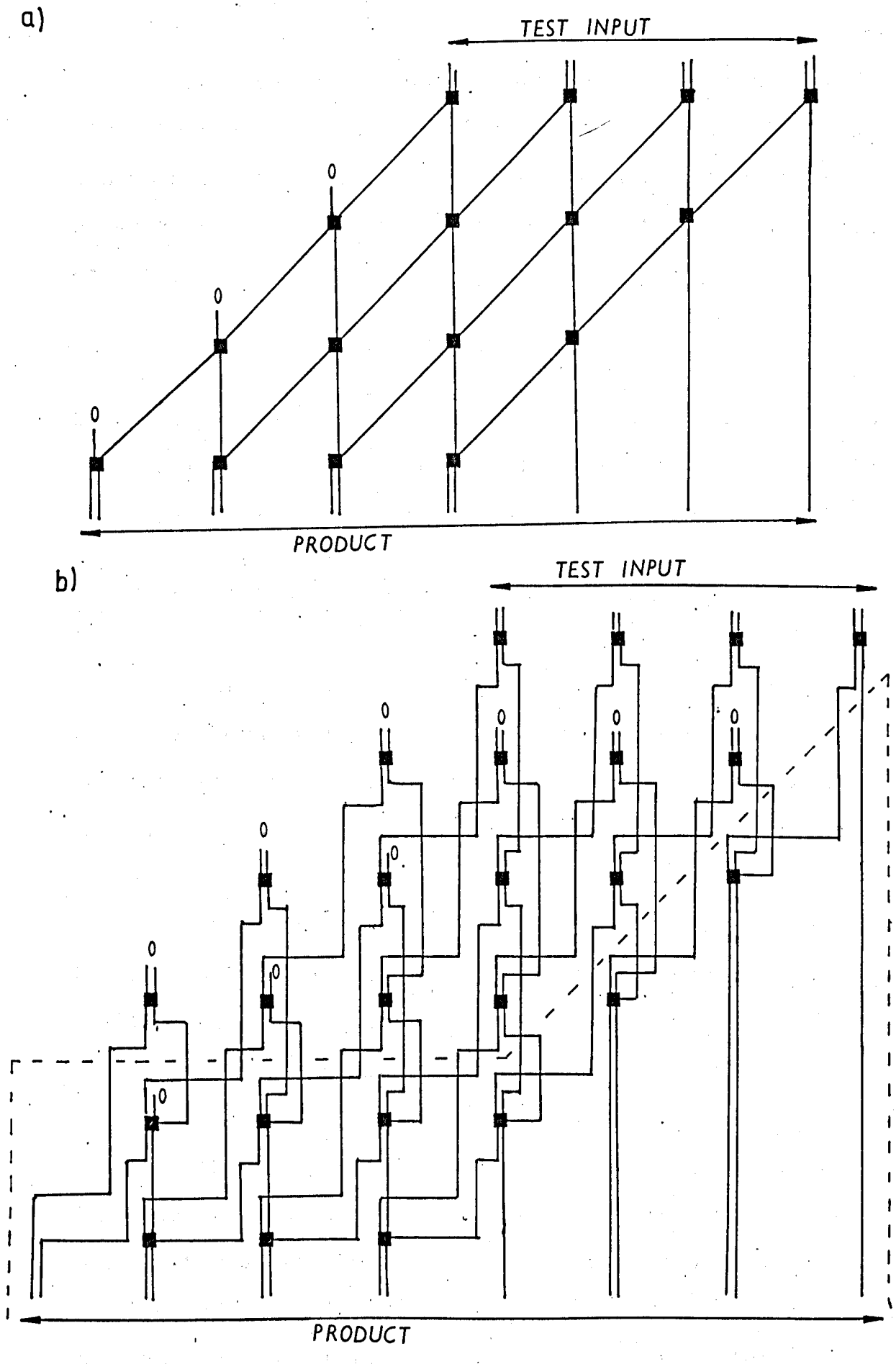


FIGURE 8.8 PARALLEL MULTIPLIER ARCHITECTURES.

Figure 8.7 shows an architecture for a fault tolerant SPM. The re-routing network (RR) allows input of test data and can bypass a cell during normal operation. The fault detecting network (FD) controls the RR and finds a faulty cell during test operation.

The number of stages chosen for a cell depend on the technology and yield figures. For very large networks a hierarchical scheme is desirable, because a single fault in the switching will destroy the operation.

8.4.5 Parallel multiplier.

Hardware implementations of the Parallel Multipliers (PM) have been available for a number of years. They offer either to increase the speed of one operation or to allow faster throughput.

The fastest method of multiplying two numbers is to use (3,2) counters. These combine the partial products either by row (Wallace tree) or column (Dadda scheme). However, their irregular interconnection schemes make them unsuited to VLSI.

The array multiplier, consisting of a diamond of gated full adders (black squares), is shown in figure 8.8a. Each row adds either 0 (if the multiplier for that row is 0) or a shifted version of the multiplicand. Though the delay is still O(n), the same as the SPM, it is much quicker in practice, and can be

pipelined (Ref.813).

8.4.6 New parallel multiplier.

For some applications it is not the throughput that matters but the absolute delay. In this section I propose a New Parallel Multiplier (NPM) that has a delay O(n^1/2).

The principle of this new scheme is shown in figure 8.8b. Instead of connecting the output of the gated adder to the next cell, it is connected to the cell n^1/2 below. To accumulate the result extra full adders (below dotted line) are used. These extra cells have connections from their nearest neighbours.

For this small example, the delay in the NPM is no better than the original PM. However, for larger operands, the new scheme has less gate delays. The NPM has more gate delays than Dadda's and Wallace's, but it is more regular.

8.4.7 Testing the PM.

It has been suggested that the conventional carry-save array multiplier is not testable with a fixed number of inputs: that is the number of test inputs is dependent of the size of the array (Ref.814). However, with control of the sum and carry inputs at the top of the array, a series of inputs can be found. Table 8.3 shows a series of test inputs for an eight bit multiplier, which fully exercise each cell (gated full adder). The sequences are extended for bigger arrays.

MULTIPLICAND	MULTIPLIER	CARRY-IN	SUM-IN
FF	00	00	00
FF	00	00	FF
FF	FF	FF	FF
FF	FF	FF	00
FF	AA	FF	00
FF	55	00	FF
AA	AA	FF	FF
55	55	00	00

TABLE 8.3 PARALLEL MULTIPLIER TEST INPUT

8.4.8 Performance comparison: Multipliers.

PARAMETER	SPM	NSPM	PM	NPM
NO. LOGIC CELLS	c	c	c	c
NO. WIRE CELLS	c	c	c	c
LOGIC AREA	n	n	n^2	n^2
WIRING AREA	n	n	n^2	$n^{5/2}$
I/O PORTS	c	c	n	n
TESTING TIME	c	c	c	c
TESTING AREA	c	c	n	n
GATE DELAY	n	n	n	$n^{1/2}$
WIRING DELAY	c	c	c	$n^{1/2}$
LOADING (Note 1)	c	c	n	n
PIPELINE AREA	-	-	c	c

Note 1. For pipelined operation the PM loading is O(c) not O(n).

TABLE 8.4 ARCHITECTURE COMPARISON TABLE

Table 8.4 demonstrates that, for quick and comparitively cheap multiplication, the NSPM is preferred. For fast throughput the pipelined PM offers optimum performance. However, if absolute delay is the primary requirement, the NPM is a better alternative.

8.5 DIVISION.

The division operation involves of finding the quotient (Q) and remainder (R), of a numerator/dividend (A) and denomonator/divisor (D). These four parameters satisfy the equation

$$A = (Q * D) + R \qquad \ldots 8.11$$

Where R is less than D and of the same sign as A. For a given 2n bit A and n bit D, there is a unique n+1 bit Q and n bit R.

The paper and pencil method of division is based on a shift and conditional subtraction technique. For an n of three this can be represented as shown below:

```
                      q3     q2     q1     q0
          ------------------------------------
d2 d1 d0 )  a5     a4     a3     a2     a1     n0
           q3.d2  q3.d1  q3.d0
                  q2.d2  q2.d1  q2.d0
                         q1.d2  q1.d1  q1.d0
                                q0.d2  q0.d1  q0.d0
                              ------------------
                                r2     r1     r0
```

Division and multiplication are in many repects dual operations. Division superficially resembles the shift and add

method of multiplication. However, division requires each operation to be completed before the next begins. This introduces a sequential ordering into division which is not present in multiplication.

The inherent serial nature of division, coupled with its relatively rare occurence in arithmetic operations, has meant there have been relatively few dedicated dividers. Most hardware uses some iterative technique (Ref.803), making use of a fast multiplier. Though this has proved successful, it is not well suited to VLSI. Not only is the throughput limited, but the hardware is relatively complex.

Dedicated dividers are usually based on fast adders. The most popular of these has been the non-restoring and SRT methods (Ref.803). Both of these methods dispense with the need for comparison, but do not overcome the O(n*logn) worst case delay.

8.5.1 Principle of the serial-parallel divider.

The new Serial Parallel Divider (SPD), though similar to the conventional SPM, has a more complex operation. Therefore before describing its operation, the principle will be described using a 'reverse SPM'. Though this analogy is useful it does not directly lead to a practical realization.

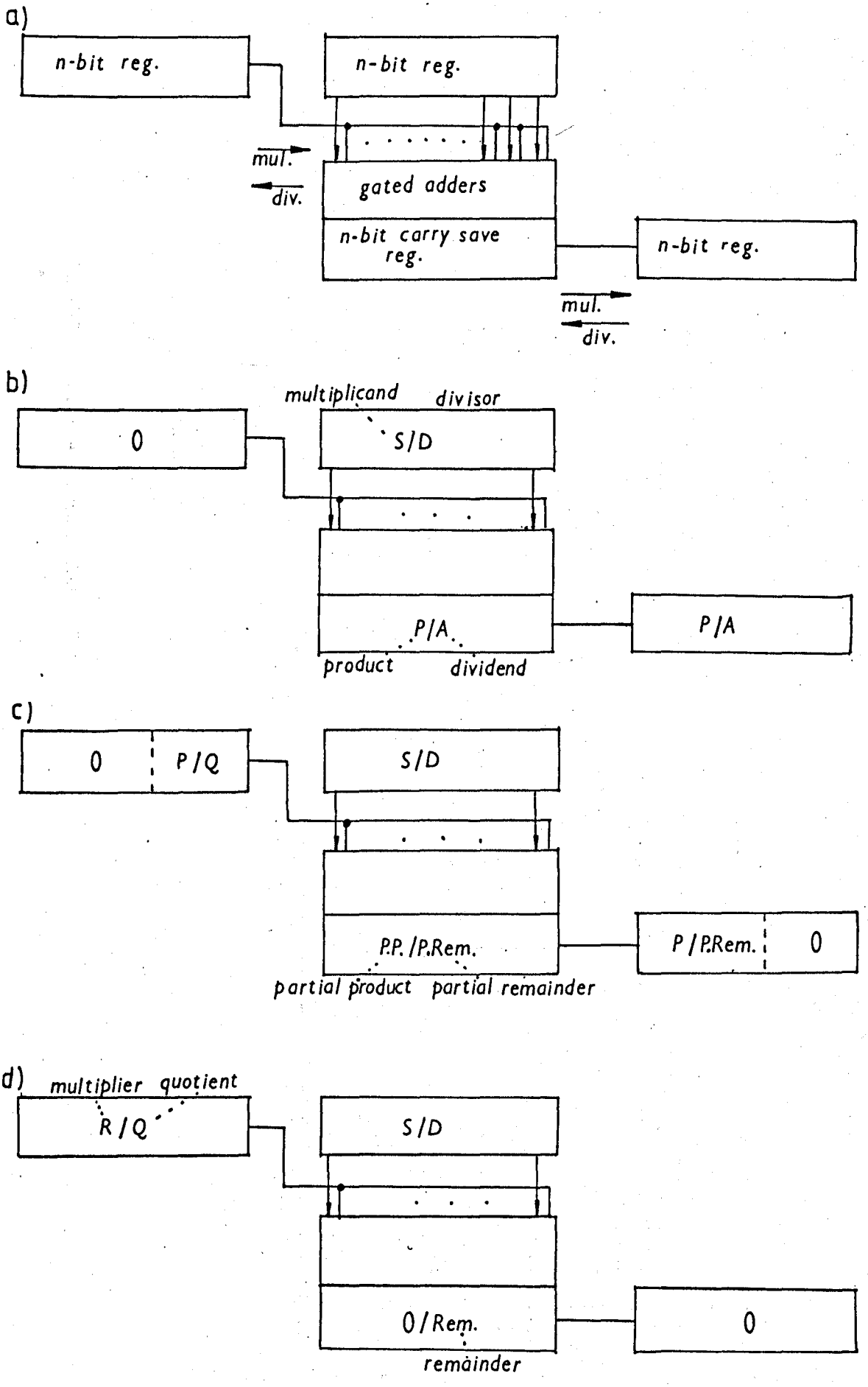


FIGURE 8.9 PRINCIPLE OF THE SPD.

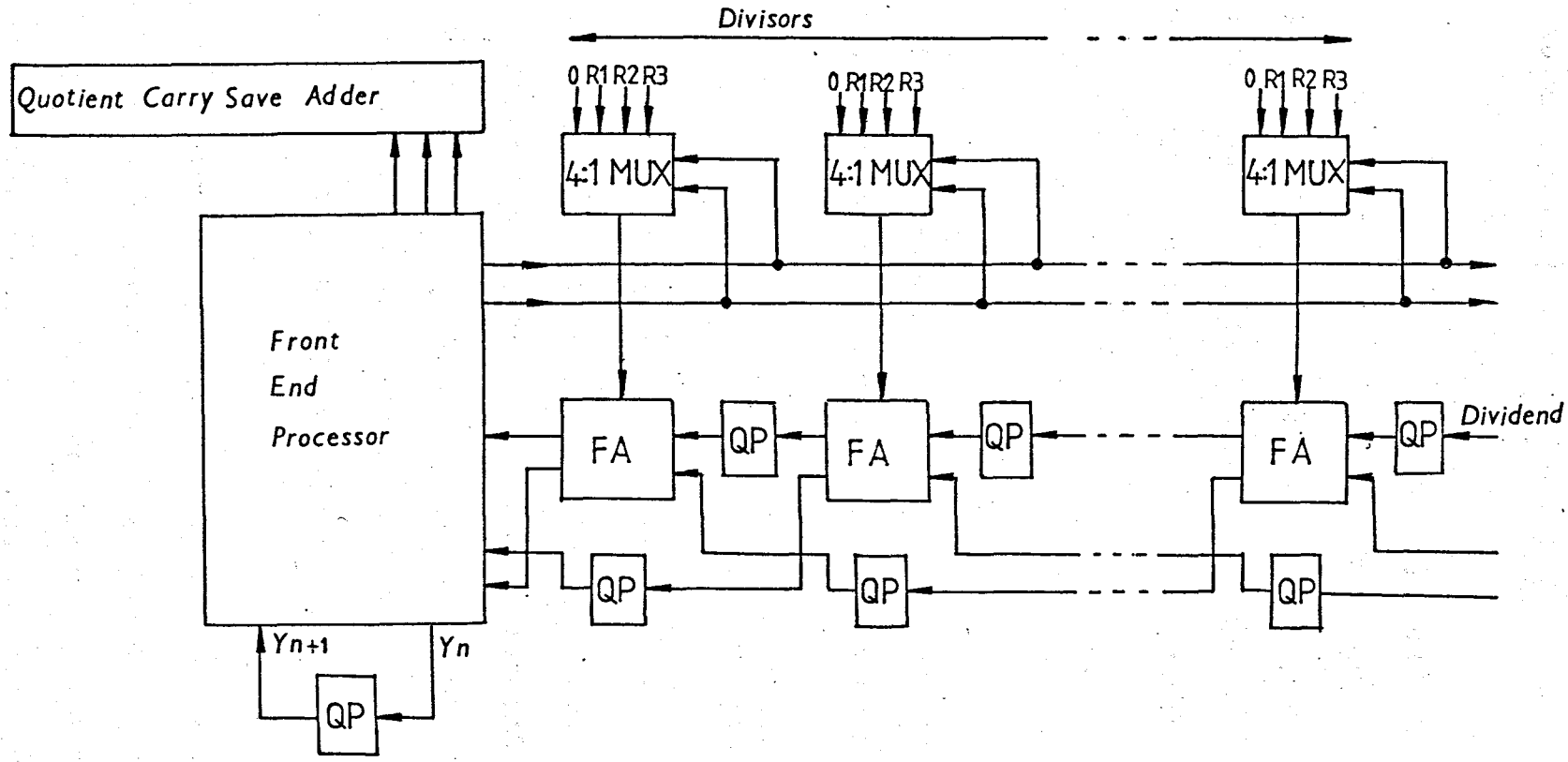


FIGURE 8.10 A PRACTICAL SPD.

Figure 8.9a shows our imaginary 'reverse SPM', with its reversible data flow. After n clock cycles the state of the conventional SPM is shown in figure 8.9b. If we now reverse the clock, figure 8.9c shows the position after k clock cycles. Finally after n clock cycles (see figure 8.9d) we are in the starting state of the SPM and the final position of the SPD.

Those familiar with division will realize the bug in our imaginary SPD: how did we know the value of the quotient / multiplier. To find this value we need to know whether the divisor is greater than the contents of the carry save register, which requires a time consuming comparison. To get around this problem the architecture has to be modified.

8.5.2 A practical SPD.

Figure 8.10 shows two new features not present in figure 8.9. Firstly, there is a Front End Processor (FEP) on the carry save adder. Secondly, there is more than one possible multiple of the divisor.

Consider first the action of the FEP. It receives a sum and carry from the most significant stage of the carry save register, and the carry from the second most significant stage. The sum of these inputs (X) can vary between four and zero. Internally the FEP has a carry save (Y).

The value of X and Y are used to determine the value of

the quotient, and the multiple of the divisor that needs to be subtracted from the partial product register. In order not to overflow the FEP, it is important to have sufficient combinations of possible numbers to subtract. Including the zero, four combinations are needed. For each possible value of X and Y table 8.4 shows the resulting action.

INPUTS			OUTPUTS	
Xn	Yn	Yn+1	quotient	'divisor'
000	0	0	0	0
000	1	0	Q1	R1
001	0	1	0	0
001	1	1	Q1	R1
010	0	0	Q1	R1
010	1	0	Q2	R2
011	0	1	Q1	R1
011	1	0	Q2	R2
100	0	0	Q2	R2
100	1	0	Q3	R3

TABLE 8.4 FRONT END PROCESSOR TRUTH TABLE.

The operation of the SPD is more complex than the other schemes discused before: but in operation it is not dissimilar to the SRT algorithm. Again, instead of a full comparison, just a few bits are used to determine the number to be subtracted. However, with the SPD only a carry save subtraction is performed. The values of Qx and Rx are given by:

$$R1 = 2^{n+1} \bmod D \quad , Q1 = 1\left(2^{n+1} / D \right)$$

$$R2 = 2^{n+2} \bmod D \quad , Q2 = 1\left(2^{n+2} / D \right)$$

$$R3 = \left(2^{n+1} + 2^{n+2}\right) \bmod D \quad , Q3 = 1\left(\left(2^{n+1} + 2^{n+2}\right) / D \right)$$

8.5.8 Performance comparison: Divders.

PARAMETER	SRT	SPD
NO. LOGIC CELLS	c	c
NO. WIRE CELLS	c	c
LOGIC AREA	n	n
WIRING AREA	n	n
I/O PORTS	n	n
TESTING TIME	c	c
TESTING AREA	c	c
GATE DELAY	n*logn	n
WIRING DELAY	c	c
LOADING	n	n
PIPELINE AREA	-	-

TABLE 8.5 ARCHITECTURE COMPARISON TABLE

Just two schemes for division are compared. The SRT and SPD are similar in cost terms, but the SPD introduced here is faster. One problem which the SPD has not overcome is the loading problem. This would limit the performance of larger dividers.

8.6 MULTIOPERAND ADDITION.

The technique of conditionally adding n+1, n-bit numbers is described here. The paper and pencil method of adding three numbers A, B, and C, if their associated constants (K) are

present, is given below:

```
            k0.a2 k0.a1 k0.a0
            k1.b2 k1.b1 k1.b0
            k2.c2 k2.c1 k2.c0
-----------------------------
  s4    s3    s2    s1    s0
```

The number of bits in the sum is given by n + l(n).

The architectures used for multiplication can easily be modified for multioperand addition (MOA). Figures 8.11 and 8.12 show the timing and architecture for adding n+1 numbers. This is the same as the SPM, except for the reversal in direction of the partial accumulation line (A0) and one operand coming from an external source (BI).

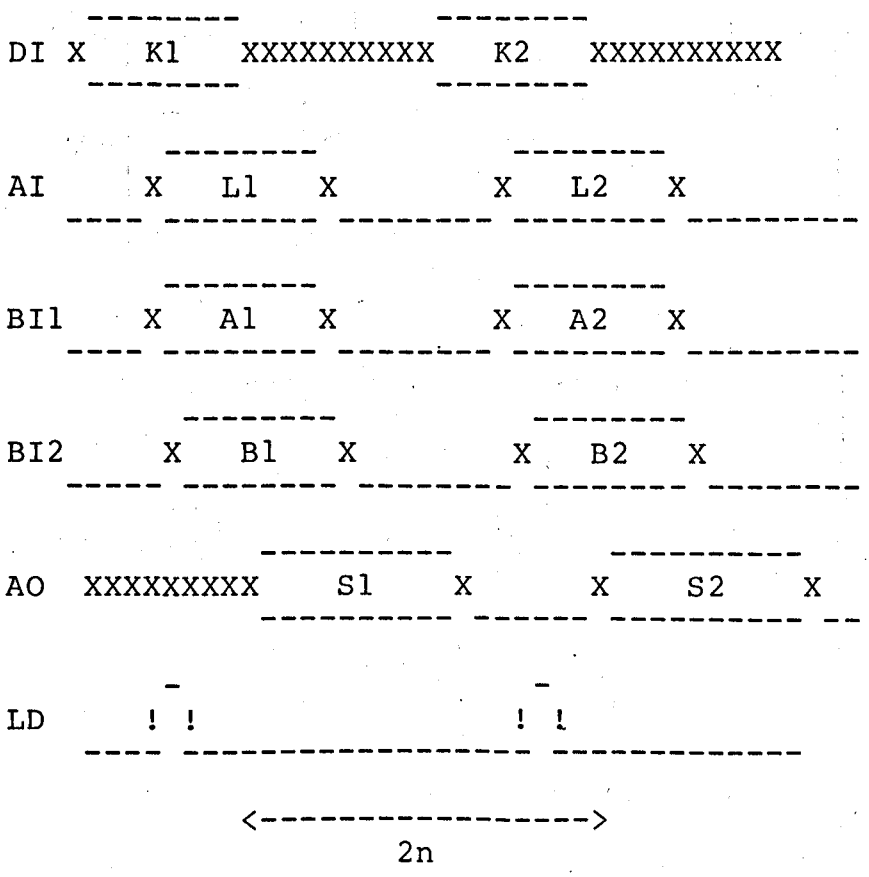


FIGURE 8.11 TIMING FOR THE MOA.

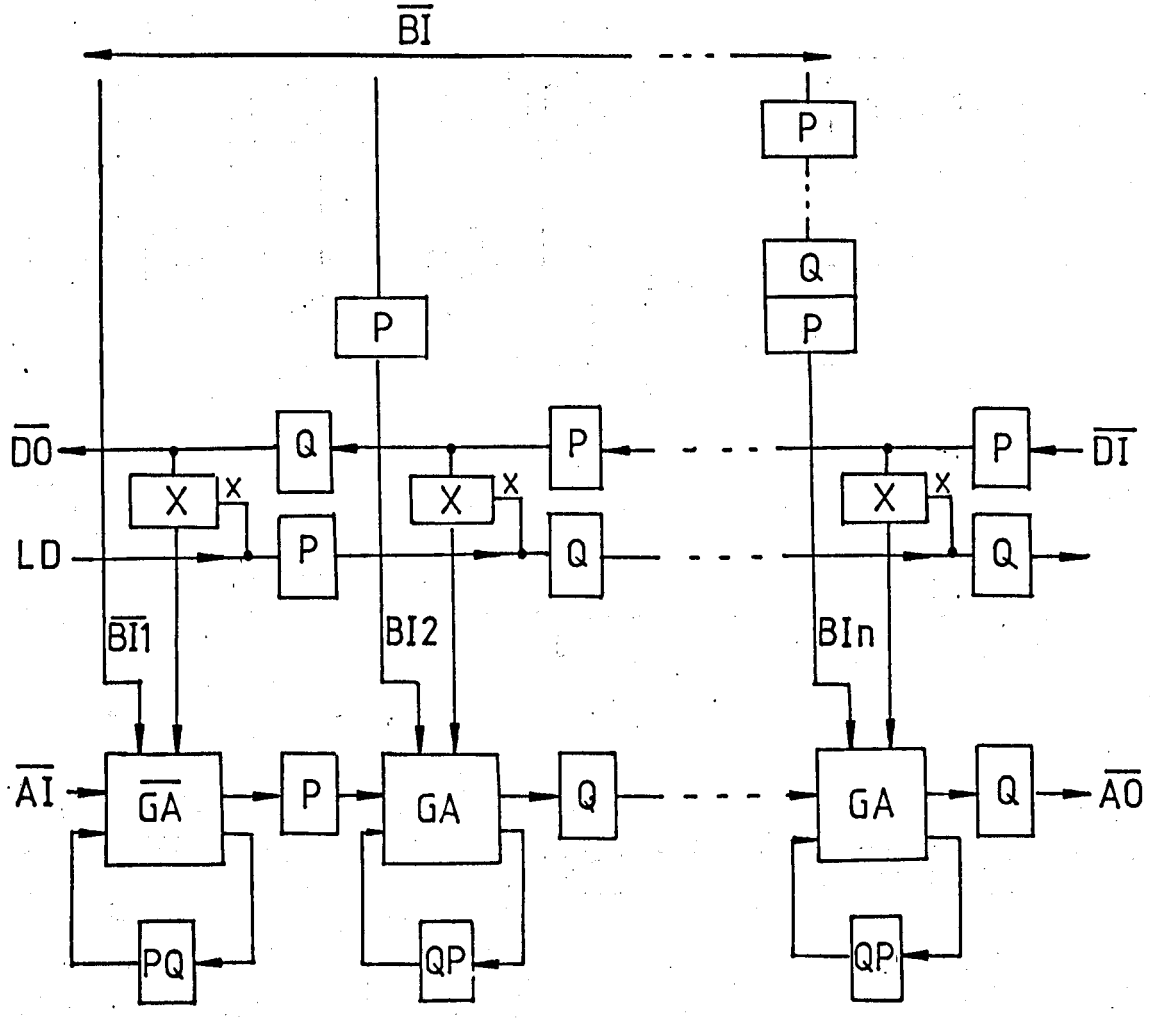


FIGURE 8.12 MULTIOPERAND ADDITION ARCHITECTURE.

