MODELS FOR VLSI IMPLEMENTATION OF RESIDUE NUMBER SYSTEM ARITHMETIC MODULES

M.A. Bayoumi, G.A. Jullien and W.C. Miller

Department of Electrical Engineering, University of Windsor
Windsor, Ontario, N9B 3P4, Canada

ABSTRACT

This paper discusses the implementation of RNS arithmetic modules using VLSI technology. The modules are based on the interconnection of read-only memory look-up tables. The paper first outlines a memory model for a single look-up table which allows the selection of the most efficient layout for memories which do not have power of 2 dimensions. The paper then discusses various examples of interconnected memory modules with associated optimizing layout algorithms. Finally, an example is given of the application of one of the modules to a large prime modulus multiplier.

1. INTRODUCTION

With recent advances in VLSI technology, it has become evident that digital signal processing (DSP) algorithms can be implemented in cost-effective technology [1-4]. Such implementations operates faster, consumes less power, and are far more reliable than their predecessors (SSI and MSI). Those advantages are due in part to the great reduction in the number of connections among chips.

In building VLSI chips for DSP applications, two approaches can be followed. The first is to build a processor with large arithmetic capabilities and having flexible data handling and function sequencing architectures. A second approach is to partition the application requirements so that the processing functions are distributed to several single chip processors. Signal processing applications are particularly well suited to this type of partitioning. The second approach has added advantages of modularity both for growth and future applications.

Residue Number Systems (RNS) have been proven by many authors [5-7] to be a successful implementation for high speed DSP applications. Combining a modular structure and the technological advantages offered by VLSI technology leads to enhancement in RNS implementation in speed, cost, power dissipation, and chip density. The VLSI approach is promising [8] as RNS supports the main VLSI design features:

1. RNS has a parallel nature where the arithmetic operations are performed independently for each module which supports distributed processing. This concurrency minimizes overall execution time and results in higher data throughputs.
2. The memory intensive architecture is very suitable for VLSI, as the memory modules achieve higher level of integration than any other computational element, this is due to the regular geometry, and straight, simple and regular interconnections.
3. The interconnections topology among modules are regular and straight which enhance the VLSI implementation.

Designing a VLSI RNS system requires developing design methodology based on a set of rules, guidelines and disciplines which makes the design of a complex system manageable and produces the most efficient design by exploring all the viable design alternatives.

This paper looks at modelling look-up table arithmetic chips either for special or general purpose useage. A model for a memory module required for storing a certain look-up table implemented in NMOS technology, has been developed featuring the performance measures in this environment: (1) area required for computational modules and the interconnection among them, and (2) time-delay contributed by the computational modules and the interconnections among these modules. A procedure has been developed to obtain the most efficient design for an RNS chip. A set of multi-look-up table chips have been proposed to be used as building blocks for implementing DSP algorithms on a modular basis.

As an application, a Number Theoretic Transform (NTT) multiplication algorithm has been proposed and then implemented using J-modules.

GLOSSARY

α_o	Basic cell length
A_o	Basic cell area
H	The height of the memory module in unit length
W	The width of the memory module in unit length
Hb	The number of bits in the vertical direction
Wb	The number of bits in the horizontal direction
A	Is the area of the memory module in unit area
T	Is the access time of a certain memory in unit time

γ, β Is the grouping of the hierarchical structure at each level in W and H directions, respectively

AT Area time-delay product, AT = A x T

m Is the Modulus

ω Is the word width of the memory

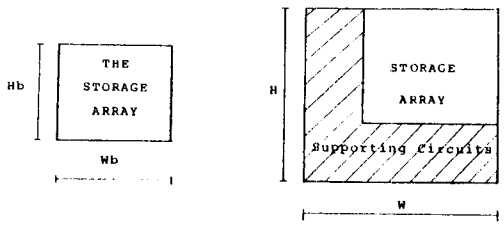
$\lceil\ \rceil$ Take next highest integer

log Log_2

2. MEMORY MODEL

We will use the hierarchical structure [9], which is based on the fact that: a bus wire of length $K\alpha_o$ that has K driving transistors of area α_o and one receiver of area $K\alpha_o$ operates in K time units, e.g., a unit-sized transistor can drive another unit-sized transistor at the end of a unit length wire in just one time unit. Or, a bus driver tree with a branching factor of K has a delay of K for each level that a signal ascends from a leaf-node driver.

The storage array is organized in this hierarchical fashion to avoid large delays. Level 0 (the leaves of the tree) comprises the basic cells for storing bits, each of square area $A_o = \alpha_o \times \alpha_o$, where α_o is the length of the basic cell in unit length. The model is organized as follows:



where H, W are the height and width of the memory module in unit length, respectively.

Hb, Wb are the number of bits in the vertical and horizontal directions, respectively. The module's parameters are: W, H, A, T, γ, β, and N, where W and H are as defined above, and A is the area of the module in unit area: A = W x H, T is the access time of the memory module, in unit time. γ, β is the grouping at each level (branching factor of the tree) in W and H directions, respectively. N is the number of levels the memory is organized in the hierarchical structure.

Following the same procedure as in [9, 12], we get:

$$W = \left(\alpha_o + \frac{1}{\gamma - 1} + \frac{2\gamma - 1}{(\gamma - 1)^2} \log \beta \right) \gamma^N \qquad (1)$$

$$H = \left(\alpha_o + \frac{1}{\beta - 1} + \frac{2\beta - 1}{(\beta - 1)^2} \log \gamma \right) \beta^N \qquad (2)$$

$$T = \text{Max} \left\{ N\alpha_o \gamma,\ N\alpha_o \beta \right\} \qquad (3)$$

3. A LOOK-UP TABLE LAYOUT

The layout of a memory module for storing a modulo m look-up table is controlled by some assumptions expressing the constraints and features of the model; they are denoted by A_i and outlined as follows:

A1: neither Wb nor Hb is necessarily a power of 2, the same for the storage array size (Wb x WH).

A2: The memory word width ω is given by:

$$\omega = \lceil \log (m-1) \rceil \qquad (4)$$

A3: The minimum γ or β is ω.

A4: $Wb \geq \omega$, $Hb \geq \omega$ (5)

A5: one of the two dimensions is selected as a reference (e.g., Wb) , so, $Wb = k\gamma^N$, where k is an integer constant.

This layout of a memory