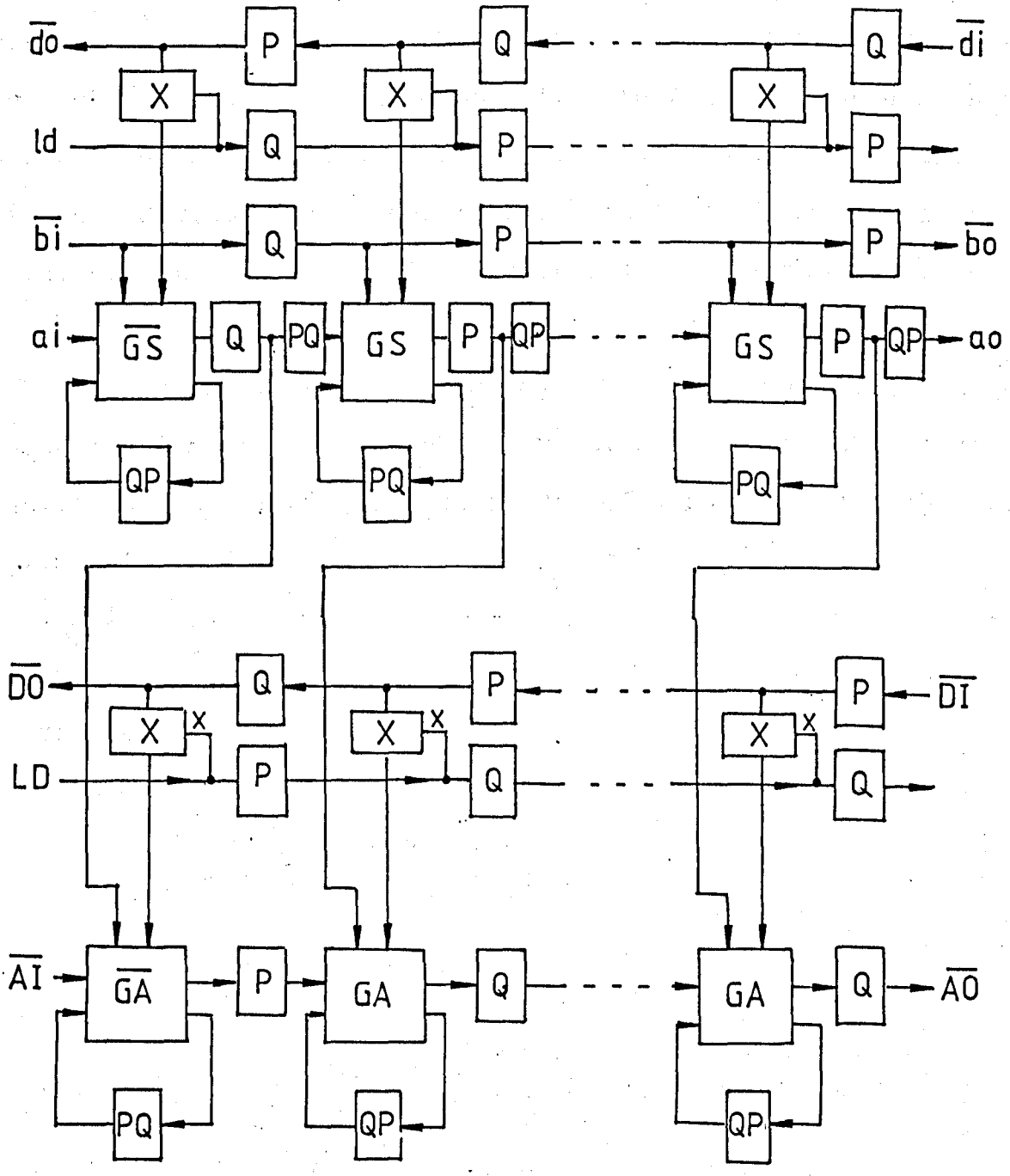


FIGURE 8.13 MODULAR REDUCTION ARCHITECTURE.

8.7 MODULAR REDUCTION.

The SPD suffers a number of serious drawbacks compared with the SPM. Not only is there a need for global communication, but there is additional complexity. In this section a method of overcoming these limitations will be described, which will work if the divisor is known in advance. This might seem a strange idea, but it is often applicable in practical situations.

In order to split a system into parallel segments, modular arithmetic is often performed. But finding the modulus is equivalent to finding the remainder of a number (A) with a fixed divisor (D). If:

$$A = \sum_{j=0}^{2n-1} a_j * 2^j$$

We can write:

$$\begin{aligned} A \bmod D = (& a_{2n-1} .[\, 2^{2n-1} \bmod D\,] \\ & + a_{2n-2} .[\, 2^{2n-2} \bmod D\,] \\ & + \ldots \\ & + a_{n} .[\, 2^{n} \bmod D\,] \\ & + a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ldots + a_0.2^0 \\ &) \bmod D \end{aligned}$$

Let $R_j = 2^j \bmod D$

$$\text{and } L = a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ldots + a_0.2^0$$

Therefore we can write:

$$A \bmod D = (a_{2n-1}.R_{2n-1} + \ldots + a_n.R_n + L) \bmod D$$

The above expression contains the variables R, which are independent of the divisor (A). For a given modulus D, R is therefore fixed. The expression can thus be simplified to one of conditionally adding n n-bit numbers (R) to another n-bit number (L). This is exactly the architecture discussed in the last section.

The value of R could be precalculated, but this would lead to an O(n^2) figure for logic area. Two methods of calculating R, using previously worked out values, are possible:

a) Working up: $R_j = (2.R_{j-1} - u_j.D)$ ...8.14

Where $R_{n-1} = 2^{n-1}$

and the u_j are precalculated.

b) Working down: $2.R_j = (R_{j+1} + v_j.D)$ ...8.15

Where $v_j = 1$ if R_{j+1} is odd, else 0

and R_{2n-1} is precalculated.

The values of u and v are the same. But because we are working backwards in b) we need not precalculate them. However,

in the latter scheme we must precalculate the most significant R.

Figure 8.13 shows a hardware implementation of a Modular Reducer (MR). It uses two carry save adder chains. The top chain calculates the values of R using method a. The bottom chain accumulates the sum, which is an n + 1(n) bit number. Figure 8.14 shows the overall timing of the MR: assuming the constants (u) have already been loaded, via ld and di lines. The Amsb and Alsb refer to the most and least significant n bits of the numerator respectively. S is the reduced n + 1(n) bit number.

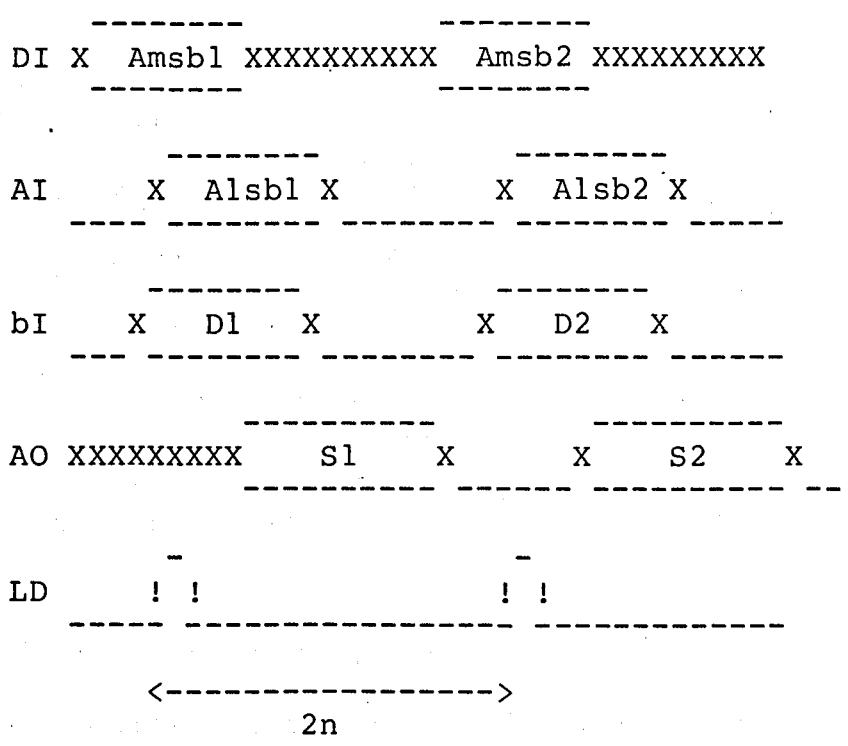


FIGURE 8.14 TIMING FOR THE MR.

8.8 SYNTHESIS OF A REQUIRED FUNCTION.

In the above sections we have described GVAs for basic

arithmetic operations. Does this help us to calculate more complex functions, such as Trigonometric, exponential and logarithmic? The answer to this question is yes.

Complex functions can be synthesized using techniques such as CORDIC (Refs.820-821). It is not possible to describe this complicated technique here, but the consequences of this algorithm are worth mentioning: with just a few cells and architectures it is possible to build any arithmetic architecture.

The architectures discussed in this chapter have all been either:

a) A tree (CLA, CBA, CSA).

b) A chain (MCC, SPM, SPD, MOA, MR).

c) An array (PM, NPM).

The combination of a small number of architectures with a few leaf cells, has numerous advantages. But perhaps the most important is its simplicity.

Design rules change frequently, requiring a new library of cells. This means it is not possible to use the same layouts for more than a few years. But by using just a few leaf cells it is possible to quickly redesign them. This simplicity allows high performance arithmetic units to be quickly designed.

To further speed up design, it is desirable to have the design automated. A simple approach would be to have a library of

predefined architectures. But more promising is the possibilty of generalizing these architectures onto a higher level of abstraction. It would then be possible for 'intelligent' software to automate the whole process of arithmetic design.

8.9 CONCLUSION.

This chapter has described Good VLSI Architectures (GVA) for the basic arithmetic functions. Using techniques such as CORDIC, more complex functions can be realized.

For each of the basic arithmetic operations a new architecture has been proposed. All architectures have very low complexity: in many cases using identical cells or architectures with minor modifications. By using asymptotic cost and performance measures, their benefits over 'ad hoc' designs will improve as technology advances.

A new adder (CLA) was proposed which is faster than any previously published. By modifying the architecture (CBA) it was shown how the area could be reduced with a small loss in performance.

Two new multiplier structures were proposed. The NSPM is similar to existing schemes but, because it removes any global communication, it has better asymptotic performance. A NPM was proposed, which is ideal for very fast calculation.

A division architecture with O(n) delay was described. This is a faster asymptotic performance than any previosly published. Though requiring more random logic, its asymptotic complexity is low.

Two other original arithmetic architectures were also given. Firstly, for adding n+1 numbers simultaneously: with an architecture almost identical to the NSPM. Secondly, a scheme for finding the modulus, based on two serial pipelines. If applicable, it is preferable to the general divider: since it has less loading problems.

This use of just a few cells makes it possible to use full custom architectures cheaply and quickly. It would be particually attractive if a software 'shell' was written to automate the task.

The incorporation of error-correction techniques were discussed in relation to the SPM. This appears to be an important technique for breaking the traditional link betwwen yield and area. However, this topic deserves more detailed investigation, particually as regards keeping complexity low.

8.10 REFERENCES.

801 H.J.HINDIN.

Digital signal processing moves into high gear.

COMPUTER DESIGN, PP.61-77, 15TH OCT. 1984.

802 ADVANCED MICRO DEVICES.

AM29500 - a new bipolar LSI family for array/signal processing.

AMD, P.O.BOX 453, SUNNYVALE, CALIFORNIA 94086.

803 C.R.CHAMPLIN & J.E.PRIOSTE.

Array-based logic boosts system performance.

COMPUTER DESIGN, PP.93-100, MAY 1984.

804 R.E.OWEN.

VLSI architectures for digital signal processing,

VLSI DESIGN, PP.21-28, JUN.1984.

805 K.HWANG.

Computer arithmetic: principles, architecture, and design.

WILEY, 1979.

806 O.SPANIOL.

Computer arithmetic.

WILEY, 1981.

807 J.B.GOSLING.
Design of arithmetic units for digital computers.
MACMILLAN, 1980.

808 R.P.BRENT & H.T.KUNG.
A regular layout for parallel adders.
IEEE T. COMPUTERS, VOL.31, NO.3, PP.260-264, MAR.1982.

809 E.BRAUN.
Digital computer design.
ACADEMIC PRESS, 1963.

810 D.HEMPEL K.E.MCGUIRE & K.J.PROST.
CMOS/SOS serial parallel multiplier.
IEEE J. SOLID STATE CIRCUITS, PP.307-313, 1975.

811 I.N.CHEN & R.WILLONER.
An O(n) parallel multiplier with bit sequential i/o.
IEEE T. COMPUTERS, VOL.28, PP.721-727, OCT.1979.

812 N.R.STRADER & V.T.RHYNE.
A canonical bit-sequential multiplier.
IEEE T. COMPUTERS, VOL.31, PP.791-795, AUG.1982.

813 A.J.KESSLER & J.K.PATEL.
Reconfigurable parallel pipelines for fault tolerance.
IEEE INT. CONF. ON CIRC. & SYST. ICCC'82, NEW YORK, PP.118-121, SEP-OCT.1982.

814 H.T.KUNG & M.S.LAM.

Fault-tolerance and two-level pipelining in VLSI systolic arrays.

CONFERENCE ON ADVANCED RESEARCH IN VLSI, PP.184-193, 1984.

815 N.R.STRADER & T.J.BROSNAN.

Error detection for serial processing elements in highly parallel VLSI processing architectures.

CONFERENCE ON ADVANCED RESEARCH IN VLSI, PP.184-193, 1984.

816 M.J.DAY.

Comparison of fault-tolerant architectures for linear pipelines.

VHPIC MEMO NO.20, GEC HRC, MIDDX. HA9 7PP, ENGLAND, FEB.1983.

817 J.V.MCCANNY & J.G.MCWHIRTER.

Completely iterative, pipelined multiplier array suitable for VLSI.

IEE PROC., VOL.129, PT.G, NO.2, PP.40-46, APR.1982.

818 D.A.HENLIN, M.T.FERTCH, M.MAZIN & E.LEWIS.

A 25MHz 16 bit x 16 bit pipelined multiplier.

RAYTHEON, BEDFORD, MASSACHUSETTS 01730.

819 J.IWAMURA, K.SUGANUMA, S.TAGUCHI, M.KIMURA & K.MAEGUCHI.

A 16-bit CMOS/SOS multiplier-accumulator.

IEEE INT. CONF. ON CIRC. & SYST. ICCC´82, NEW YORK, PP.151-154, SEP-OCT.1982.

820 J.E.VOLDER.

The CORDIC trigonometric computing technique.

IEEE TRAN. ELEC. COMP., VOL.9, PP.222-231, SEP.1960.

821 J.S.WALTHER.

A unified algorithm for elementary functions.

SJCC, PP.379-385, 1971.

CHAPTER 9

9.1 INTRODUCTION.

This chapter will describe the design of the MA743 Advanced Cipher Processor (ACP). The ACP implements a 512 bit RSA algorithm (see section 1.6.3) with Multibus interface. The design is in 2.5 micron, double level metal, bulk CMOS. This work was carried out by the author while at GEC Hirst Research Centre (Ref.901).

Public Key Encryption algorithms are based on simple arithmetic operations. However, all PKCs have one distinguishing factor compared with other DSP operations: the need for very large operands. This security restraint on the minimum size of operands mean that they are very computationally intensive.

The computational intensive PKCs are too slow if implemented on a microprocessor and too costly for a TTL breadboard. However, VLSI offers an ideal solution. The size of the operands make the PKCs well suited to the arithmetic designs of chapter 8.

The documentation is not intended to be comprehensive. Some details of design, such as the microcode, have been omitted. However, all major sections of the ACP will be covered using the hierarchical design style of chapter 6.

9.2 HISTORY.

The proposal to build an ACP was first put forward by the author in 1981. This lead to a more detailed proposal in 1982 (Ref.902). Three alternative cryptosystems were nominated.

Two of the proposed cryptosystems, the MH-knapsack (see section 1.6.2) and the DH-key distribution scheme (see section 1.6.1) have proven popular, with several companies using them. However, continued improvements in cryptanalytic techniques have left their long term security in doubt. The question of security was also the reason for rejecting our own cryptosystem (see chapter 3).

To call a cryptosystem secure is relative, since no absolute guarantee of security has ever been given. However, probably the best test is to publish the algorithm and wait. Because our system has not had this cryptanalysis, it would have been unwise to invest so much in it. Therefore, the RSA scheme (see section 1.6.3) was chosen.

It was envisaged that the mass market for these chips would not exist until the late 1980's. For this reason, together with the contemporary nature of the proposed technology and the novel architectures used, a test design was undertaken.

This buffer between design and marketing was the reason for going for a single chip implementation. Although it would

have low yield under present technology, when required it could be economically fabricated. The test design could have all the features of the final chip, without the additional logic of a multichip system.

The work was intented to form the practical part of my PhD, therefore, I worked alone. This solo approach had mixed consequences. The major drawback was the lack of someone to check for errors. However, on the positive side it allowed me to visualize the whole design cycle, which compensated for any delay caused by small errors.

9.3 SYSTEM DESIGN.

The operation of the ACP is described in ISPS (see appendix A). This provided an effective way of debugging the initial design. After the design was proven, a provisional data sheet (see appendix B) was produced. Figures 9.1-9.4 show blow-ups of the data sheet diagrams.

Figure 9.1 illustrates the functional block diagram of the ACP. The various components of this diagram are described below.

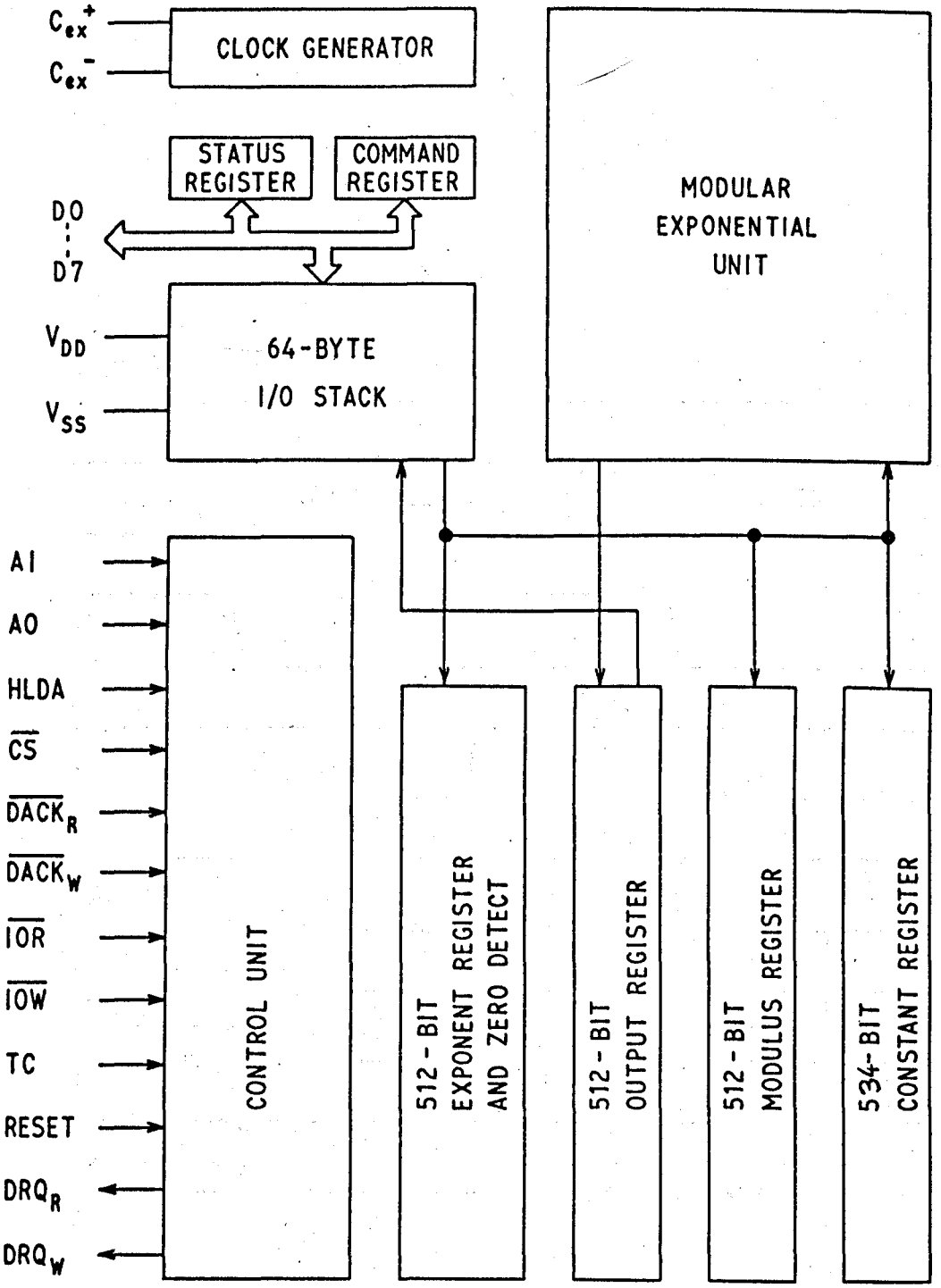


FIGURE 9.1 ACP BLOCK DIAGRAM.

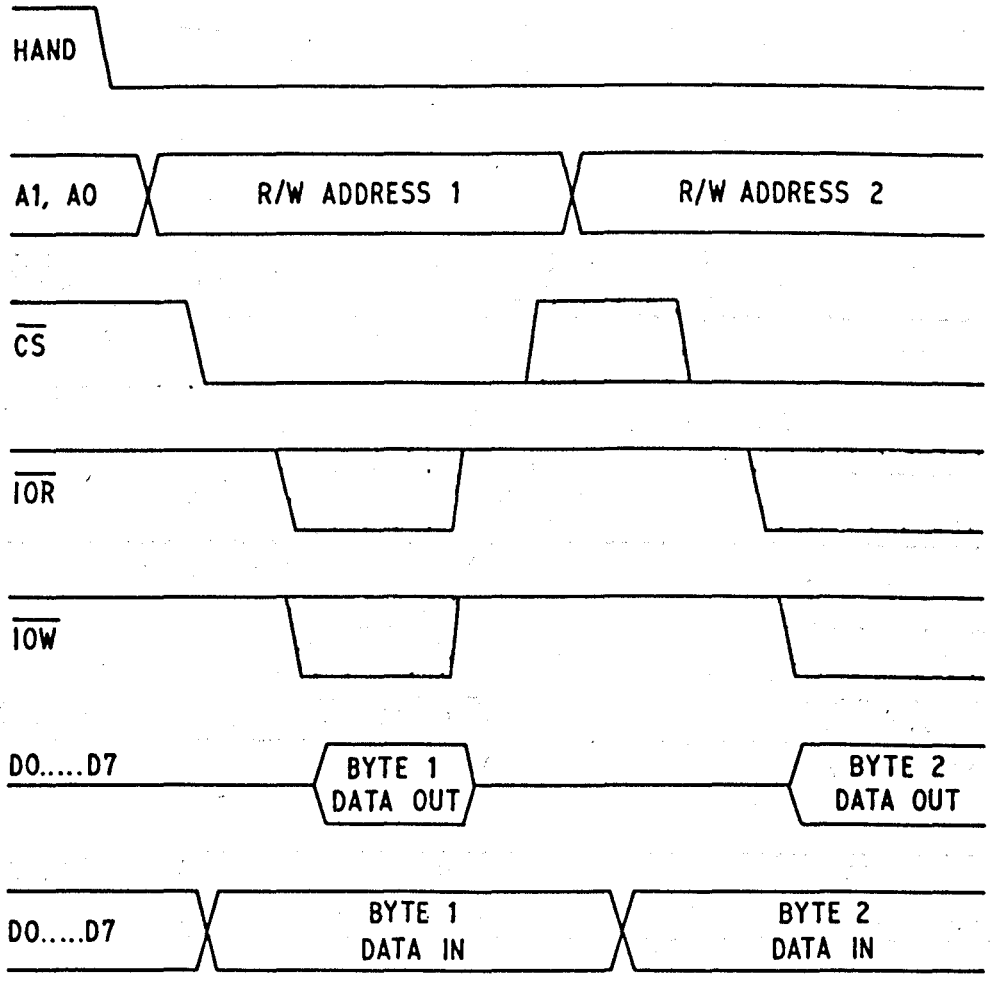


FIGURE 9.2 DIRECT TRANSFER OF DATA.

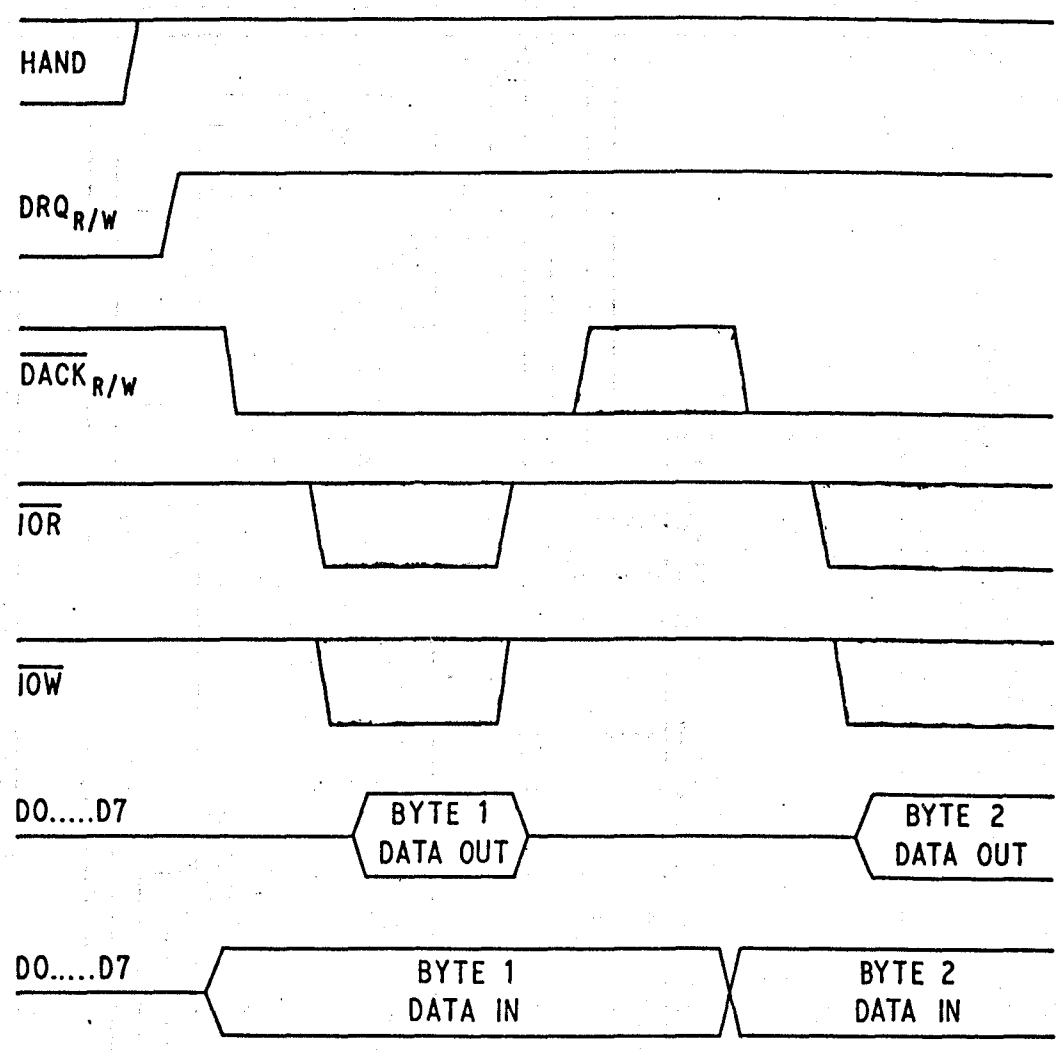


FIGURE 9.3 TRANSFER OF DATA IN HANDSHAKE MODE.

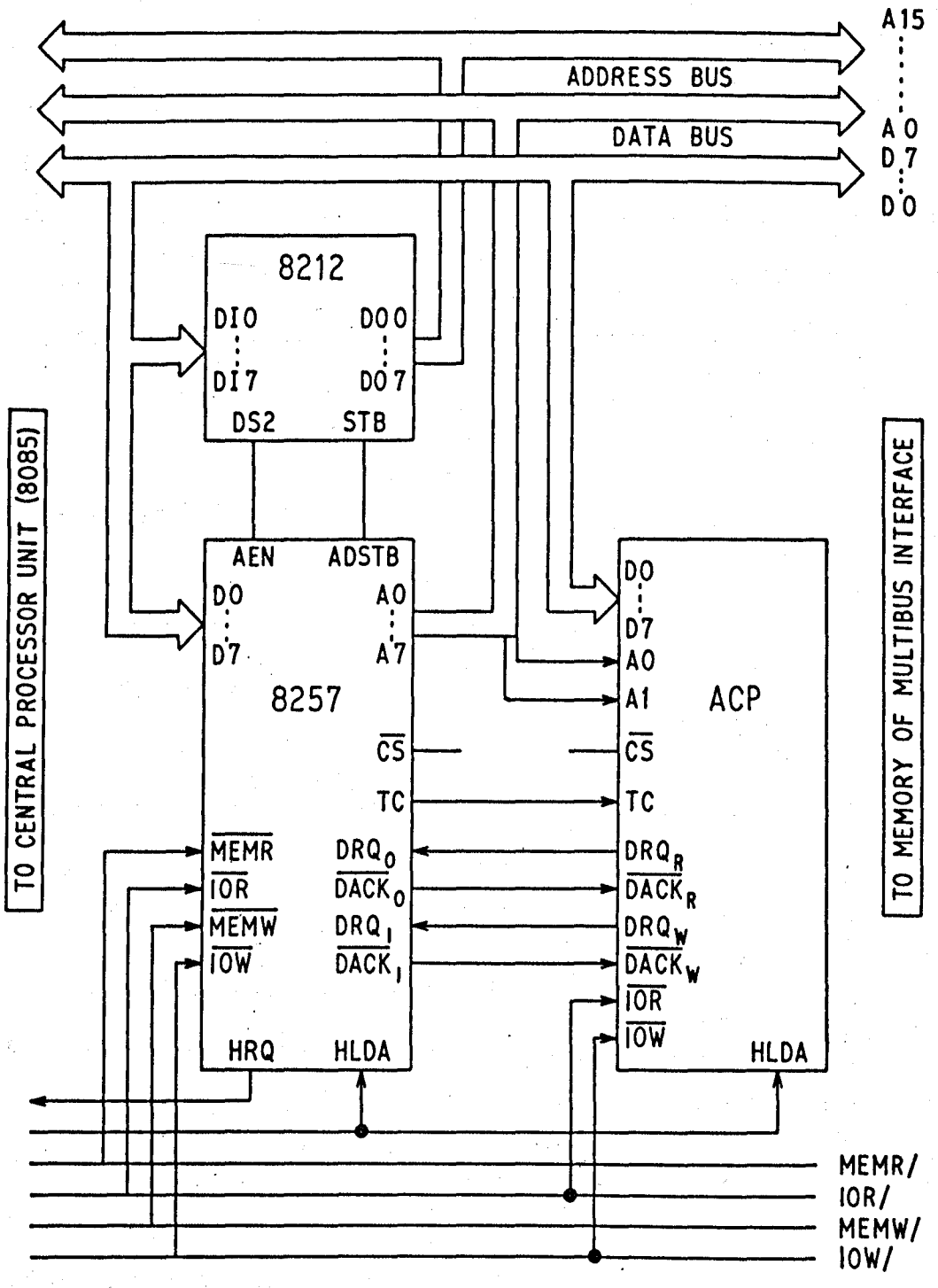


FIGURE 9.4 SYSTEM INTERFACE SCHEMATIC.

9.3.1 Modular Exponential Unit.

The heart of the ACP is the Modular Exponential Unit (MEU). It takes the contents of the stack (M) and raises it to the power of E, where E is the contents of the exponent register. The result (C) is then reduced mod X, where X is the contents of the modulus register. C is then left in the output register until the stack is free. If e(i) is the i'th bit of E, the formula used is:

```
C = 1;
FOR i = 0 TO 511
BEGIN
  If e(i) = 1 THEN C = C * M mod X;
  M = M * M mod X;
END.
```

Notice that the reduction is performed after each operation. This stops overflow of the registers holding M and C.

This binary method for evaluating powers is not optimal in terms of the number of multiplications. The power tree method is faster for small n, while the factor method is better for powers with a large number of factors (Ref.114). However, as a general method the binary method is always close to the optimum, and for large n the reduced complexity is important.

9.3.2 Interface.

An important part of any design intended for the

commercial market is the system interface.

The throughput of the ACP is not input/output limited, making it possible to use a standard 24-pin Dual In Line (DIL) package. All data is communicated through an 8-bit bus, either under control of a DMA or microprocessor. Figures 9.2 and 9.3 show the timing diagrams for transferring data in direct and handshake modes respectively.

The chip contains all the signals necessary to interface directly to a Multibus (Intel trademark) system. Figure 9.4 shows a typical system interface.

9.3.3 Synchronization.

The MEU has been optimized for high performance, so there are no global signals or off-chip communication. This allows the MEU to operate at much faster clocking speeds than the rest of the system. However, in order to take full advantage of this potential, it is vital not to have the delay through the clock drivers. By having an asynchronous internal clock the delay through the clock drivers is immaterial, since it is not relative to anything else.

The ACP has two independent processes: the external process, concerned with conveying data between the internal stack and off chip devices; and the internal process, which transfers data between the stack and the MEU. The two operations are

asynchronous.

The asynchronous nature of the external and internal processes are reflected by the use of the 'process' qualifier in the ISPS description (see appendix A). Communication between the two processes is via the status register. The data sheet gives a fuller description: showing how it is used by external devices to avoid conflicting use of the stack.

In the timing diagrams (figures 9.2-9.3) there is no system clock. The internal process is timed by a quartz crystal controlled oscillator (see section 9.5.3). It 'pops' data off the stack into the MEU and 'pushes' data onto the stack from the output register. The external process regulates the reading from and writing to the ACP. It 'pushes' data onto the stack and control register and 'pops' data off the stack and status register.

If a signal from one of the processes violates the input constraint of the control unit, it can leave the circuit in a non-stable state. Though there are no absolute ways of guaranteeing synchronization (Ref.903), by allowing enough response time it is possible to reduce this danger. The ACP control unit allows over ten microseconds before a sampled signal is allowed to change a state. This reduces the chance of failure to virtually zero, without affecting the performance of the ACP.

9.3.4 CONTROL UNIT.

The control unit is responsible for switching data on the serial internal buses, and parallel external bus. Using its own internal state machine, and signals from internal and external components, it ensures no conflicts exist.

The decision to go for a simple interface meant greatly increasing the chip complexity. Though adding only a small amount to manufacturing costs, approximately half the design costs were taken up with the control unit, the switching logic and its associated microcode.

9.3.5 Stack.

The name stack is something of a misnomer, since it is functionally a first in first out (FIFO) buffer. Data is written at the top of the 64-byte FIFO and read at the bottom. Extenally data is read and written least byte first. Internally data is read and written least significant bit first.

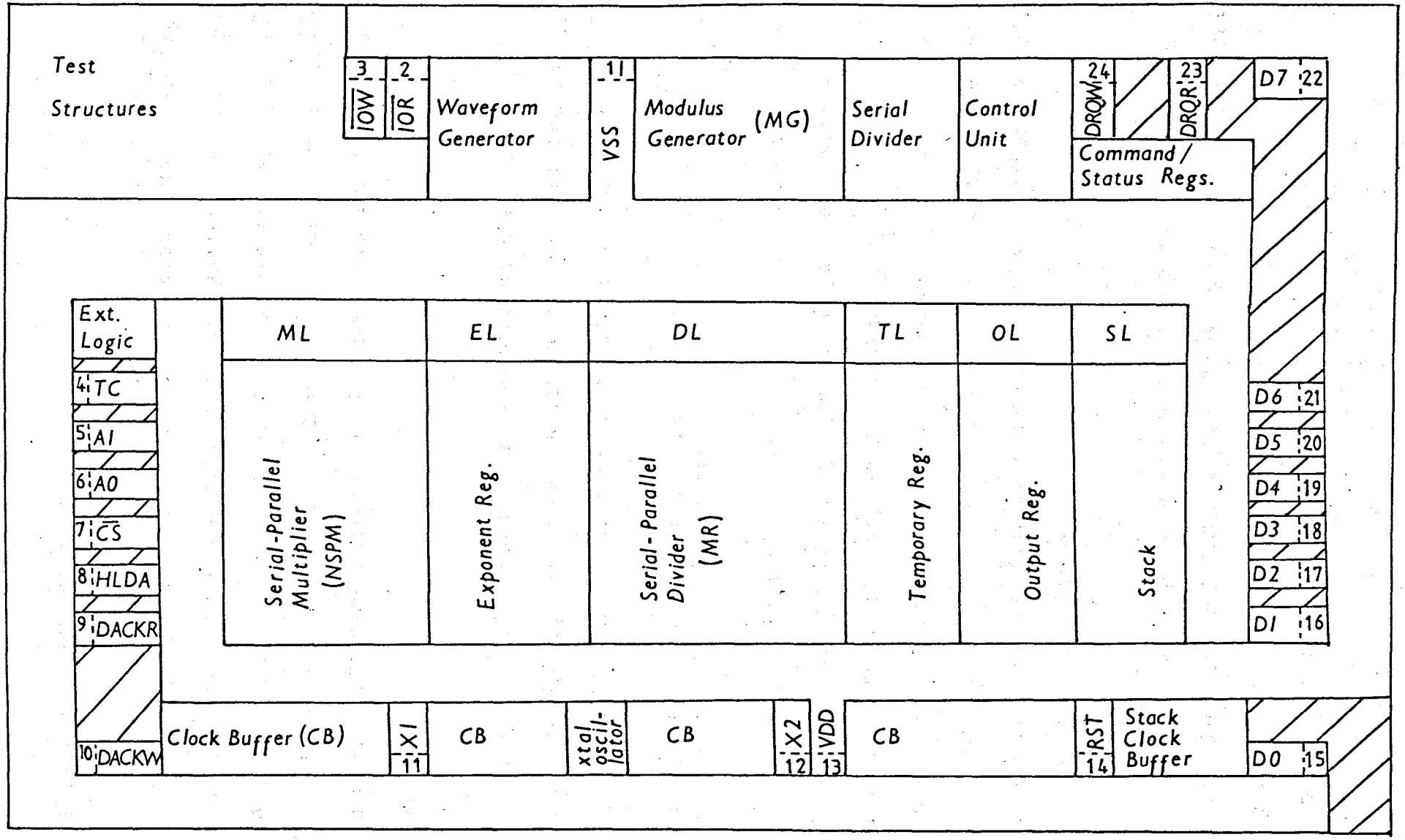


FIGURE 9.5 ACP FLOORPLAN.

9.4 ARCHITECTURE DESIGN.

At the time the design was in progress a good simulator, such as HILO (see section 6.2.1), was not available. This meant considerable more care had to be taken in the design. Having since designed a chip with HILO, I would estimate that several months would have been saved if it had been available for the ACP design.

Figure 9.5 shows the floorplan of the ACP. The blocks around the periphery with numbers attached represent the 24 pads (see appendix B). The other blocks are described below.

9.4.1 Clock generator and buffer.

When designing a two-phase clock generator (xtal. oscillator) and buffer (CB), the most important parameter is delay. It is normal to worry about delay between the system clock and the clock after it has been phase split and buffered. However, because we generate our own internal clock, only clock skew is a problem. The four outputs of the clock buffer, P, P´, Q and Q´ are shown in figure 9.6.

Various multiples of the basic clock frequency are used. But the fundamental multiple is 524 clock cycles. The name period will be used for this multiple. That is:
524 * (one clock cycle) = one period.

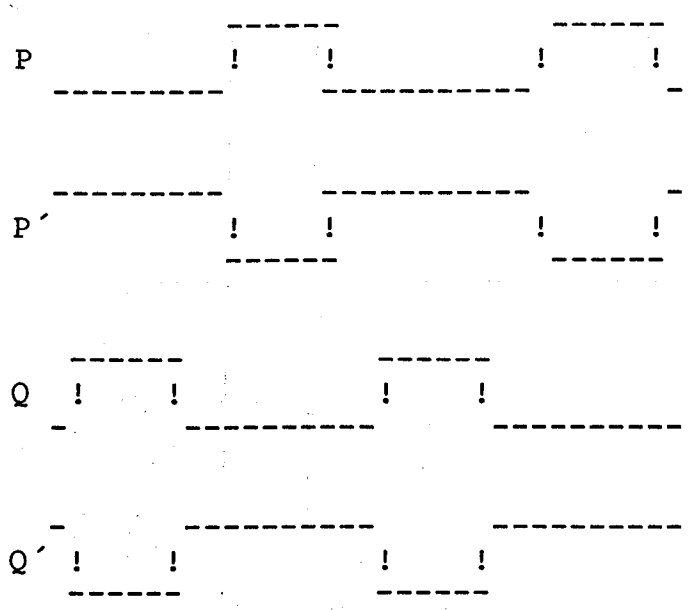


FIGURE 9.6 NON OVERLAPPING CLOCK WAVEFORMS.

9.4.2 Modular Exponential Unit.

To perform the binary modular exponentiation algorithm of section 9.3.1, a multiplier and modular reducer are needed. Firstly, to square M, and, secondly, to conditionally multiply C by this result. A GVA for doing this is shown in figures 9.7 and 9.8. Its main components are the 524-bit serial-parallel multiplier (the NSPM of section 8.4.2) and the 524-bit modular reducer (the MR of section 8.7). As well as this there is a 522-bit register (Temporary reg.), a multiplexer and two extra delay registers.

In addition to the components described above, there is a modulus generator (MG). This is responsible for generating the R (see section 8.7) required by the MR.

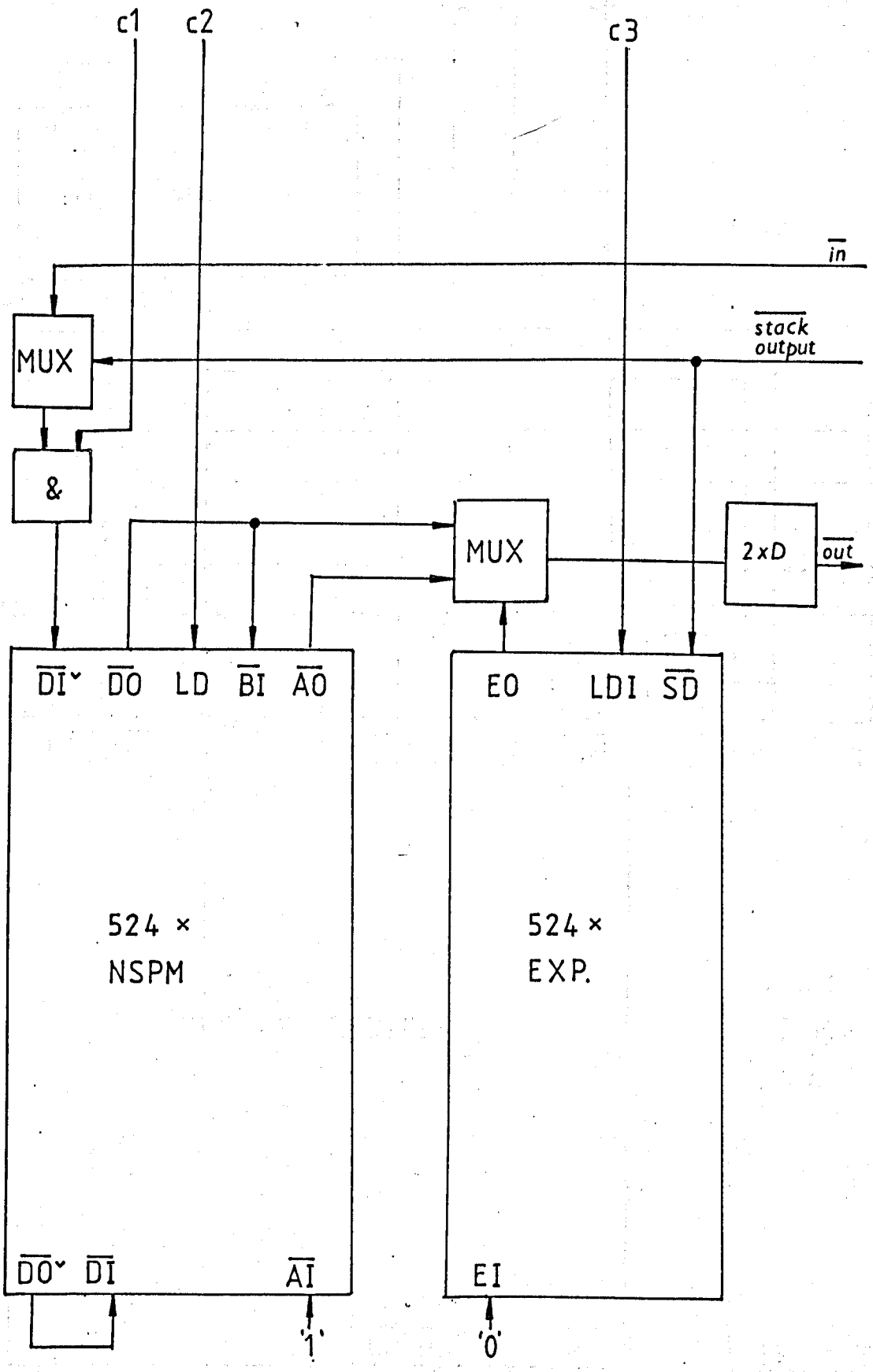


FIGURE 9.7 LEFT HAND HALF OF THE MEU.

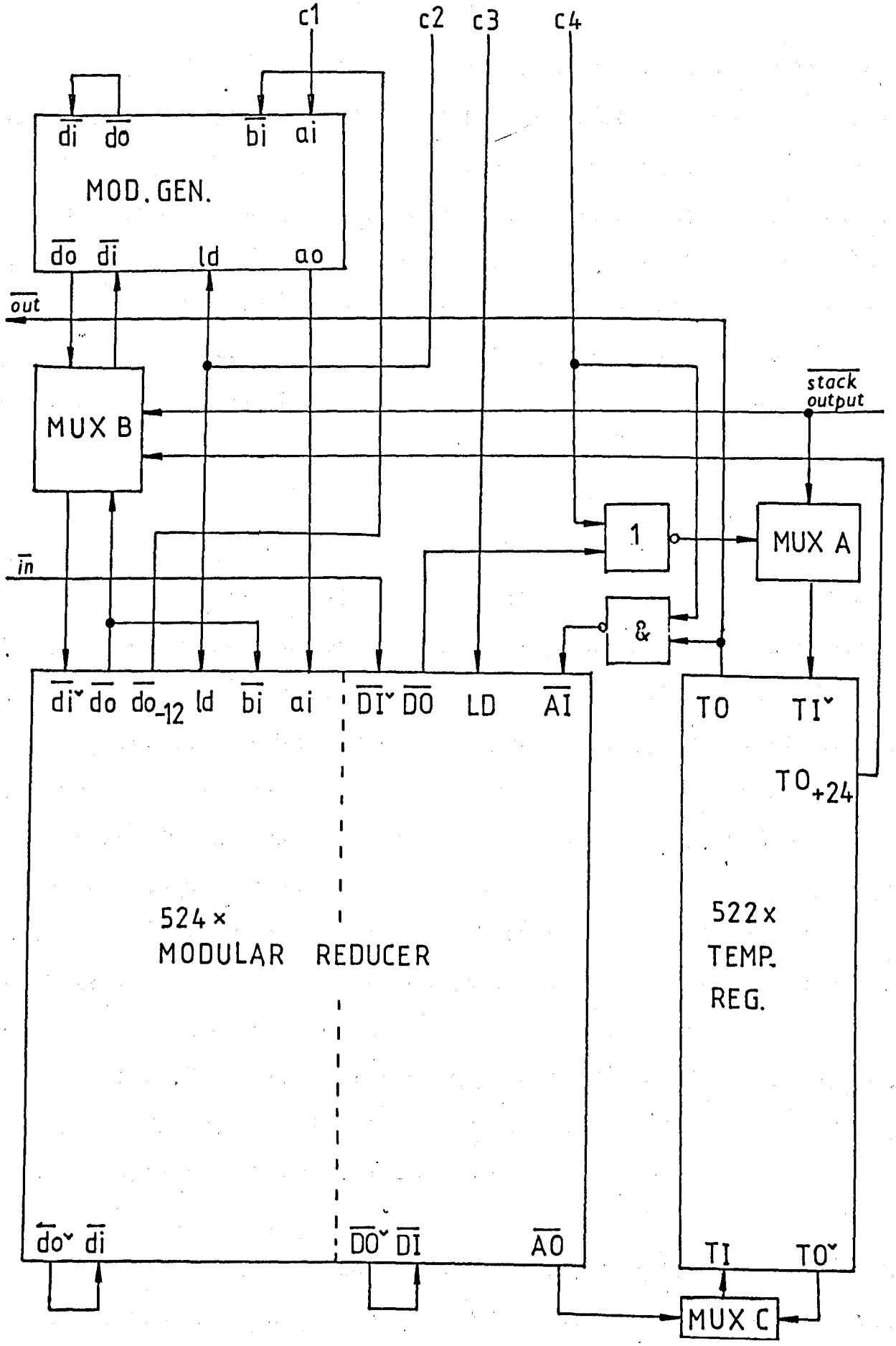


FIGURE 9.8 RIGHT HAND HALF OF THE MEU.

Before data can be enciphered, the exponent, modulus and constant register must be loaded. The first step is to load the command register (CR) with 00010000. Following this, the first 512 bits of the constant register are loaded into the stack. After loading, the data is directed from the stack, in serial format, to the modular reducer (di): via mux B. After 524 clock cycles (1 period), the output of the modular reducer (do) is directed back to its input (di). Thus, the first 512 bits of the constant will circulate round the d-register of the modular reducer.

To load the final 24 bits, bit 4 of the command register is forced high (CR=00001000). The four bytes of data are then transferred to the stack. After receiving the signal 'last' (see section 9.4.3), the data from the stack is transferred, via mux A into the temporary register (TI). For the next 476 clock cycles the 'rubbish' at the bottom of the stack is pushed out. The subsequent 24 cycles the real data is directed through TI, via TO+24 and mux B, to the modulus generator (MG) input (di). In the last 12 cycles of the period (the 512'th-524'th clock cycle), the output (do) of MG, goes through mux B into the modular reducer (di). For the first clock cycle of the next period, the load line (ld=c2) is held high. This loads all 536 bits of the constant u (see section 8.7) into the MR and MG.

Bit three of the command register is now held high (CR=00000100), and the 512 bit modulus is transferred to the stack. The stack contents are then conveyed to the modular

reducer, in the same way as the first 512 bits of the modulus. This will leave the d register with the modulus circulating once every period.

The exponent is transferred to the ACP by holding bit two of the command register high (CR=00000010), and loading the stack with the 64 bytes of data. It is then moved to the exponent register as described in section 9.4.8.

After the exponent, modulus and constant registers have been loaded, the ACP is ready to exponentiate. To tell the ACP that subsequent blocks of data are for exponentiating, bit one of the command register is held high (CR=00000001). When a 512-bit message (M) is loaded, it is padded with twelve zeros and gated into the multiplicand register (DI) of the NSPM. After 262 clock cycles (1/2 period) the load signal goes high for one clock cycle: loading the multiplicand with M. At the same time the message, emerging of the multiplicand register, is put into the multiplier register (BI). The resultant product (AO) is the message squared (M * M).

After one period, a 1 (initial state of C) padded with 523 zeros, is loaded serially into the multiplicand register. At the beginning of the second period, the last bit of the previous product (M * M) will be shifted out. The output of the multiplicand register (C) is again loaded into the multiplier register. However, this time the load line is kept inactive. The new product is the previous multiplicand (M) times the new

multiplier (C). This new product (C * M) together with the multiplier (C) is passed to a multiplexer: whose output is the product if e(0) is 1, otherwise it is the multiplier.

At the same time that this second multiplication is going on (second and third period), the previous product (M * M) is being reduced. Assume the previous product was a 1046-bit number (actually only 1024 bits are generated the first time). The output; having passed unhindered through the multiplexer, via the two delay registers, through the D register of the MR, into the temporary register; arrives at the start of the second period at the MR (AI).

During the first clock cycle of the second period the MR load line (LD = c3) is held high. At this time the 522 lsbs (L) are in the temporary register, while the remaining 524 most significant bits (msbs) are latched into the MR. The contents of the temporary register (L) are directed into the MR (AI). The resulting output (AO) will be a 523-bit number: the sum of 524 512-bit numbers and a 522-bit number (see section 8.7). The least significant bit of the result will come out of TO, via mux C, into the input of the NSPM, at the beginning of the third period.

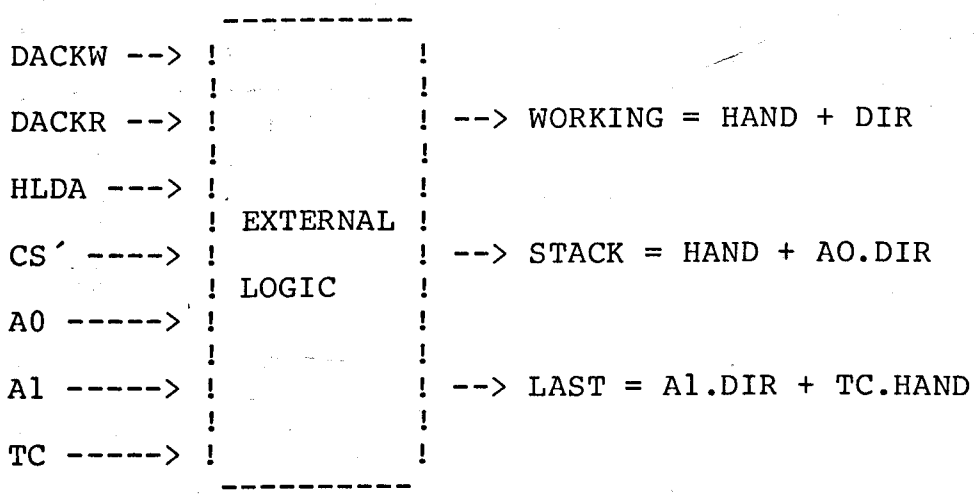
When the second product (C * M) has been shifted out, and the first product has been reduced (M * M mod X), the whole process repeats. At the start of the fifth period, the multiplier starts to square the result of the MR (M * M mod X); while the MR is now free to reduce the second product (C*M or C). The two

operations, squaring and multiplication, occur at the same time, while one is using the NSPM, the other is using the MR. There is no overflow problem, because the NSPM and MR have been carefully matched in size.

Data flows rhythmically, in a continuous stream. This means, most of the MEU latches need only store data for one clock cycle, permitting use of dynamic latches. The use of dynamic latches has a number of advantages. Most important they occupy one third less area (see section 9.5.2), which is very important for a design with around twenty percent of the total area being taken by latches. One slight disadvantage is the need for two-phase clocks.

9.4.3 External logic.

In order not to have any external 'glue' chips, a number of extra control pins were employed. This redundancy is removed in the external logic (Ext. Logic), where seven of the control pins are reduced to three lines. The meaning of these are:
LAST: The last byte of data is being transferred.
WORKING: An external device is transferring data.
STACK: If 1 then then external device is accessing the stack; else if 0, and working = 1, command or status register being written or read respectively.
Figure 9.9 shows their logic equations.



where DIR = (HLDA´.CS´) & HAND = HLDA.(DACKR +DACKW)

FIGURE 9.9 EXTERNAL LOGIC.

9.4.4 Control Unit.

As its name suggests, the control unit is responsible for switching the source and destination of data. The paths that are open are determined by the state of the control unit, and the part of the period it is currently in. Figure 9.10 shows the eight possible states (S0-S7) of the control unit. Changes in state are controlled by the output of the three registers shown in figure 9.11.

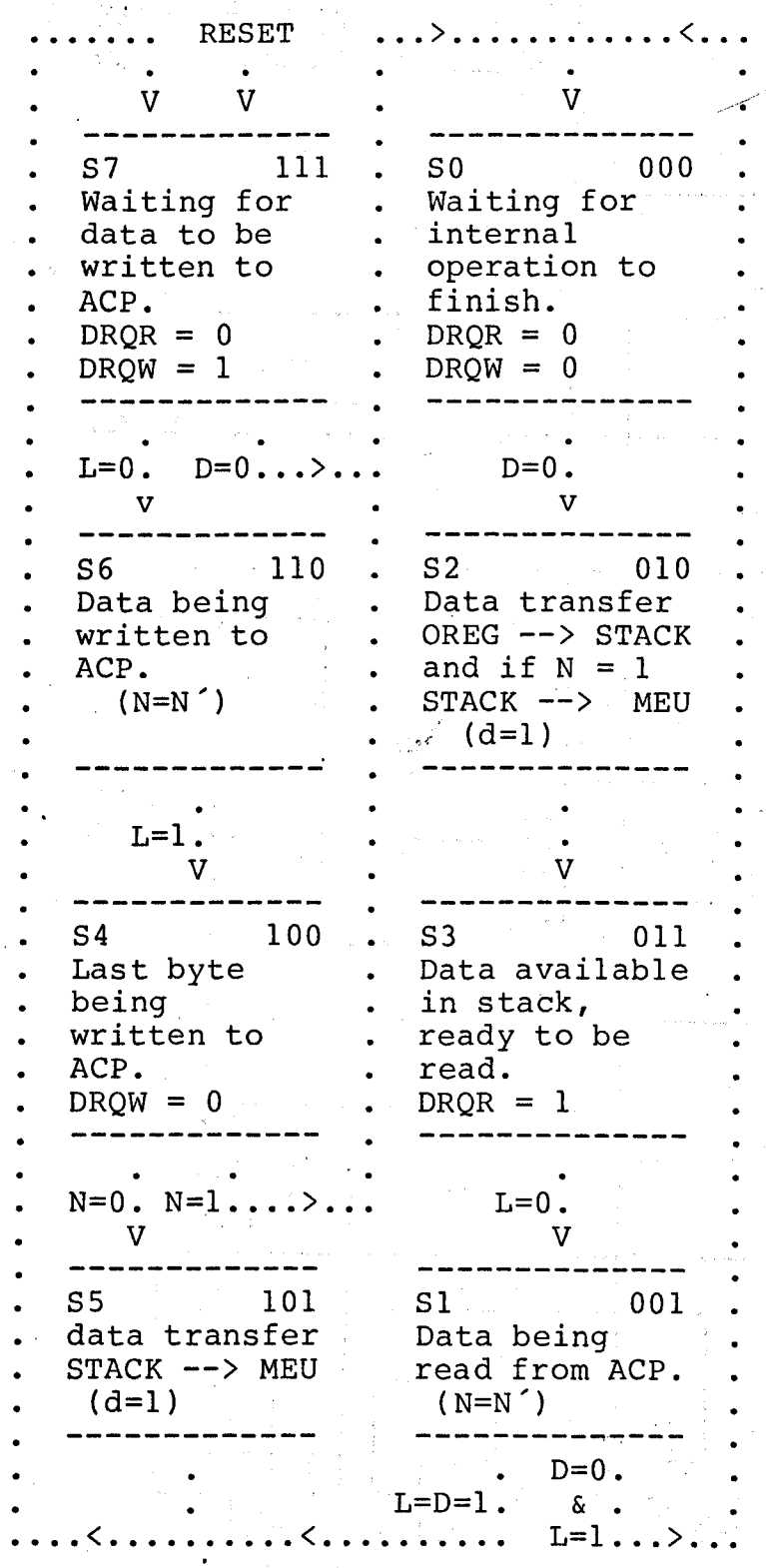


FIGURE 9.10 CONTROL UNIT STATE DIAGRAM.

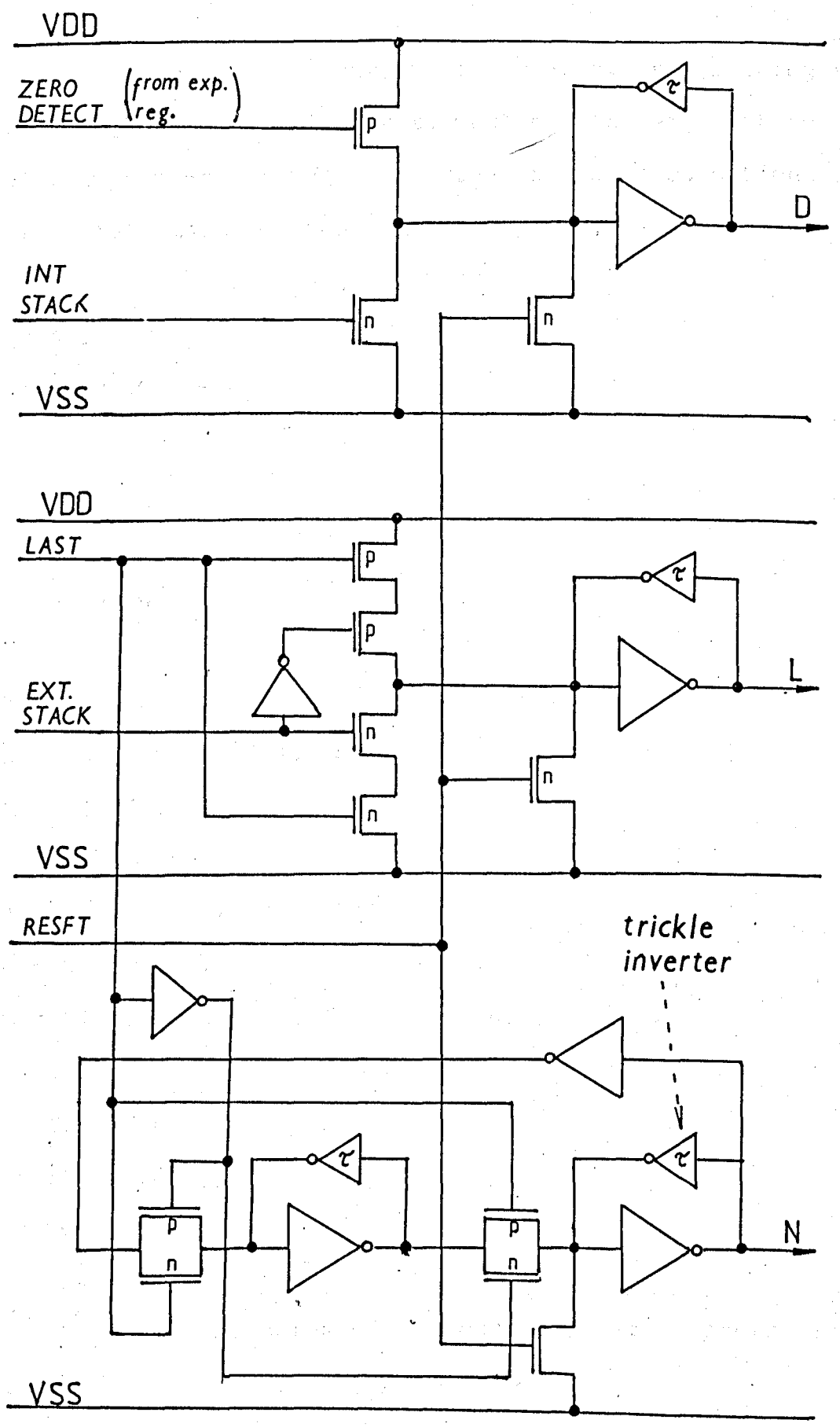


FIGURE 9.11 N,L AND D REGISTERS.

There are a number of ways of transferring from the state diagram to a finite state machine. The way I chose was to use the turn-on (TOx) and stay-on (SOx) conditions. For the finite state machine shown in figure 9.10, we can write:

$S7 = A \,.B \,.C$

$S6 = A.\ B \,.C'$

.

.

.

$S0 = A'.B'.C'$

$TOA = S1.L.D$

$TOB = S5 + S0.D' + S1.L.D$

$TOC = S4.N' + S2$

$SOA = S7.D + S6 + S4.N'.D + S5$

$SOB = S7.D + S6.L' + S2 + S1 + S3.L$

$SOC = S7.D.L + S5 + S3 + S1.(L' + D)$

$AO = TOA + SOA$

$BO = TOB + SOB$

$CO = TOC + SOC$

A block diagram of the control unit is shown in figure 9.12.

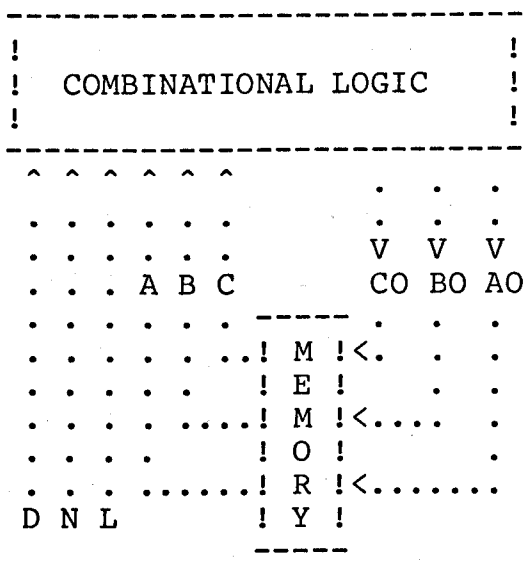


FIGURE 9.12 CONTROL UNIT ARCHITECTURE.

9.4.5 Stack.

The stack is organized as a 64 x 8 wide FIFO register. It is responsible for storing data when it is being loaded, and providing a place where processed data can be accessed. The stack logic (SL - see figure 9.5) converts the external byte format into the serial format required internally.

Figure 9.13 shows the architecture of the stack and SL. The stack clock can either be derived from external signals or from a 1/8th internal clock (F/8). When external data is being loaded, the external control logic ensures that the internal control is switched off. The stack clock is bufferd by the 'stack clock buffer' (see figure 9.5).

9.4.6 Waveform generator.

The main part of the waveform generator is a 524 state counter. A 511-bit sequence can be obtained from a 9-stage shift register, with exclusive-or feedback from stages 4 and 9. The 524 state design is based on this 511 state pseudo-random binary sequence (PRBS), with logic to increase the cycle length to the 13 extra states required.

The design is similar to that used for the control unit. A more detailed diagram is given in figure 9.20. For the waveform generator we have the equations below: the inputs A-J represent the outputs of our PRBS (with extra logic in the feedback path), and X the extra state required to extend our sequence.

```
FEEDBACK = D'.J + D.J';
S0       = A'.B'.C'.D'.E'.F'.G'.H'.J';
S505     = A'.B'.C .D'.E'.F'.G'.H .J';
S511     = A'.B'.C'.D'.E'.F'.G'.H'.J ;
S523     = (X.C.D);
L        = S523';

AO = (FEEDBACK.L) + S0;
BO = A;
CO = B;
DO = C.L;
EO = D.L;
```

```
FO = E;
GO = F;
HO = G;
JO = H.L;
XO = S511 + X.L;
S  = S0;
I  = S505;
```

The two extra outputs, S and I, are used in other parts of the microcode. The operation of the counter has been simulated in ISPS (see appendix C).

9.4.7 Serial divider and output register.

The serial divider reduces the final 523-bit result to one of 512 bits. The design is based on the non-restoring division algorithm (Ref.805). Figure 9.14 shows the architecture used.

The output from the MEU is directed to the 512-bit output register. After 512 clock cycles the lsb is loaded into the input of the serial divider: which has been initialized to subtract. The output will then equal:
outputl = input - (modulus * 2^11).

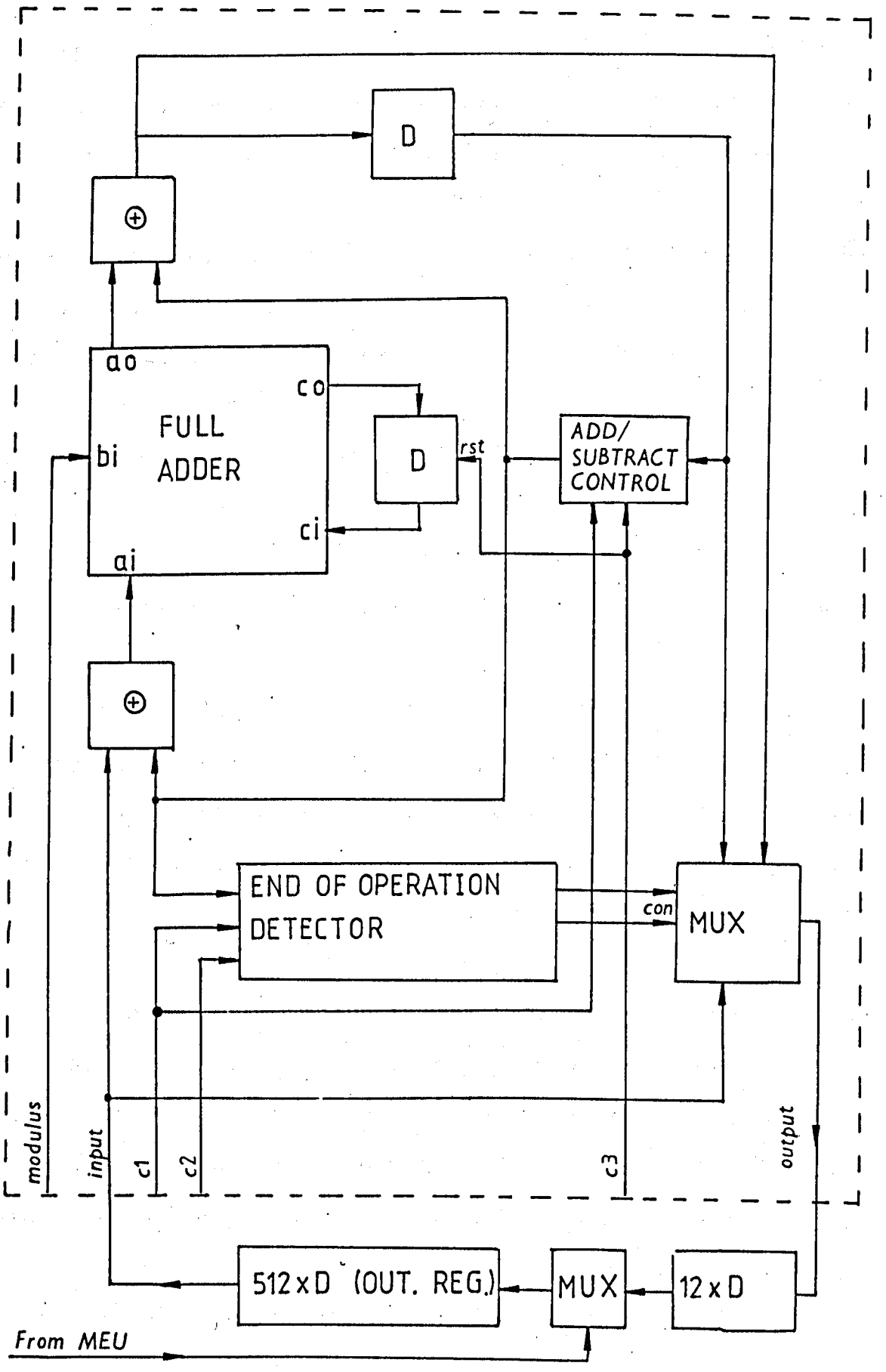


FIGURE 9.14 SERIAL DIVIDER.

If the output is negative, that is the most significant bit (msb) is a 1, then the operation during the next period will be an addition. If the output is positive another subtraction is performed. The lsb of the output will take 525 clock cycles (1 period + 1 cycle) to arrive back at the input: 1 delay in the serial divider, 12 in the extra register and 512 in the output register. The result of the next eleven operation will therefore be:

output2 = output1 +/- (modulus * 2^10)

output3 = output2 +/- (modulus * 2^9)

...

output12 = output11 +/- (modulus * 2^0)

On the twelth and subsequent periods the result is delayed by one less clock cycle, bypassing the internal delay. Output 12 is a 512 bit number, which could be negative. If it is negative a final addition is performed: output 13 = output12 + modulus. Thereafter the input is directed straight to the output.

The three microcode lines c1-c3 control the serial division operation. C1 and c2 tell the 'End of oeration detector' when the first and twelth cycle are in progress. It, in turn, switches the output data from one of its three inputs. At the beginning of each addition/subtraction, the c3 signal resets the carry register and tells the 'add/subtract control' when to sample the data (i.e most significant bit).

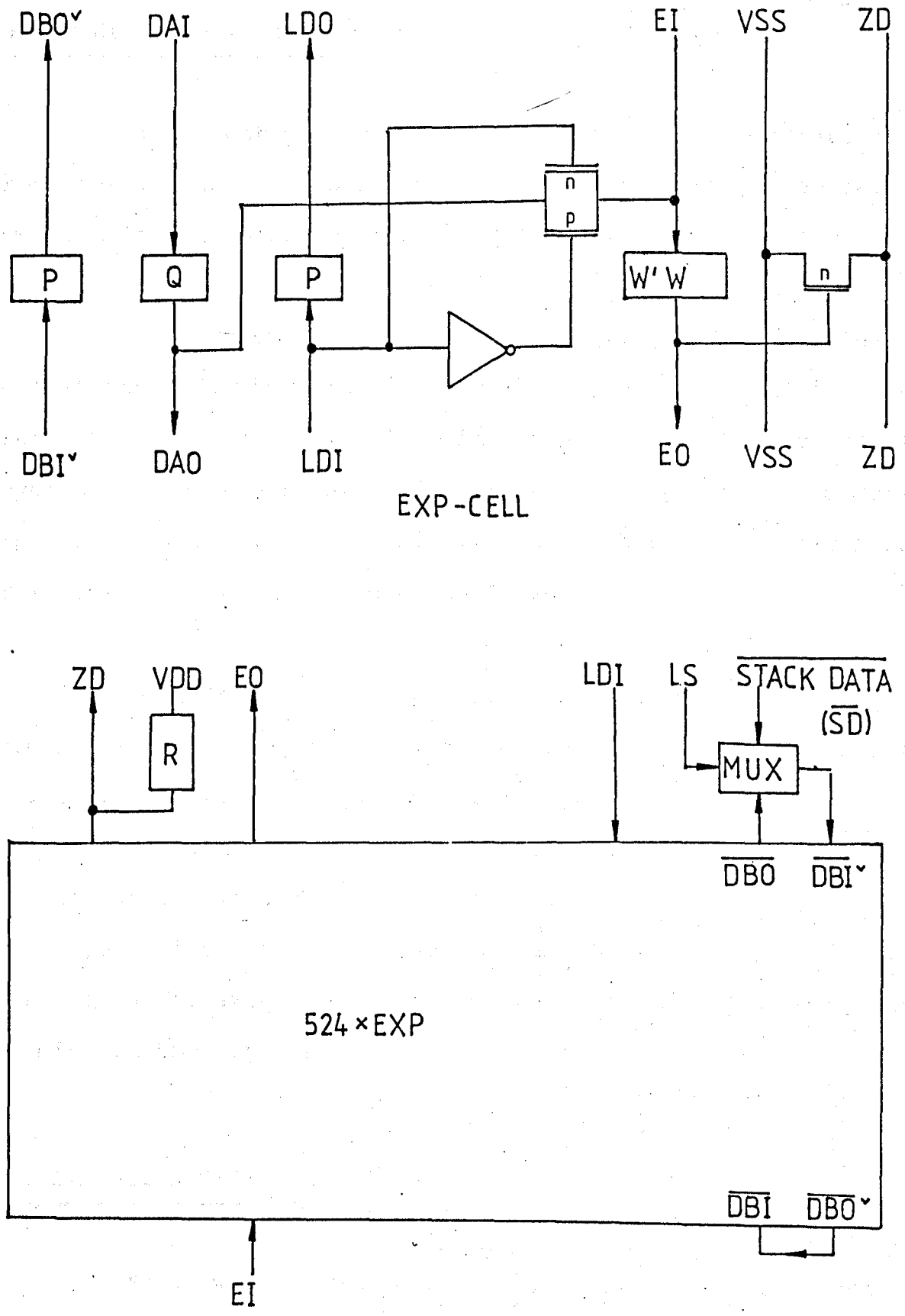


FIGURE 9.15 EXPONENT REGISTER.

9.4.8 Other registers.

The control and status registers are tristate static registers, connected to the data pads. Whether they are read or written is determined by the control unit.

The exponent register is a little more complex: its architecture is shown in figure 9.15. The stack data is loaded into the D register, where it circulates once every 524 clock cycles. During the first period of the MEU operation, described in section 9.4.2, the LDI signal goes high for one clock period. This loads the exponent into the E register.

Once loaded, the E register is clocked out serially with clock W (which changes state at the start of each period). The E register contains a zero detection circuit, so that when all the significant bits of the exponent have been shifted out, the ZD signal goes high. The ZD signal is sampled and if active passes the result of the MR to the output register (see section 9.4.7). The speed of execution is therefore proportional to the number of bits in the exponent.

9.5 CIRCUIT DESIGN.

There were a number of design styles used in the ACP leaf cells. For the main processing element, registers, crystal oscillator and pads, full custom design was used (see section

6.5). For the interface, serial divider and control unit a combination of PLA's and standard cells were used. The main building blocks are described below.

9.5.1 Gated adder and subtractor.

The circuit for the adder has already been described in the design example of chapter 6. This adder is employed in the NSPM, MR and serial divider. By inverting the A input we get the subtractor needed for the MR and serial divider.

9.5.2 Static and dynamic latches.

Two types of latches were used in the ACP: static latches, which maintain their state by feedback; and dynamic latches, which use the temporary capacitive charge on gate inputs (Ref.905). The static latches (see figure 9.16a) are used wherever data is not refreshed every clock cycle. The dynamic latches (see figure 9.16b) are used only when data is refreshed every clock cycle, but account for approximately 75% of all latches on the ACP.

9.5.3 Crystal oscillator.

There is now widespread use of quartz crystals in high performance oscillator circuits. The left hand half of figure 9.17 shows the ciruit diagram of the Pierce oscillator used in the ACP. The simplicity of the circuit belies the complexity of analysis (Ref.906).

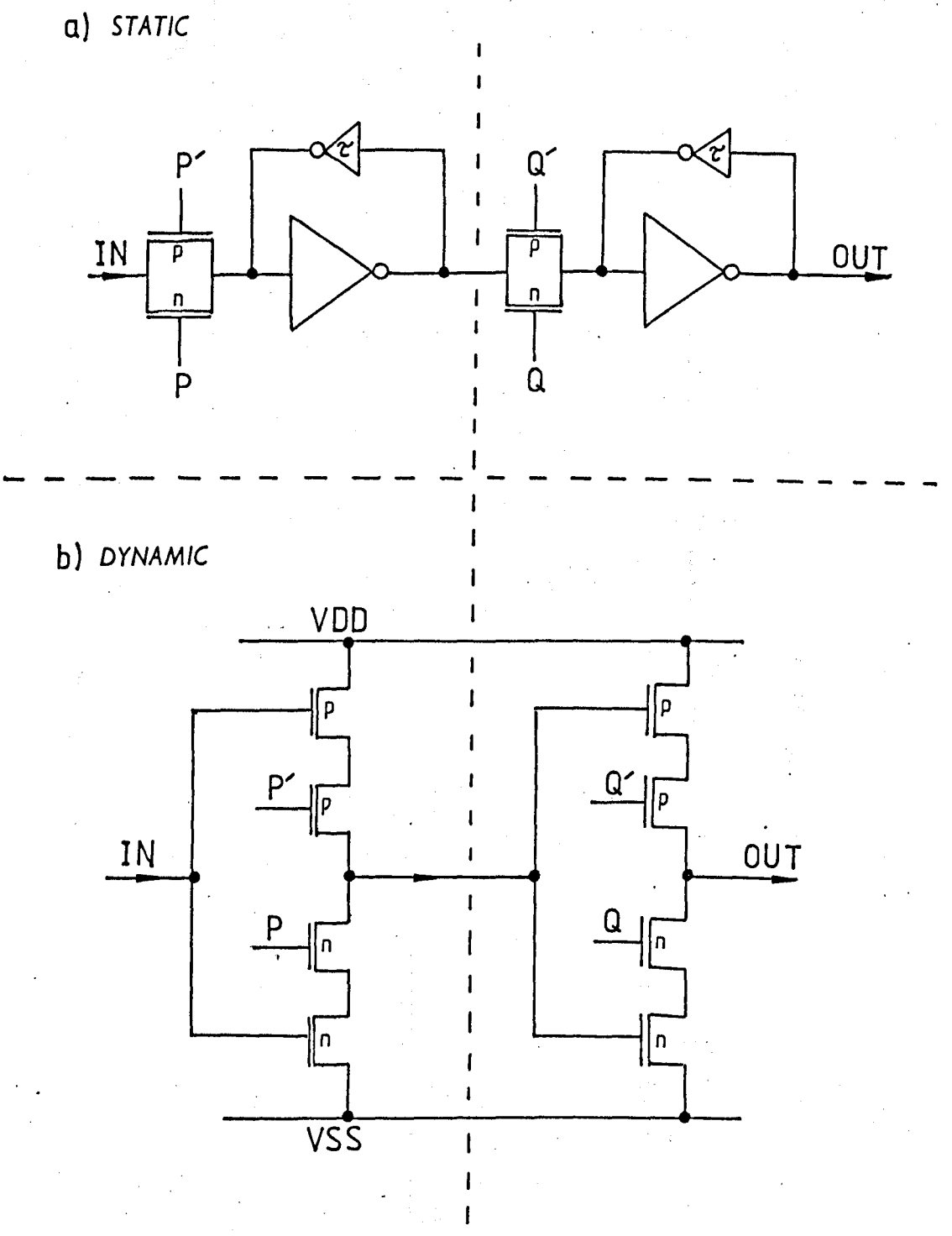


FIGURE 9.16 STATIC AND DYNAMIC LATCHES.

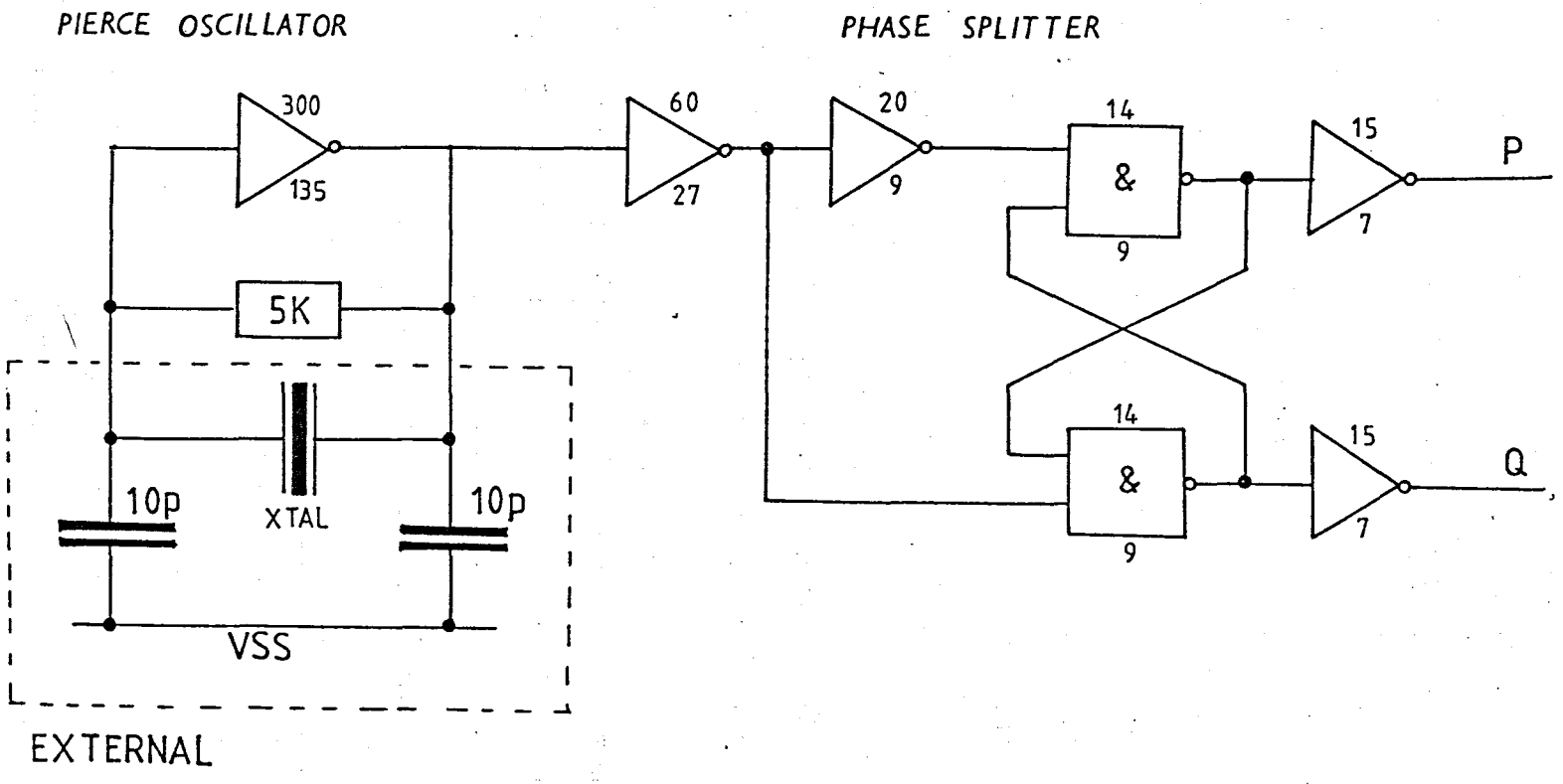


FIGURE 9.17 PIERCE OSCILLATOR AND PHASE SPLITTER.

The crystal oscillator was extensively simulated with SPICE (see section 6.5). These simulations show the circuit has high frequency and temperature stability, mainly due to the high Q of the crystal.

9.5.4 Phase splitter and buffer.

The non-overlapping two phase clock is generated using cross coupled NAND gates. The circuit used is shown in the right hand half of figure 9.18. After being buffered, the output of this circuit feeds the nine clock buffer circuits.

Each of the clock buffers employs a cascade arrangement, with each inverter driving successively larger inverters (Ref 905). Although a ratio of 2.7:1 is optimal for performance, slightly larger ratios usually give denser layouts without significantly affecting performance. Therefore, a ratio of 4-5:1 was used both here and in the output buffers. The distributed clock buffers have a combined drive capability equivalent to a single inverter, with 2200 (n) and 5000 (p) micron transistors.

9.5.5 Pads.

Three types of pad are required by the ACP: input, output and bidirectional. The output pad uses four cascaded inverters, with the final transistor of 120 (n) and 270 (p) microns. Simulations showed the output pad can drive a 25pF load in 10ns.

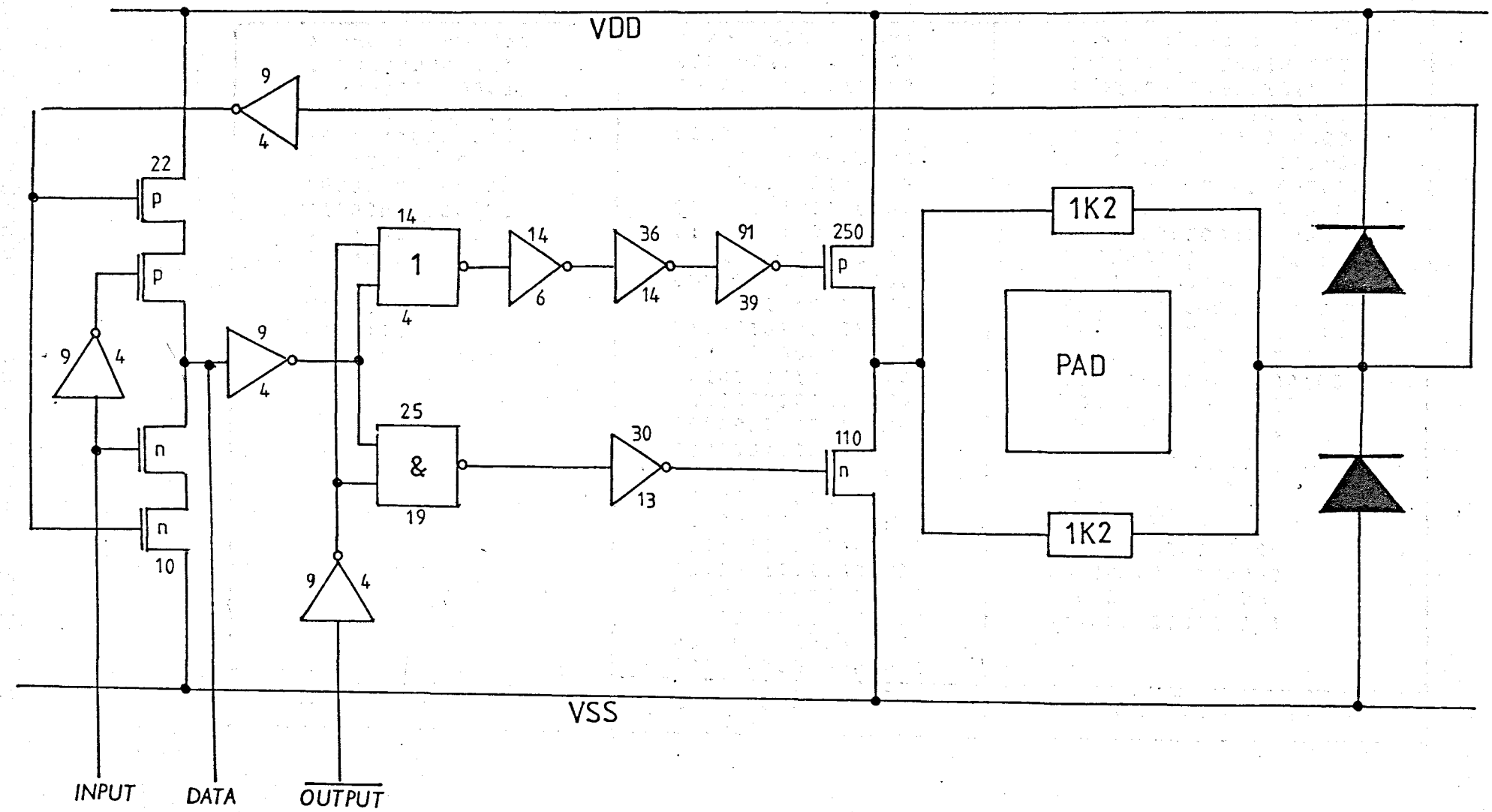


FIGURE 9.18 BIDIRECTIONAL PAD CIRCUIT.

EXT. LOGIC	CONTROL UNIT	WAVEFORM GEN.
* and plane data C1000C000000101 01000000100001 10010010000000 10000110000000 10010000000000 10000100000000 C1000000000001 * or plane data 0000000 1000000 0111000 0000111	* and plane data 00000010001000000 00100000000010000 00100001100110000 00100010001010000 00010010001000000 00010101001010000 00100001000100000 01010001000101100 00010010011000000 00010010000100000 00100010100101000 00000010000100000 00100010101010000 00100001010100000 00100001000110000 00011010000100000 * or plane data 1000000000000000 C1000000000000000 0011111100000000 0011001111100000 C0000010000111110 0000000000000000 0000001001000000 0000001000000001	* and plane data C1010101010001010101 1001100101000101010 1 0101010101001001010 1 0000010000100000000 0 0000000000100000010 0 1000000000010000000 0 1000010000000000000 0 1000000000000000010 0 0000000000010000100 0 0000010000000000100 0 0000100000010000000 0 0000100000000000010 0 0000000000010100100 0 0000010000000100100 0 * or plane data 000000000000000 100000000000000 010000000000000 001110000000000 000001110000000 000000001100000 000000000011000 100000000000111

FIGURE 9.19 PLA AND-OR PLANE CONNECTIONS.

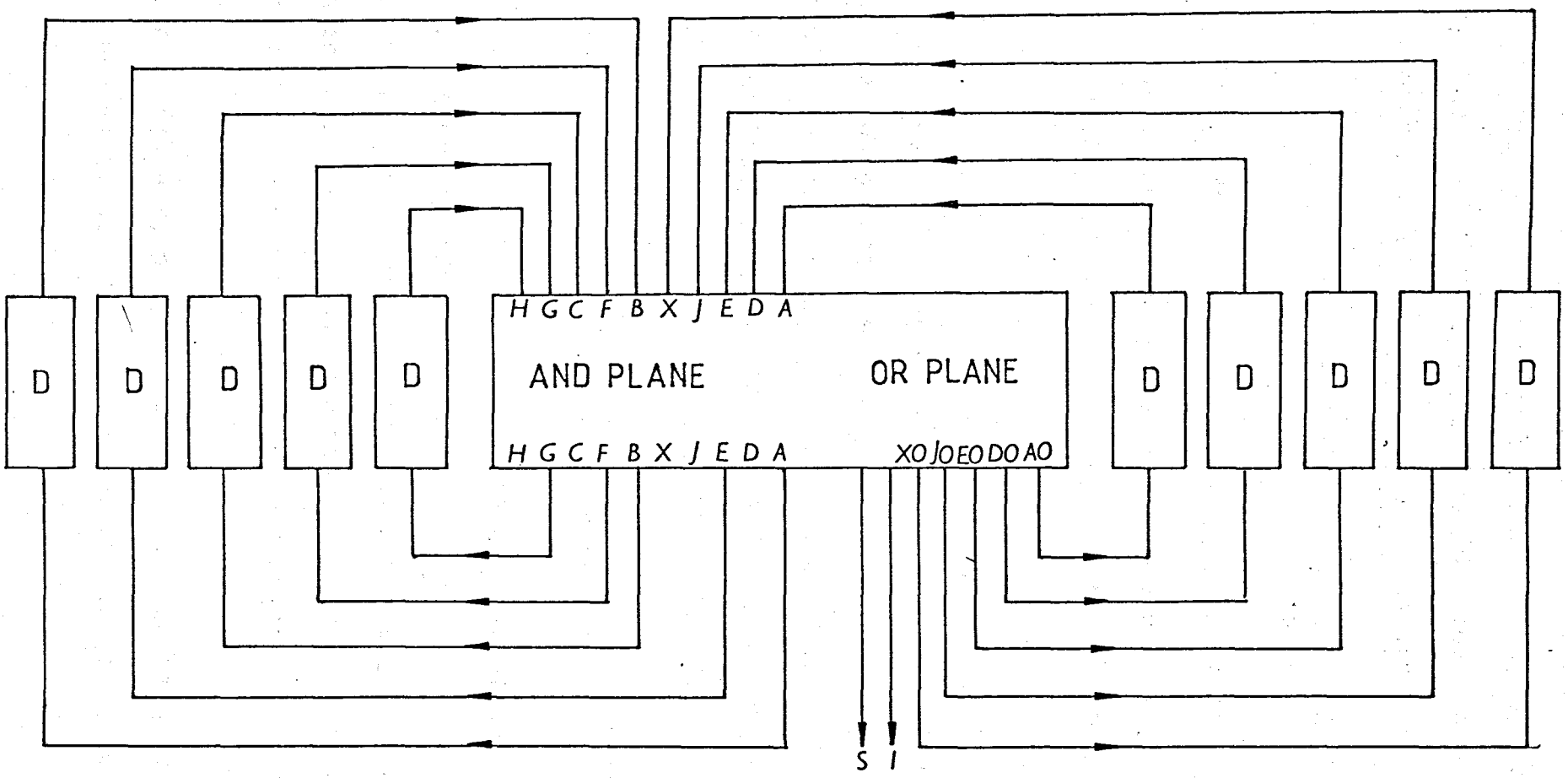


FIGURE 9.20 WAVEFORM GENERATOR.

The input pad has diodes to short out voltage spikes and a resistor to dissipate the power. The circuit is the same as the right hand half of the bidirectional pad (see figure 9.18).

The bidirectional pad is basically the input and output pads combined. However, the need to tristate the output demands slightly more complex circuitry. Figure 9.18 shows the final circuit and transistor sizes. Simulations showed this pad can drive a 25pF load in 12ns.

9.5.6 Programmable Logic Arrays.

The PLA generator allows fast turn around for both sequential and combinatorial circuits (Ref.905). The input to the PLA generator are the logical equations. Three PLA's were used in the ACP: for the external logic, waveform generator and part of the control unit. The AND and OR plane connections are shown in figure 9.19.

The circuit for the waveform generator is shown in figure 9.20. The orderings of the inputs and outputs are chosen to avoid wire crossovers.

9.5.7 Random Logic.

The random logic circuitry is used extensively in the gating and control sections at the top of the ACP (see figure 9.5). Particulary in the ML, EL, DL, TL, OL, SL, serial divider, control unit and crystal oscillator.

Using simple building blocks such as inverters, multiplexers, NAND and NOR gates; a traditional logic design approach was taken.

9.6 LAYOUT.

Figures 9.21 to 9.32 show some of the cells and macrocells used in the ACP. The final layout of the ACP is given in a pouch connected to the back page. The whole drawing has passed DRC and ERC (see section 6.6.2). Including scribe channel, the drawing measures 8.6 x 6.8 mm (3.4 x 2.7 microns).

The ACP contains some 140 000 transistors of which it was necessary to design:

a) 200 FULL CUSTOM (e.g. SPM and pads).

b) 100 PLA (e.g. control, external and waveform generation logic).

c) 800 SEMI-CUSTOM - CELL DESIGN (e.g. serial divider and data control logic).

Therefore, the regularity factor (see section 6.1) is approximately 100.

Layout of pads required some extra knowledge of processing. To ensure there was no static breakdown, they employ good substrate contacts, thick power rails and no overlap of the input line with gate polysilicon.

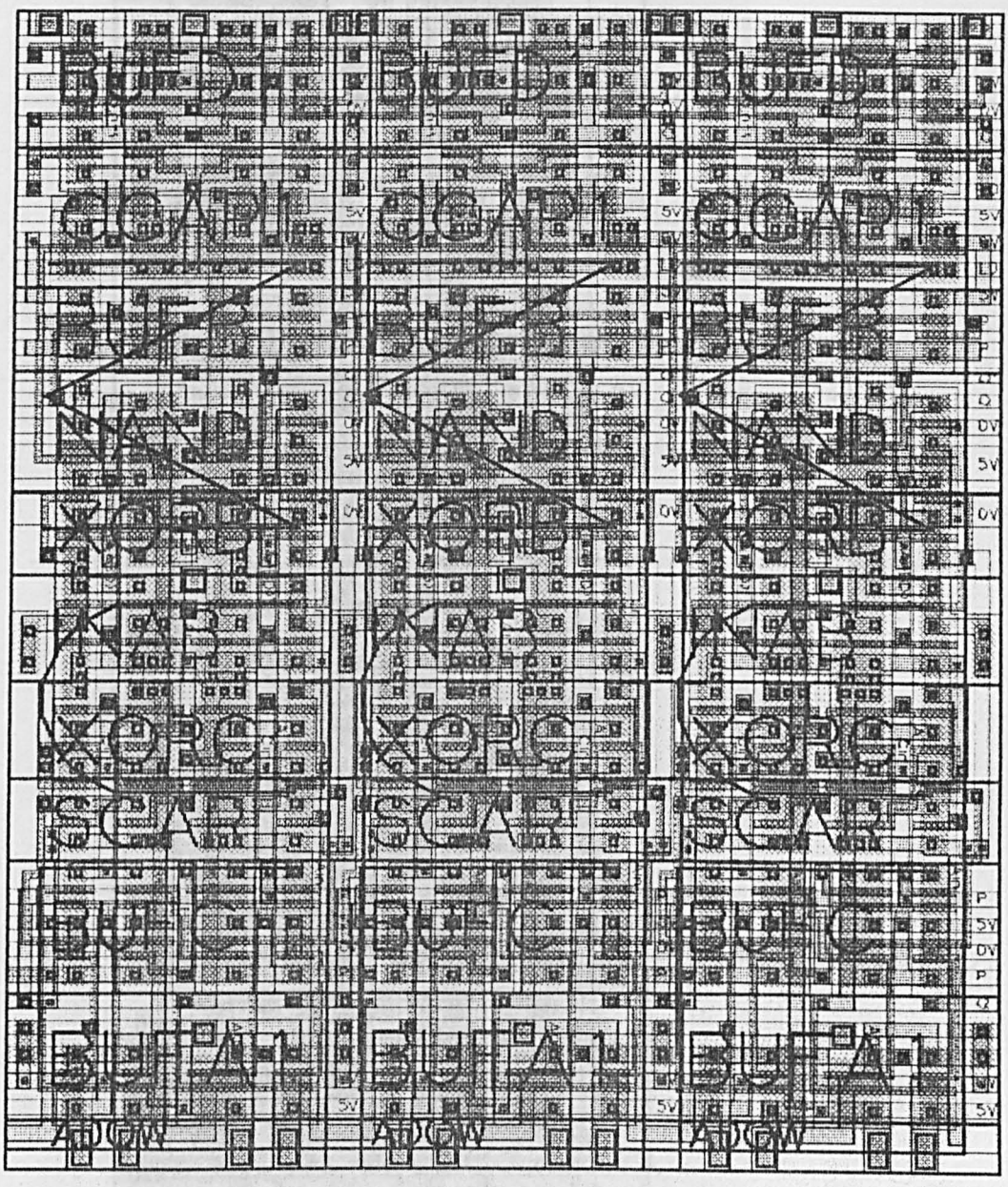


FIGURE 9.21 SIX STAGE SERIAL PARALLEL MULTIPLIER.

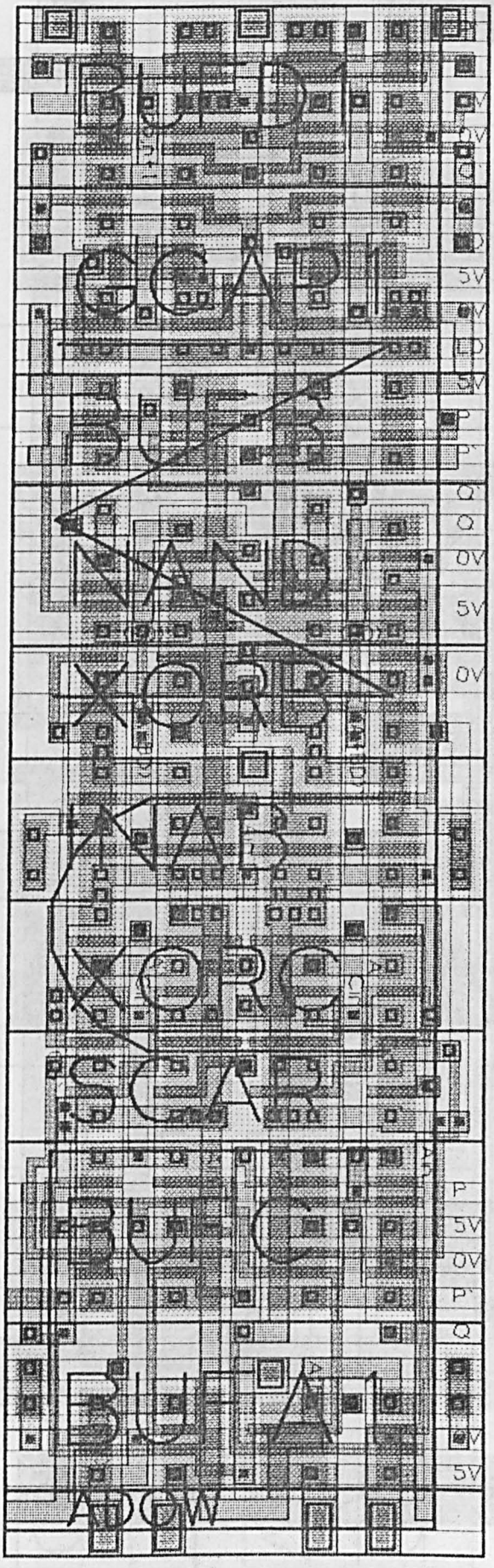


FIGURE 9.22 TWO STAGE SERIAL PARALLEL MULTIPLIER.

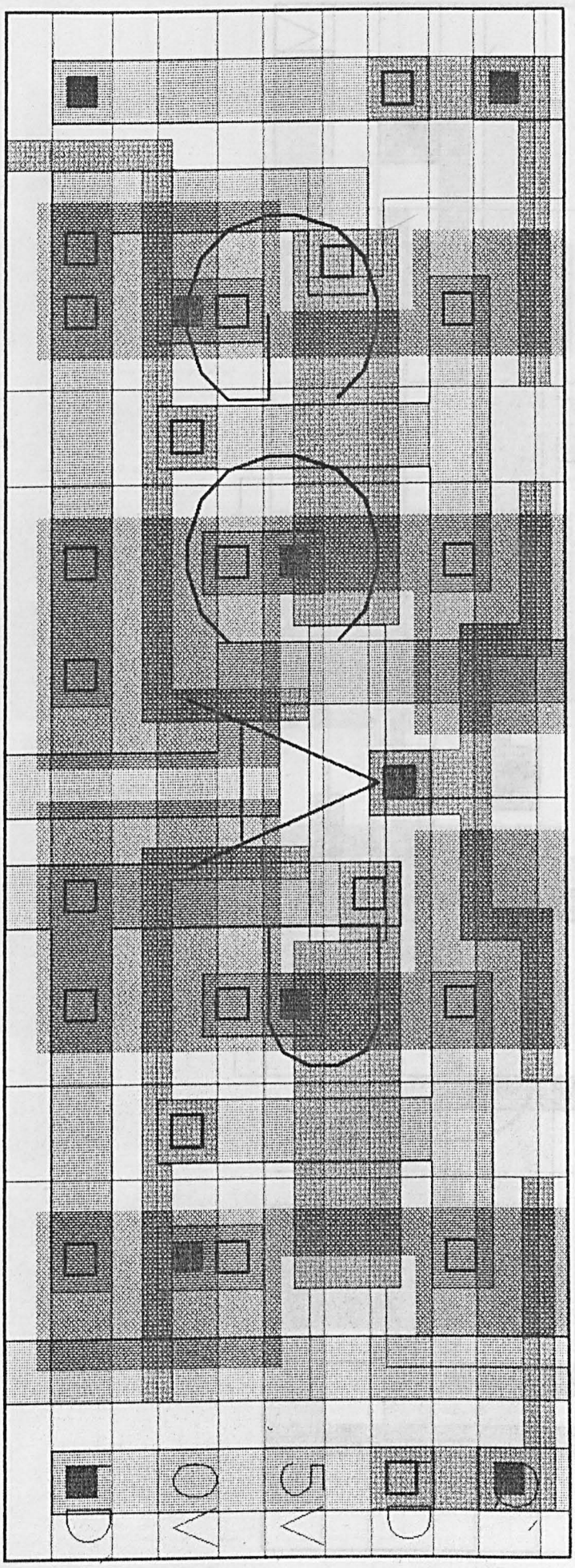


FIGURE 9.23 STATIC LATCHES.

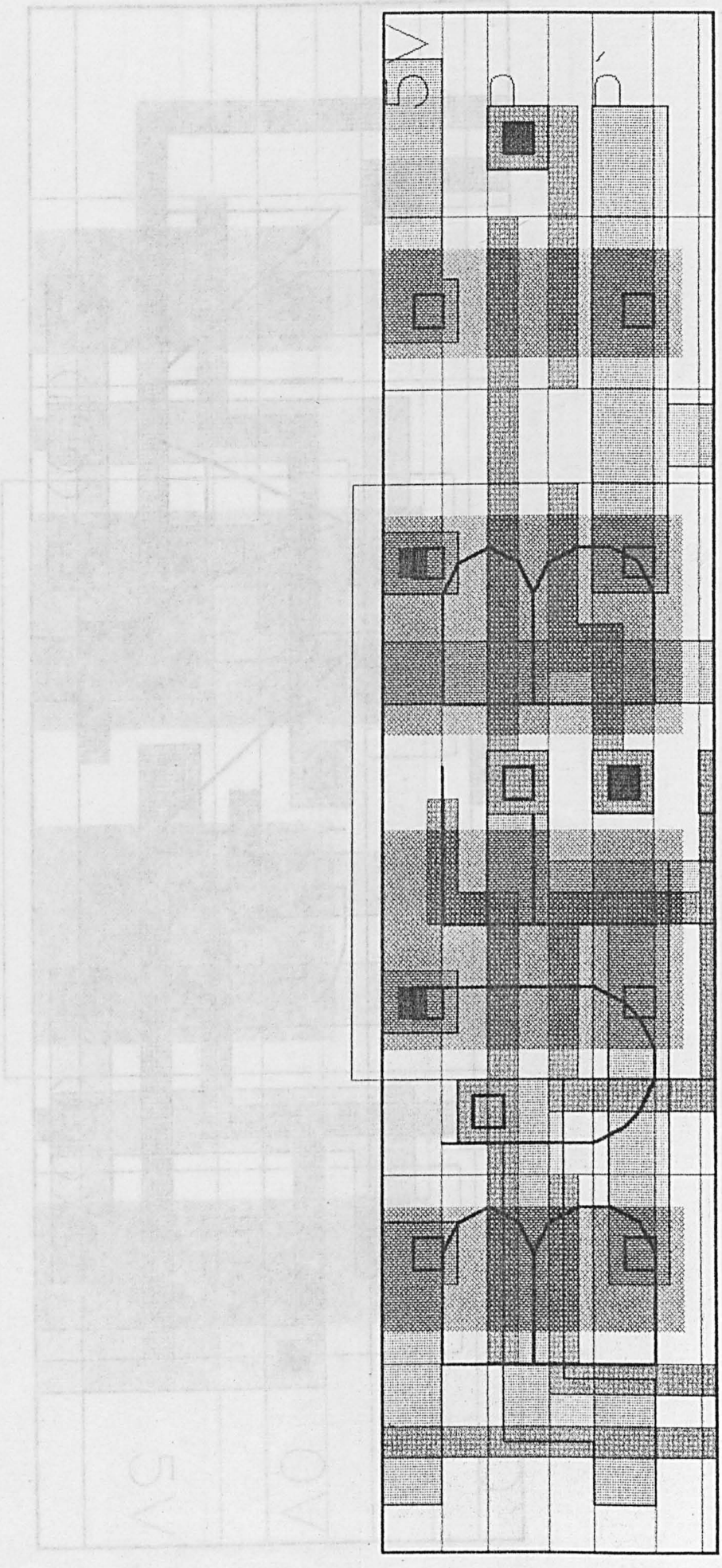


FIGURE 9.24 DYNAMIC TRANSPARENT LATCHES.

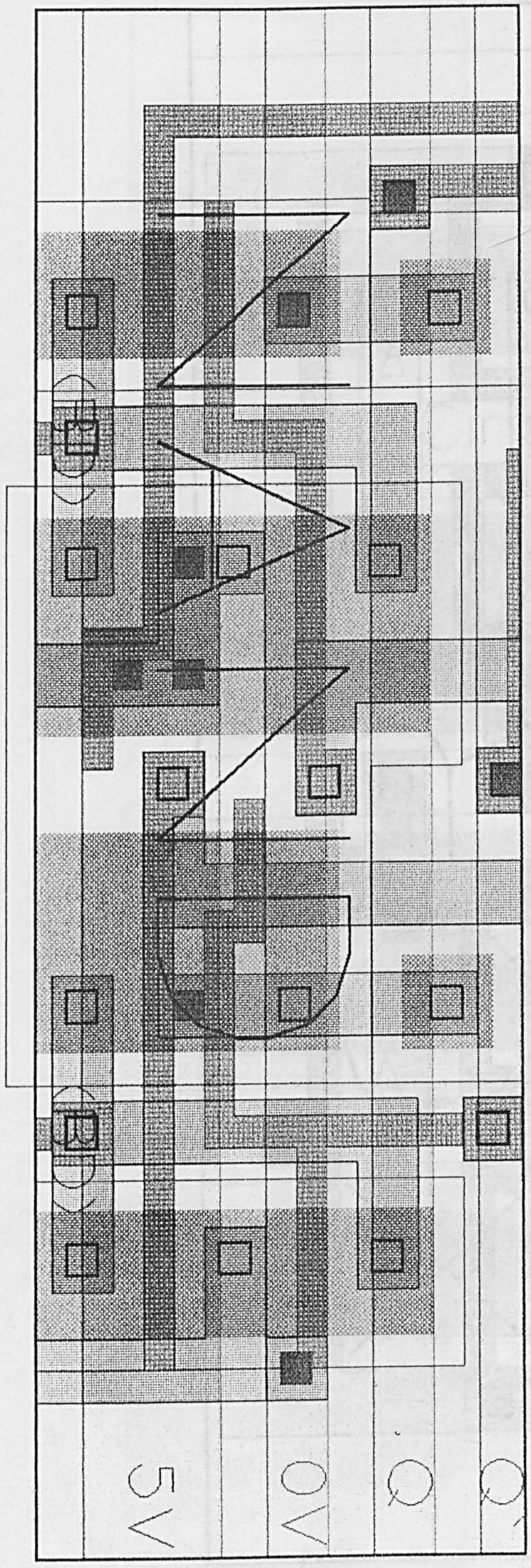


FIGURE 9.25 NAND GATES.

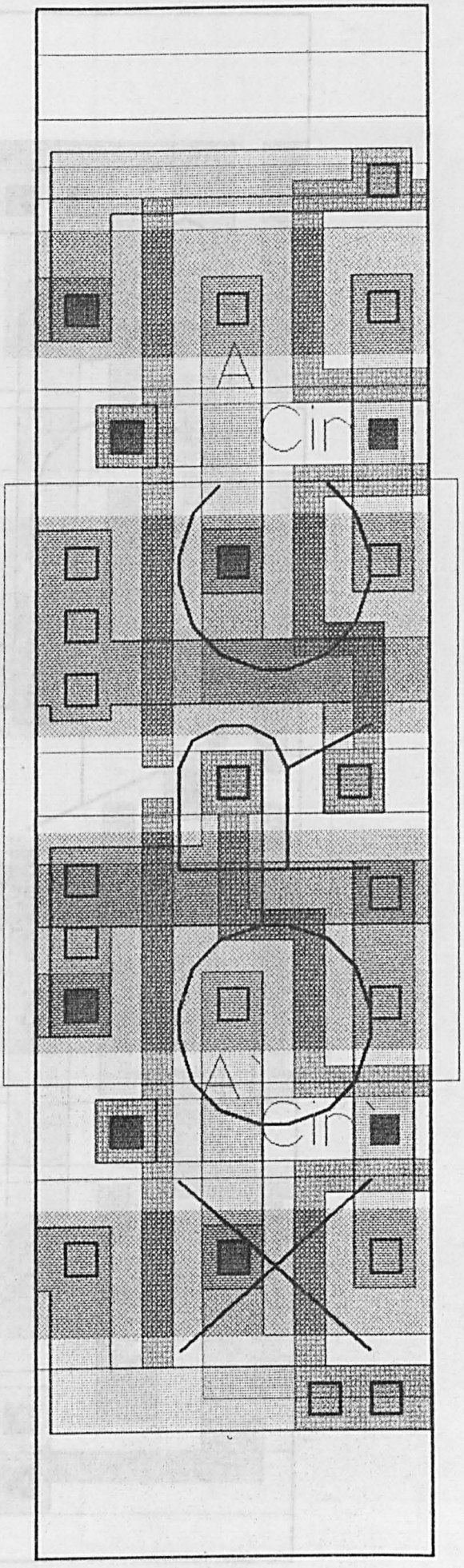


FIGURE 9.26 EXCLUSIVE-OR GATES.

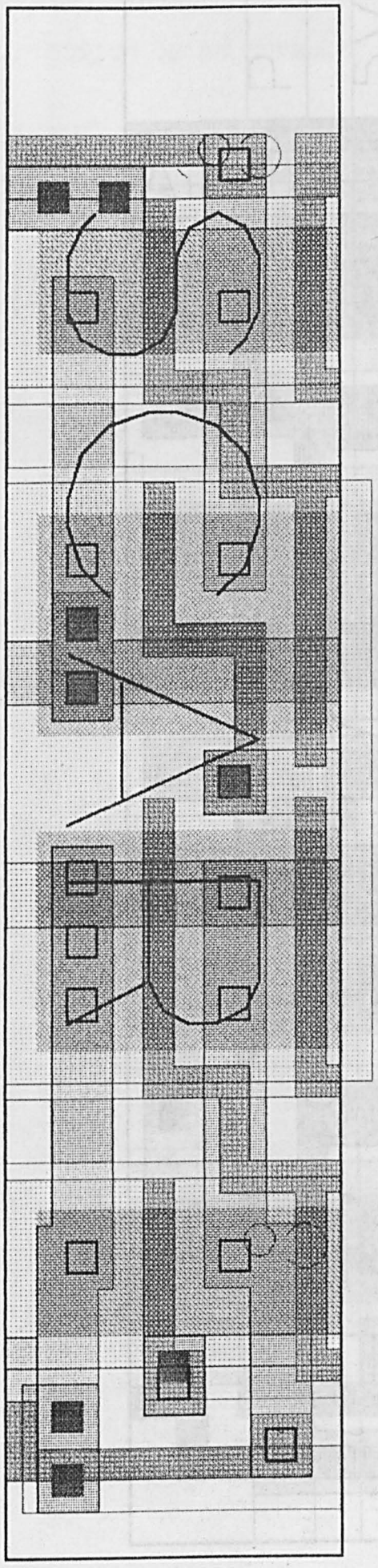


FIGURE 9.27 MULTIPLEXERS.

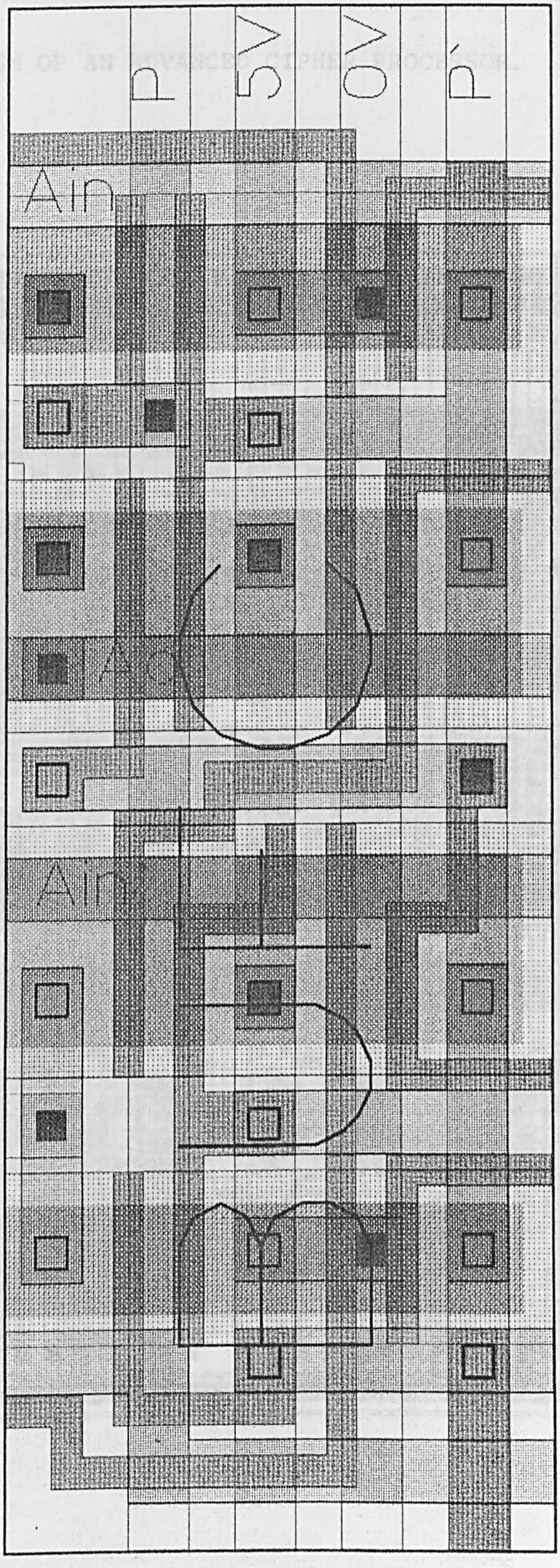


FIGURE 9.28 DYNAMIC LATCHES.

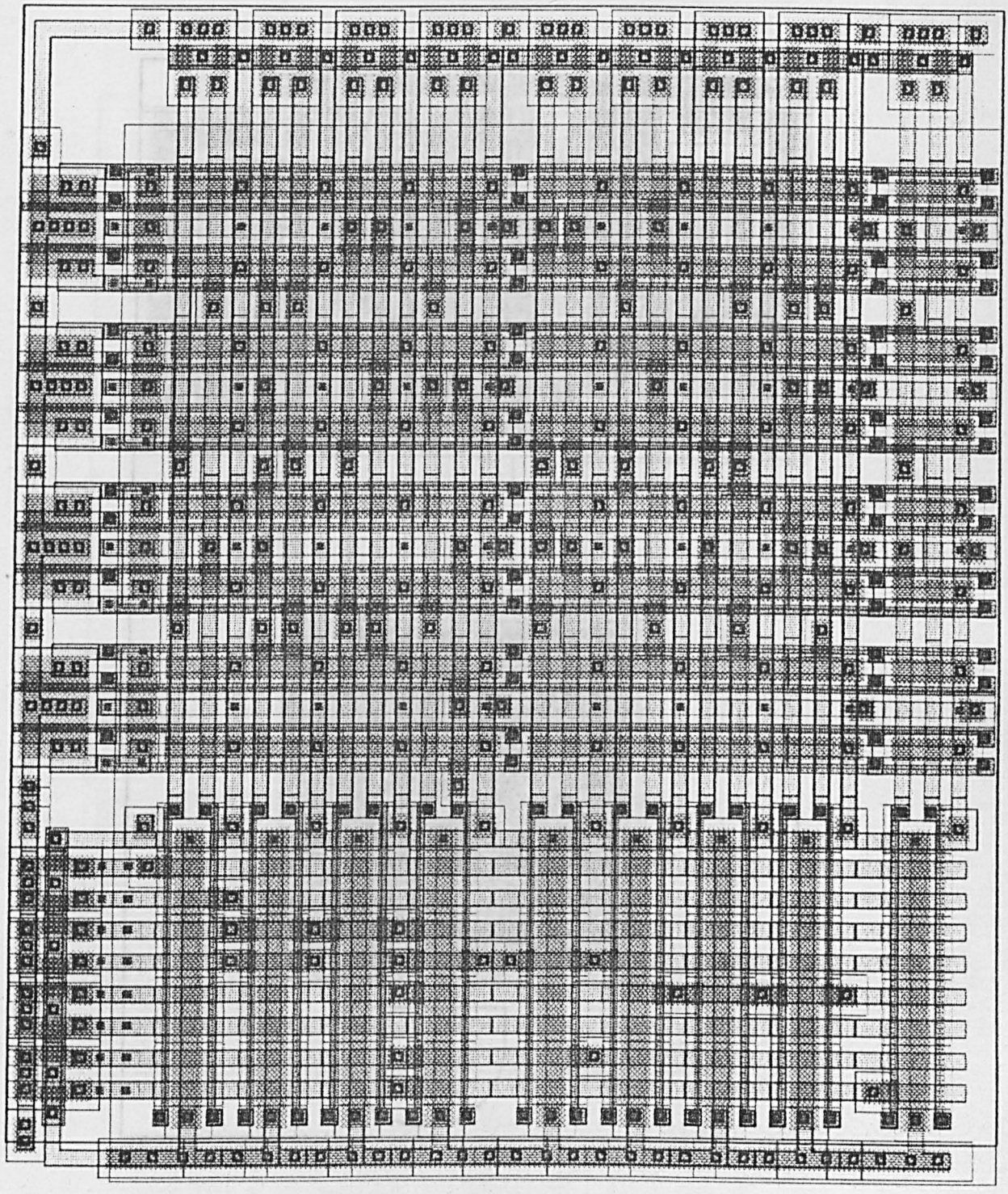


FIGURE 9.29 CONTROL UNIT PLA.

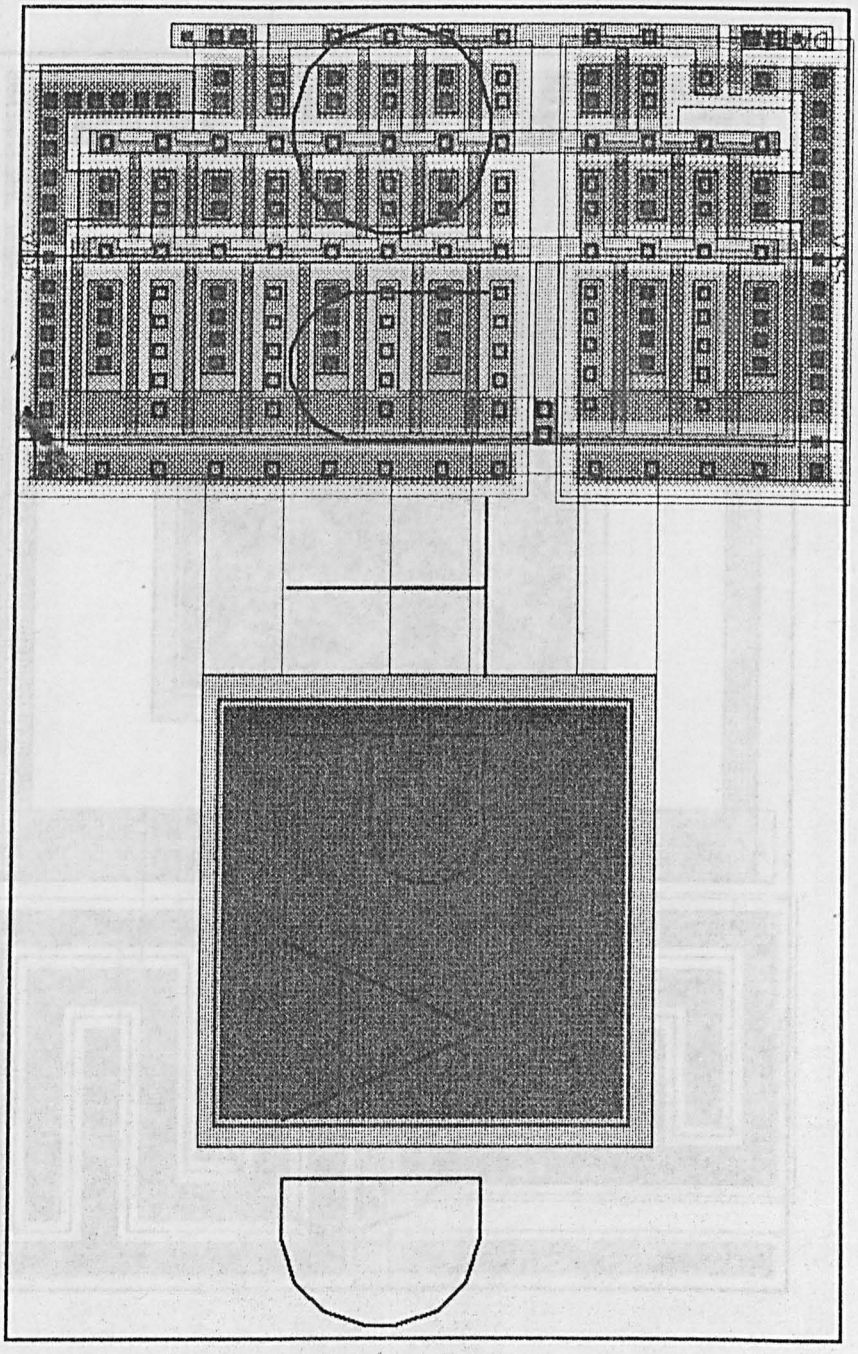


FIGURE 9.30 OUTPUT PAD.

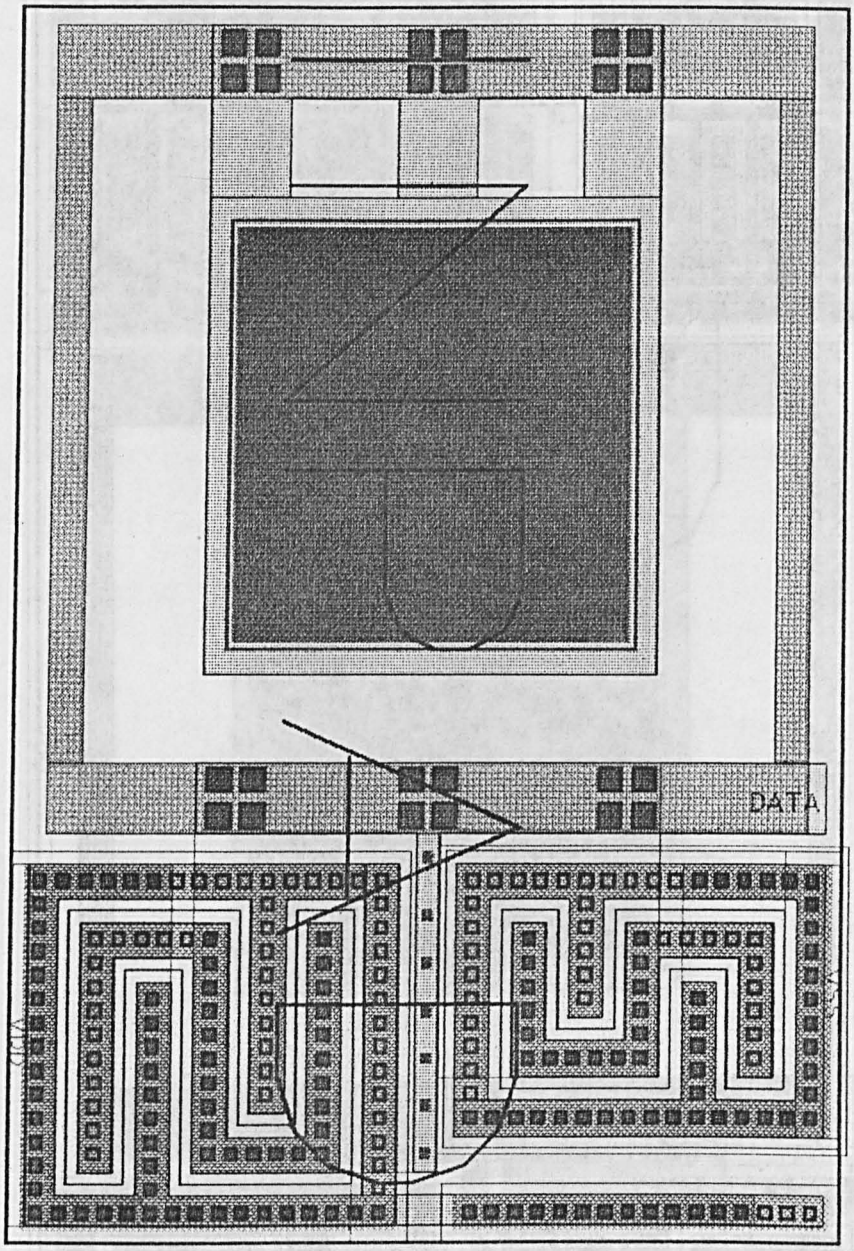


FIGURE 9.31 INPUT PAD.

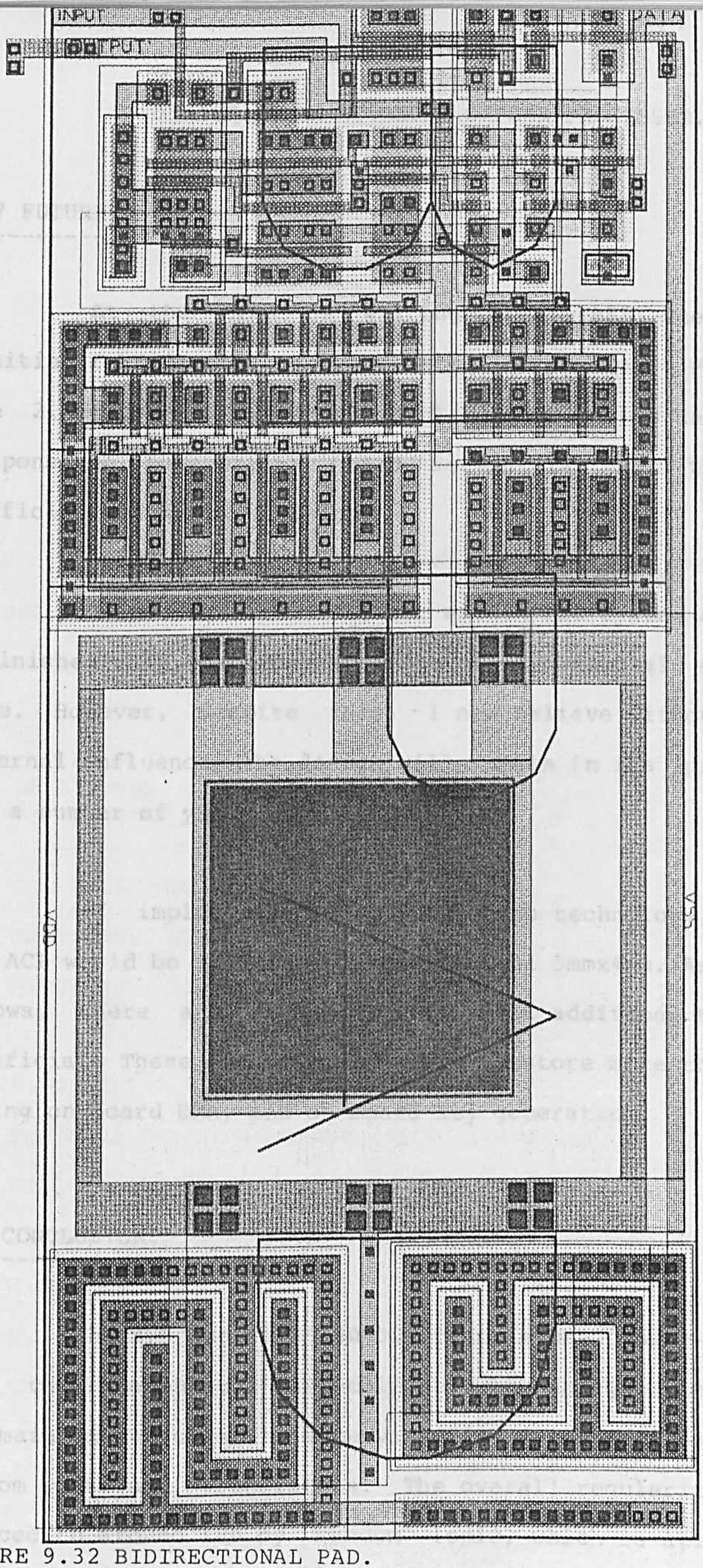


FIGURE 9.32 BIDIRECTIONAL PAD.

9.7 FUTURE.

At the time of writing (April 1985) the ACP is still awaiting fabrication, one year after completion. Problems with the 2.5 micron double-level metal bulk CMOS line has been partly responsible, however, the main reason is that there is not sufficient financial backing.

There is still interest within GEC system companies for a finished chip, particualary from the commercial communication side. However, despite this, I now believe without some major external influence, the design will remain in its present state for a number of years.

If implemented using 1.5 micron technology, the size of the ACP would be reduced to an economical 5mmx4mm. As technology allows, there are a number of small additions that would be beneficial. These include being able to store more than one key, having on-board DMA, and on board key generation.

9.8 CONCLUSION.

The ACP contains 140 000 transistors. However, it is not this that makes it a VLSI chip, but the design techniques used. The main execution unit, occupying 80% of the area uses just 100 custom designed transistors. The overall regularity factor was reduced to around 100 by 'random' logic, which is still very high

for a chip of this type.

Other attempts to design PKCs have met with little success (Ref 905), with the notable exception of Sandia Laboratories (Ref 906). The reason being is that, despite in most cases better technology, they still used LSI design methods. I am still confident that despite many new ideas, the architecture used in the ACP has optimum performance. Other designs all use global data communication, which restrict their performance to less than half the potential 50K bits per second of the ACP.

9.9 REFERENCES.

901 GEC HIRST RESEASRCH CENTRE.
EAST LANE, WEMBLEY,
MIDDLESEX. HA9 7PP. ENGLAND.

902 A.MCAULEY.
Proposal for an advanced cipher processor.
VHPIC DESIGN LABORATORY, pp.1-3, 2ND DEC.1982.

903 M.J.STUCKI & J.R.COX.
Synchronization strategies.
CALTECH CONFERENCE ON VLSI, PP.375-386, JAN.1979.

904 I.R.WILSON & A.M.ADDIMAN.

A practical introduction to Pascal.

MACMILLAN, 1981.

905 J.MAVOR, M.A.JACK & P.B.DENYER.

Introduction to MOS LSI design.

ADDISON-WESLEY, 1983.

906 R.G.MEYER.

MOS crystal oscillator design.

IEEE J. SOLID STATE CIRCUITS, VOL.15, NO.2, APR.1980.

907 R.L.RIVEST.

A description of a single-chip implementation of the RSA cipher.

LAMBDA (NOW VLSI DESIGN), VOL.1, NO.3, 1980.

908 E.F.BRICKELL.

A fast modular multiplication algorithm with application to two-key cryptography.

SANDIA LABS. ALBUQUERQUE, NEW MEXICO 87185, 1982.

SUMMARY AND FORECAST

SUMMARY AND FORECAST.

Information is now recognized as an important asset in commercial and industrial areas. The availabilily of data communication networks will allow its increased exploitation. But before the 'information age' arrives a number of important issues need to be addressed. One of these is the susceptability of modern telecommunication equipment to inexpensive electronic interception.

The determination of an individual's right to privacy is mainly a non-technical matter, but the pragmatics of providing it is the central concern of the cryptographer. This thesis has sought answers to some of the outstanding issues in cryptography. In particalar some of the theoretical, application and implementation problems associated with a Public Key Cryptosystem (PKC).

The Data Encrytion Standard (DES) answers part of the need for security. Being a government sponsored scheme, banks and financial institutions can be confident about its security. But the availability of DES hardware alone is not sufficient. Without the capabilities of a PKC, user authentication and key distribution will prove major obstacles.

The Trapdoor Knapsack (TK) of Merkle and Hellman is the only PKC capable of matching the speed of the DES chips. However, the TK-PKC suffers from serious disadvantages. In chapter 2, a

more general approach to the TK-PKC is described. Using the general knapsack problem, it was shown how the redundancy and public key size can be significantly ameliorated, while achieving a security equivalent to the original binary knapsack problem.

The security of the TK-PKC appeared to be based on a secure foundation: a NP-complete problem. Nevertheless, as so often in cryptology, the cryptanalyst proved too resourceful. In 1982 Shamir showed that the original system could be broken in polynomial time.

The most successful attacks on the secuity of the TK-PKC were on the trapdoor information. To overcome this, a new TK-PKC was designed based on transformations between the radix and residue number systems. After our paper on this was presented, a number of weaknesses were pointed out. These have been taken into account for our improved new TK-PKC described in chapter 3.

Chapter 1 described how a PKC can be used to securely exchange messages between two users, in the presence of a third party who can monitor all their traffic. Unfortunately, this system is not generalizable to more than two users. A secure broadcast of a single message to n users would require n separate cryptographic exchanges, which is clearly inefficient.

It has been shown that considerable saving in transmission time can be achieved using multi-addressed packets of information. In chapter 4, we consider how cryptography can

best be applied in this situation. We show how security or network structure can be used to advantage. We then propose a more general method, in which redundancy between the messagetext and ciphertext is used to solve the problem of key distributuion. The method effectively trades security, redundancy, number of simultaneous receivers and transmission time. An example of the broadcast cryptosystem, based on the TK-PKC, is presented.

The problems experienced by software publishers in their battle with the pirate, will frighten publishers of other forms of information. If they cannot provide security equivalent to existing published media, then the benifits of the electronic distribution will be forfeit.

Copyright is traditionally used to protect the publisher from the pirate. In chapter 5, we show how to protect information when it is in an easily copyable digital form. The software program was shown to be probably the easiest to protect. However, more general solutions were also put forward.

The future of DSP is certain to be increasingly cross disiplinary, involving VLSI and computer engineering. It is unreasonable to expect the DSP man to be an expert in the device physics of chip design. However, in order to take advantage of the capabilities of VLSI, some knowledge of its potential and constraints are needed. In Chapter 6, the author describes what is involved in a low complexity custom design.

Throughout the whole design it is necessary to be flexible, without losing the structure of top-down design. The importance of this adaptability is true at all levels of design, but is particually significant at higher levels. Specialization, while allowing one level to be optimized, will degrade the whole.

Two other points raised in chapter 6 are also worth emphasising. Firstly, the importance of simulations, not only for checking but for finding hidden complexity. Secondly, the benifits of not using logic gate primitives was shown by examples of good custom CMOS design.

The key to low complexity VLSI design is a vocabulary of good structures. However, in order to appreciate the advantages of a particular architecture, it is necessary to know something of the characteristics of a good design. This was the aim of chapter 7.

To design an architecture requires knowledge of the effect on performance and cost. Chapter 7 put forward a model which, it is hoped, will be useful for many years. The ten key parameters will, with an overall view of design, provide a method of comparing architectures.

It is important to have an efficient method for calculation in the basic arithmetic processes. There are an increasing number of applications requiring high performance arithmetic units. These special purpose machines vary widely in

application, but their internal architectures are remarkably similar. Chapter 8 discussed arithmetic architectures for these diverse applications.

New architectures, for all the basic arithmetic operations, were described. Each is characterized by using a few simple cells repeated many times. These are not specifically for cryptography, though this is certainly a good application area, but for any big number cruncher. I believe these architectures could be used for a wide variety of future DSP applications.

More than providing a vocabulary of new architectures, I believe the structures described in chapter 8 could provide the basis for higher order generalizations. Only the surface has been scratched in the work on VLSI computer arithmetic. There is ample of opportunity for rewarding and stimulating exploration of more formal ways of designing architectures.

The design of a VLSI device to implement the RSA algorithm was the major practical part of this thesis. The design was unusual, in that all the work was carried out by the author. Normally, a number of people will participate, from initial marketers to the final layout draughtsmen. This lead to slow discovery of errors, and the benefits of specialization were rejected. However, the greater insights into the overall design more than compensated for this.

The Advanced Cipher Processor (ACP) communicates with a

host microcomputer via an eight-bit bidirectional data bus, and a number of control pins. DMA transfers are supported. Internally, the device consists of the control unit, modular exponential unit, registers, oscillator, clock driver, inteface logic and 24 pads. Their design was described in chapter 9.

The heart of the ACP is the modular exponential unit. It consists of the serial-parallel multiplier and modular reducer described in chapter 8. Data flows in a regular stream, without the need for storing. When operating at maximum performance, every cell in the MEU is working all the time.

Despite careful design, the ACP is limited to speeds around 50Kbits/second. Technological advances will allow slightly faster speeds, but no dramatic improvements are likely in the next twenty years. This leaves a gap in the cryptosystem market, since many applications require a PKC operating up to 20Mbits/second. The new TK-PKC of chapter 3 might fill this gap, but continued improvements in breaking the knapsack problem leave this open to doubt.

The one regret of the past four years, is that the ACP was not fabricated. There were an number of reasons for this, but the principle explanation was the belief that commercial cryptographic devices represent a small market. Even today, ten years after publicity about the vulnerability of computer banks and communication, secure systems are still a rarity. Yet there is increasing evidence to show computer crime is becoming a big

business. The availabilty of RSA chips, would certainly be big step to helping solve the security problems.

In the future I would confidently expect to see cryptography become a standard feature in telecomunication equipment. Technology, as so often happens, has found answers to the problems it has created. Recent results have undermined the confidence in cryptography, particually the PKC. Nevertheless, though it may never be possible to quantify their security, I believe that the PKC will be an important ingredient in the information revolution.

I now believe that the PKC cannot match the speed of the conventional cryptosystem, unless based on some non-mathematical transformation. However, the benefits of a PKC will ensure that research into a fast PKC will continue.

PAPERS PRESENTED BY THE AUTHOR.

-1-

R.M.F.GOODMAN & A.J.MCAULEY.

Public key cryptosystems for multi-destination addressing and broadcast.

INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY; LES ARCS, FRANCE; JUNE 21-25, 1982.

-2-

R.M.F.GOODMAN & A.J.MCAULEY.

Broadcast public key cryptosystems for multi-destination addressing.

ELECTRONIC PRIVACY AND AUTHENTICATION TECHNOLOGY WORKSHOP; THE HATFIELD POLYTECHNIC, ENGLAND; JULY 19-20, 1982.

-3-

R.M.F.GOODMAN & A.J.MCAULEY.

Modifications to the trapdoor-knapsack public key cryptosystem.

IEEE SYMPOSIUM ON INFORMATION THEORY; ST. JOUVITE, CANADA; SEP 26-30, 1983.

-4-

R.M.F.GOODMAN & A.J.MCAULEY.

Key exchange in a broadcast environment via a public key cryptosystem.

4'th INTERNERNATIONAL CONFERANCE ON DIGITAL PROCESSING OF SIGNALS IN COMMUNICATIONS; LOUGHBOROUGH; APRIL 1985.

-5-

R.M.F.GOODMAN & A.J.MCAULEY.

Electronic publishing and copyright protection.

IEEE SYMPOSIUM ON INFORMATION THEORY; BRIGHTON, ENGLAND; JUNE 23-28, 1985.

-6-

R.M.F.GOODMAN & A.J.MCAULEY.

Bit serial integer arithmetic for VLSI public key cryptosystems.

IEEE SYMPOSIUM ON INFORMATION THEORY;

BRIGHTON, ENGLAND; JUNE 23-28, 1985.

-7-

R.M.F.GOODMAN & A.J.MCAULEY.

A new trapdoor knapsack public key cryptosystem.

ACCEPTED BY: IEE PROCEEDINGS-E; COMPUTERS AND DIGITAL TECHNIQUES.

-8-

R.M.F.GOODMAN & A.J.MCAULEY.

Key exchange in a broadcast environment via a public key cryptosystem.

SUBMITTED TO: IEE PROCEEDINGS-E; COMPUTERS AND DIGITAL TECHNIQUES.

ABBREVIATIONS.

ACP	Advanced cipher processor.
BNC	Big number cruncher.
CBA	Carry bypass adder.
CC	Conventional cryptosystem.
CLA	Carry look-ahead adder.
CMOS	Complementary MOS.
CSA	Conditional sum adder.
DES	Data encryption standard.
DIL	Dual in line chip package.
DSP	Digital signal processing.
EPROM	Erasible programmable read only memory.
GF	Galois field.
GVA	Good VLSI Architecture.
KEPROM	Key EPROM.
LSB	Least significant bit.
MA743	Mask allocation for ACP.
MG	Modulus generator.
MOA	Multi-operand adder.
MOS	Metal oxide silicon technology.
MR	Modular reducer.
MSB	Most significant bit.
NMOS	N-channel MOS.
NPM	New parallel multiplier.
NSPM	New serial parallel multiplier.
PKC	Public key cryptosystem.

PLA	Programmable logic array.
PM	Parallel multiplier.
PROM	Programmable read only memory.
RSA	Rivest-Shamir-Adleman PKC.
SPD	Serial-parallel divider.
SPM	Serial parallel multiplier
TK-PKC	Trapdoor knapsack PKC.
TRC	Tamper resistant computer.
TTL	Transistor transistor logic.
VLSI	Very large scale integration.

APPENDIX

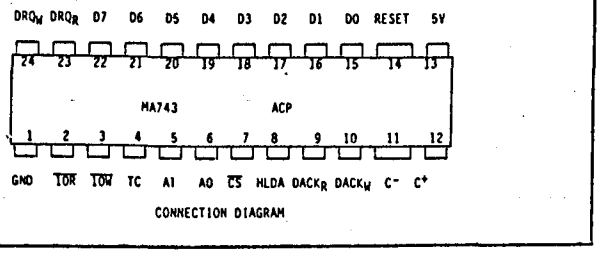
A

MA743 ADVANCED CIPHER PROCESSOR

PRELIMINARY DATA

DISTINCTIVE CHARACTERISTICS

* 50 K bits/s data rate : consistent with use in X25 wide area network
* Message secrecy
* Message and user authentication
* Simplified key distribution
* Single chip encryption/decryption
* MULTIBUS[1] compatibility



CONNECTION DIAGRAM

PIN	I/O	DESCRIPTION
D0-D7	I/O	Bidirectional three state line. Input for keys. Input/output for plain text/cipher text
A1,A0	I	Used to select distination of data bus, in direct mode operation
HLDA	I	Selects between direct (low) and handshake (high) mode of operation
$\overline{CS}$	I	Enables the IOR/IOW, in handshake mode
$\overline{DACK_R}$	I	Enables the IOR, in handshake mode
$\overline{DACK_W}$	I	Enables the IOW, in handshake mode
$\overline{IOR}$	I	I/O Read, used to strobe data out
$\overline{IOW}$	I	I/O Write, used to strobe data in
TC	I	End of current block transfer
RESET	I	Initialize
DRQ_R	O	Block of processed data available on stack
DRQ_W	O	Can write block of data on to stack
V_{DD}	PS	5 V power supply
V_{SS}	PS	0 V power supply
Cex^+		External timing crystal/External clock
Cex^-		External timing crystal

TABLE 1 : PIN FUNCTIONS

GENERAL DESCRIPTION

The MA743 Advanced Cipher Processor (ACP) is a special purpose 24 pin CMOS chip, designed to overcome the serious deficiencies of traditional encryption methods such as the US Data Encryption Standard. It is based on the best of the Public Key Cryptosystems, the RSA algorithm, which has defeated all attacks on its integrity.

A secure system based on the ACP would operate as follows. Each user would generate a pair of keys, one of which would be kept secret, the other would be published in a directory. Anyone wishing to communicate with you could then encipher a message encrypted under your public key. Since only you know the secret inverse key, you are the only one able to decipher the message. The problems of secure key distribution do not then arise. By sending a message 'deciphered' under your secret key, anyone can use your public key to read your message. However, since only you could have generated the 'deciphered' message, this "electronic signature" may be used for authentication.

The ACP will interface easily to MULTIBUS[1] and other microprocessor bus systems, and will facilitate direct handshake with a DMA controller for both input and output. With the full 512-bit key, data rates up to 50K bits/s will be possible. With a reduced encryption key, encryption speeds will exceed 1M bit/s.

Applications include: Electronic Funds transfer, Computer Network Communications and secure Database Systems.

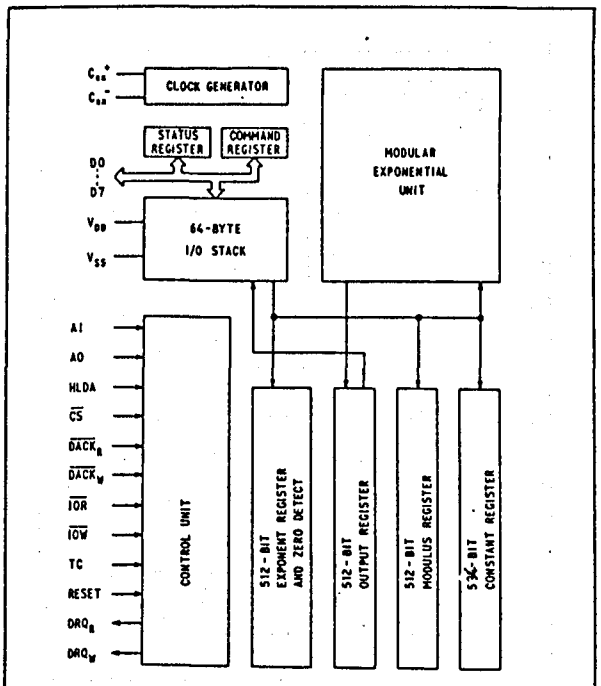


FIGURE 1 : ACP BLOCK DIAGRAM

[1]MULTIBUS is a trademark of Intel Corporation

FUNCTIONAL DESCRIPTION

The ACP is a high performance device for performing modular exponentiation on 512-bit numbers. Its speed of execution is proportional to the number of bits in the exponent. It communicates with the host microprocessor system through a bidirectional eight-bit data bus, and a number of control pins (see Table 1). Figure 1 shows the internal layout of the ACP, all internal paths are bit serial.

READING AND WRITING DATA

a) Direct mode

In the direct mode, the HLDA line must be inactive and the $\overline{CS}$ line active. Data can then be read or written on the rising edge of the $\overline{IOR}/\overline{IOW}$ pulse (see figure 2). The destination is selected using the A_0A_1 address lines (see Table 2). If data is written into the stack, the final destination is determined by the contents of the command register (see Table 4). Before data is written the contents of the status register (see Table 5) should be checked.

b) Handshake mode

With the HLDA line active, the CS line is disabled (see Table 3). Communication is then through the Data Request (DRQ_R,DRQ_W) and Data Acknowledge ($\overline{DACK_R}$,$\overline{DACK_W}$) lines. With Data Acknowledge active, data is again strobed on the rising edge of the $\overline{IOR}/\overline{IOW}$ pulse (see Figure 3). The Data lines are automatically connected to the stack.

A_1,A_0	DESTINATION/SOURCE
0 0	Command Register/Status register
X 1	Stack/Stack
1 1	Last byte (stack)

TABLE 2 : ADDRESS DECODE

HLDA,$\overline{CS}$	OPERATION
0 1	Ignore system, except for DRQ as necessary
0 0	Reads data and address with $\overline{IOR}/\overline{IOW}$ pulse
1 X	DMA active - data strobed with $\overline{DACK}$ and $\overline{IOR}/\overline{IOW}$

TABLE 3 : CHIP STATUS DECODE

X	X	X	X	KR	MR	EP	MEU

BIT 1 : MEU = Write to modulation exponential unit
BIT 2 : EP = Write to exponent register
BIT 3 : MR = Write to modulus register
BIT 4 : KR = Write to constant register

TABLE 4 : COMMAND REGISTER DECODE

DRQ_W	DRQ_R	SF	SB	X	MB	OW	OF

BIT 1 : OF = Output ready
BIT 2 : OW = Output register full
BIT 3 : MB = MEU busy
BIT 4 : X
BIT 5 : SB = Stack busy
BIT 6 : SF = Stack full of input
BIT 7 : DRQ_R = Read request to DMA
BIT 8 : DRQ_W = Write request to DMA

TABLE 5 : STATUS REGISTER DECODE

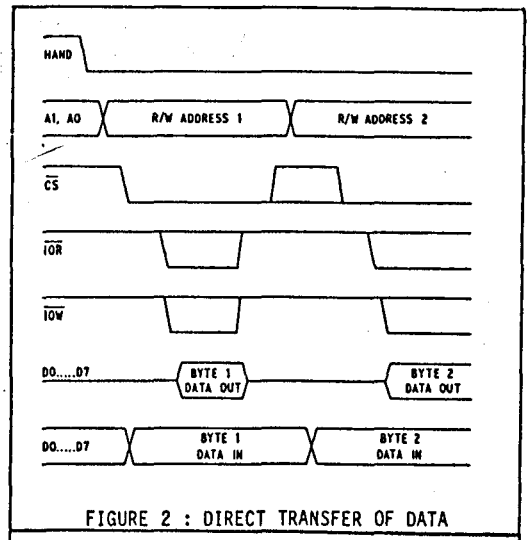


FIGURE 2 : DIRECT TRANSFER OF DATA

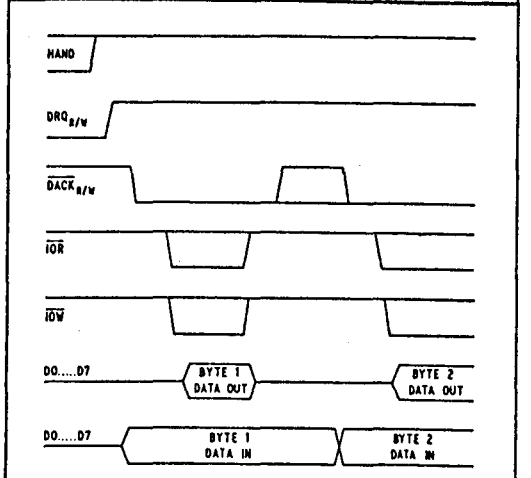


FIGURE 3 : TRANSFER OF DATA IN HANDSHAKE MODE

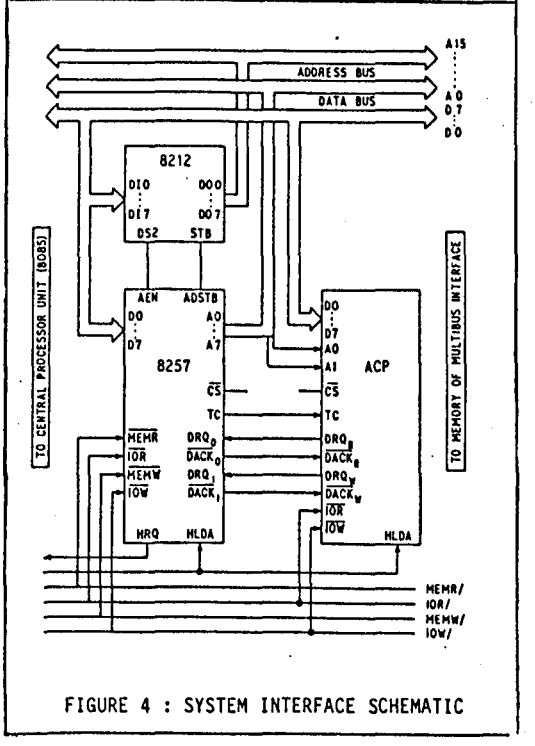


FIGURE 4 : SYSTEM INTERFACE SCHEMATIC

For further information please contact : VHPIC Design Laboratory, GEC Research Laboratories, Hirst Research Centre, East Lar
Wembley, Middlesex, HA9 7PP.
Tel: No 01-904 1262

APPENDIX B

```
MA743:=
    Begin (US)

! ISPS Description of the MA743 Advanced Cipher Processor (ACP)

! Page 1:  Carriers associated with the ACP chip.
! Page 2:  System carriers, and implementation declerations.
! Page 3:4 System operation. Page 3 simulates the operation of a CPU, by
!               putting the necessry signals on the acp's inputs. Page 4
!               similarly simulates the operation of a DMA, and chip
!               select logic.
! Page 5:9 ACP description. Pages 5&6 describes the control logic of the
!               ACP associated with exchanging data with the outside world.
!               Page 7 describes the control logic associated with the
!               internal transfer of data. Page 8 describes the operation
!               of the MEU, incalculating the modular exponentiation of the
!               input data. Finally Page 9  describes the subroutines used
!               in the description.
! Page 10: Program description. Simulates the program of the CPU.



**ACP.STATE**


!ACP registers

    SR<7:0>,                            !STATUS REGISTER
        OF<>  :=SR<0>,                  !OUTPUT READY
        OW<>  :=SR<1>,                  !OUTPUT REG. FULL
        MB<>  :=SR<2>,                  !MEU BUSY
        SB<>  :=SR<4>,                  !STACK BUSY
        SF<>  :=SR<5>,                  !STACK FULL OF INPUT
        DRQR<>:=SR<6>,                  !WRITE REQ. TO DMA
        DRQW<>:=SR<7>,                  !READ REQ. TO DMA
    CR<7:0>,                            !COMMAND REG.
    ET<15:0>,                           !EXPONENT REG. (TEMP)
    EP<15:0>,                           !EXPONENT REG. (PERM)
    OR<19:0>,                           !OUTPUT REG.
    MR<15:0>,                           !MODULUS REG.
    KR<15:0>,                           !CONSTANT REG.
    STC1:0]<7:0>,                       !STACK

!Modular Expontiatial Unit registers

    ZA<39:0>,                           !PRODUCT
    ZB<19:0>,                           !MULTIPLIER
    ZD<19:0>,                           !MULTIPLICAND

    DA<19:0>,                           !RESULT
    DB<15:0>,                           !MODULUS
    DD<39:0>,                           !DIVIDEND
```

```
**EXTERNAL.STATES**

!Multibus signals                                                Page 2

    D<7:0>,                          !DATA BUS
    A<15:0>,                         !ADDRESS BUS
    IOW<>,
    IOR<>,

!Control pins

    HLDA<>,                          !SELECTS HANDSHAKE MODE IN ACP
    DACKR<>,                         !ACK. BY DMA OF DRQR
    DACKW<>,                         !ACK. BY DMA OF DRQW
    TC<>,                            !END OF BLOCK TRANSFER
    RST<>,                           !RESET

**Memory.State**

    RAM[0:15]<7:0>,                  !EXTENAL MEMORY

**IMPLELENTATION.DECLERATIONS**

    P<6:0>,                          !CLOCK
    Q<4:0>,                          !DELAY VAR.

    A1<3:0>,
    A2<3:0>,

    RW<>,                            !READ OR WRITE FLAG
    DV<7:0>,                         !CPU'S DATA
    ADDE<15:0>,                      !END ADDRESS   (HANDSHAKE MODE)
    ADDS<15:0>,                      !START ADDRESS (HANDSHAKE MODE)
```

! DIRECT READING AND WRITING OF ACP (Remember to check SR)

```
    DIR:=                                       !(A) <---> D
        BEGIN
        HLDA=0 NEXT                             !DIRECT MODE
        CS() NEXT                               !CALCULATE CS<> FROM ADDRESS
        DECODE RW =>
            BEGIN
            0:= BEGIN                           !WRITE TO ACP (DIR)
                IOW=0 NEXT
                WT() NEXT                       !WAIT FOR EXT PROC. TO LATCH D
                IOW=1
                END,
            1:= BEGIN                           !READ (DIR)
                IOR=0 NEXT
                WT() NEXT
                IOR=1 NEXT
                DV=0                            !STORE DATA BUS INTO (ADDM)
                END
            END NEXT
        CS=1
        END,
```

```
! HANDSHAKE READING AND WRITING (REMEMBER TO CHECK SR)

    HAND:=                                          !(STACK) <-> (RAM[ADDS..ADDE])
        BEGIN
        A=ADDS NEXT
        REPEAT
            BEGIN
            IF A EQL ADDE => TC=1 NEXT              !SIGNAL LAST BYTE
            HLDA=1 NEXT                             !HANDSHAKE MODE
            CS() NEXT                               !(CS IS INEFFECTIVE HOWEVER)
            DECODE RW =>
                BEGIN
                0:= BEGIN                           !WRITE (HAND)
                    WAIT (DRQW) NEXT                !WAIT FOR ACP READY
                    D=RAM[A] NEXT                   !NEXT BYTE OF DATA ON DATA BUS
                    DACKW=0 NEXT
                    IOW=0 NEXT
                    WT() NEXT
                    IOW=1 NEXT
                    DACKW=1
                    END,
                1:= BEGIN                           !READ (HAND)
                    WAIT (DRQR) NEXT
                    DACKR=0 NEXT
                    IOR=0 NEXT
                    WT() NEXT
                    IOR=1 NEXT
                    DACKR=1 NEXT
                    RAM[A]=D                        !STORE PROCESSED DATA
                    END
                END NEXT
            TC=0 NEXT
            A=A+1 NEXT                              !INCREMENT DMA ADDRESS
            IF (A GTR ADDE) => LEAVE HAND           !FINISHED BLOCK TRANSFER
            END
        END,


!SIMULATE DECODE LOGIC

    CS<>:=
        BEGIN
        DECODE A =>
            BEGIN
            "000?    := CS=0,                       !SELECT CHIP
            OTHERWISE:= CS=1
            END
        END,
```

```
    PROCESS EXT:=
      BEGIN
      REPEAT
        BEGIN
        IF RST OR CR<5> => RESET() NEXT
        DECODE HLDA =>
          BEGIN
          0:= BEGIN                                          !DIRECT MODE
              IF NOT CS =>                                   !IF CHIP SELECTED
                BEGIN
                DECODE IOR @ IOW =>
                    BEGIN
                    '10:=                                    !WRITE
                      BEGIN
                      DECODE A<1:0> =>                       !DECODE ADD. BUS
                        BEGIN
                        '00:= CR=D,                          !LOAD CR
                        '01:= BEGIN
                              SB=1 NEXT                      !TAKE STACK
                              PUSH()                         !LOAD STACK
                              END,
                        '1?:= BEGIN
                              SB=1 NEXT                      !TAKE STACK
                              PUSH() NEXT
                              SF=1 ;                         !END OF TRANSFER
                              DRQW=0
                              END
                        END NEXT
                      WAIT (IOW)                             !SIMULATE LATCH
                      END,
                    '01:=                                    !READ
                      BEGIN
                      DECODE A<1:0> =>
                        BEGIN
                        '00:= D=SR,
                        '01:= POP(),
                        '1?:= BEGIN
                              POP() NEXT
                              SB=0 ; SF=0 NEXT
                              DRQR=0
                              END
                        END NEXT
                      WAIT (IOR)                             !SIMULATE LATCH
                      END,
                    OTHERWISE:= NO.OP()
                    END
                END
              END,
```

```
        1:= BEGIN                                   !HANDSHAKE MODE.
            IF DACKW OR DACKR =>
              BEGIN
              DECODE IOR @ IOW =>
                  BEGIN
                  '10:= BEGIN                       !WRITE
                        SB=1 NEXT
                        PUSH() NEXT
                        IF TC => BEGIN
                                 SF=1 ;
                                 DRQW=0
                                 END NEXT
                        WAIT (IOW)
                        END,
                  '01:= BEGIN                       !READ
                        POP() NEXT
                        IF TC => BEGIN
                                 SB=0 ; SF=0 NEXT
                                 DRQR=0
                                 END NEXT
                        WAIT (IOR)
                        END,
                  OTHERWISE:= NO.OP()
                  END
              END
            END
          END
      END
    END,
```

```
    PROCESS INT:=
      BEGIN
      REPEAT
        BEGIN
        DECODE MB @ SF @ OF @ DRQR =>
            BEGIN
            '01??:= BEGIN                                   !STACK-->INTERNAL
                    DECODE CR =>
                        BEGIN
                        "01:=   BEGIN                       !STACK-->MEU
                                MB=1 NEXT                   !MEU READY FOR USE
                                MEU()                       !START EXPONENTIATION
                                END,
                        "02:=   ( EP=ST[0]@ST[1] ),         !STACK-->EREG
                        "04:=   ( MR=ST[0]@ST[1] ),         !STACK-->MREG
                        "08:=   ( KR=ST[0]@ST[1] ),         !STACK-->KREG
                        OTHERWISE:= NO.OP()
                        END NEXT
                    SB=SF=0
                    END,
            '?01?:= BEGIN                                   !OUT.REG.-->STACK
                    IF NOT SB =>
                        BEGIN
                        SB=1 NEXT
                        DRQW=0 NEXT
                        ST[0] @ ST[1] = QR NEXT
                        OW=0;OF=0 ;
                        DRQR=1
                        END
                    END,
            '?000:= DRQW=1,                                 !STACK FREE
         OTHERWISE:= NO.OP()
            END
        END
      END,
```

BRITISH LIBRARY

IMAGING SERVICES NORTH

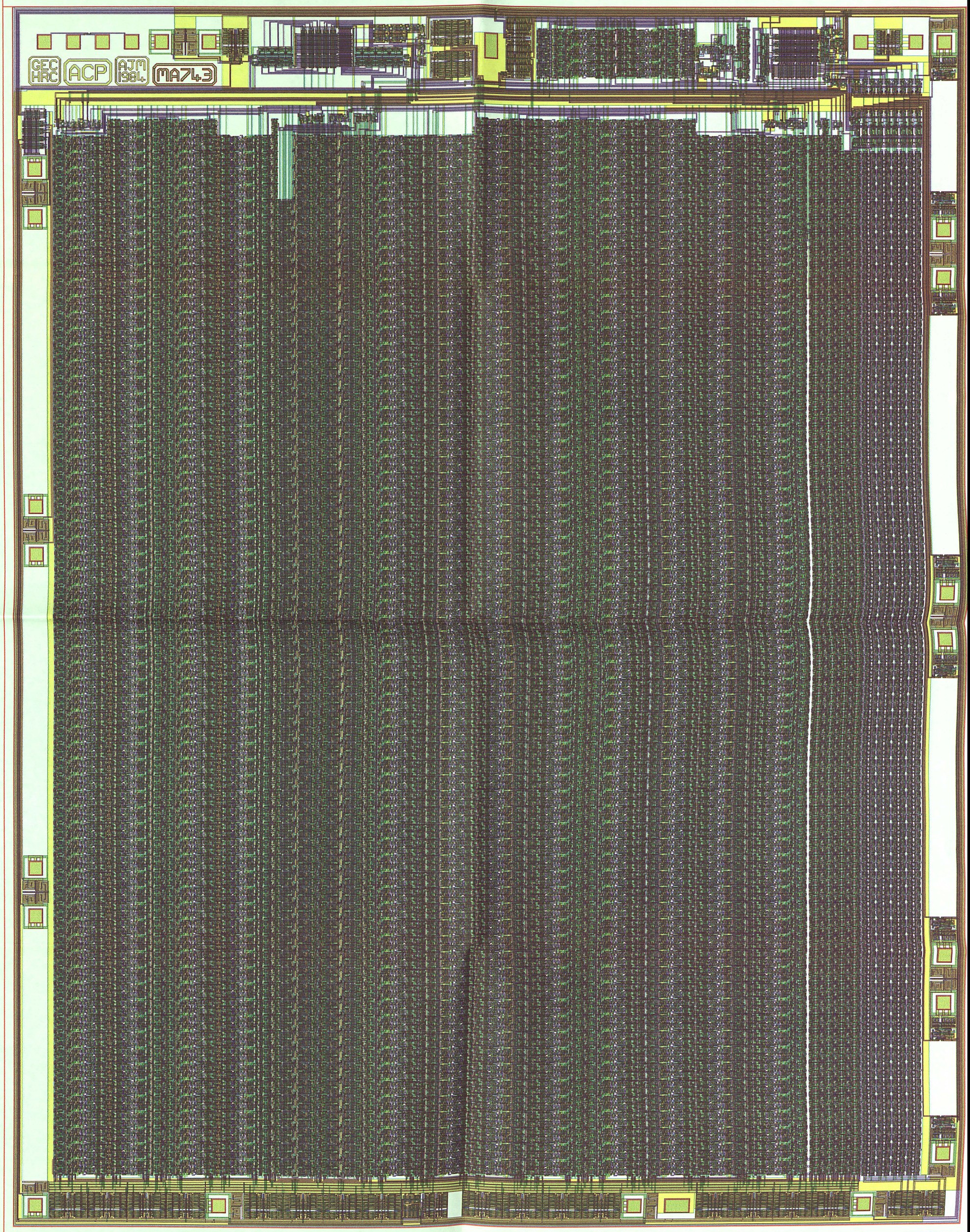
Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

PAGE MISSING IN ORIGINAL

```
!
**RESET.CYCLE**

    RESET:=
        BEGIN
        DACKW=DACKR=CS=IOW=IOR=1 ;
        TC=HLDA=0 NEXT
        SR="80 ;                                    !RESET COMPLETE
        CR=0 NEXT
        EXT() ;                                     !START ACP'S EXTERNAL PROCESS
        INT() NEXT                                  !START INTERNAL PROCESS
        STOP()
        END,

**STACK.OPERATIONS**

    POP:=
        BEGIN
        DRQW=0 NEXT
        D=ST[1] NEXT
        ST[1]=ST[0]
        END,

    PUSH:=
        BEGIN
        DRQR=0 NEXT
        ST[1]=ST[0] NEXT
        ST[0]=D
        END,


**DELAY.SIMULATION**

    WT:=
        BEGIN
        P=1 NEXT
        REPEAT
                BEGIN
                P=P+1 NEXT
                IF P<6> => LEAVE WT                 !DELAY BY 2^7 OPS.
                END
        END,


    WT1:=
        BEGIN
        Q=1 NEXT
        REPEAT
                BEGIN
                Q=Q+1 NEXT
                IF Q<4> => LEAVE WT1                !DELAY BY 2^7 OPS.
                END
        END,
```

```
!
**PROGRAMMING**

    RUN:=
        BEGIN
        RAM[5]  @  RAM[4] ="0002 ;                    !EXPONENT
        RAM[7]  @  RAM[6] ="0008 ;                    !MODULUS
        RAM[9]  @  RAM[8] ="0CDA ;                    !CONSTANT
        RAM[11] @  RAM[10]="0003  ;                   !DATA
        DV=D=2 ;                                      !KEY DESTINATION
        A2=4 ;                                        !INIT. VAL.
        RW=0 NEXT                                     !WRITE TO ACP
        KEY() NEXT                                    !LOAD KEYS
        TRAN()                                        !TRANSFER DATA
        END,

    KEY:=
        BEGIN
        REPEAT
            BEGIN
            A=1 NEXT                                  !WRITE TO ACP'S CR
            DIR() NEXT                                !LOAD KEY DESTINATION
            ADDS=A2 ; ADDE=A2+1 NEXT                  !SOURCE OF KEY
            WAIT (NOT S8) NEXT                        !STACK FREE ?
            HAND() NEXT                               !LOAD KEY
            A2=A2+2 ; D=DV=DV*2 NEXT                  !NEXT KEY SOURCE
            IF D EQL 16 => LEAVE KEY                  !FINISHED ?
            END
        END,

    TRAN:=
        BEGIN
        A=1 ;
        D=1 NEXT
        DIR() NEXT
        REPEAT
            BEGIN
            RW=0 NEXT
            ADDS=10 ; ADDE=11 NEXT
            HAND() NEXT
            RW=1 NEXT
            HAND()
            END
        END,

    END,
```

APPENDIX C


GEC
HRC
ACP
AJM
1984
MA743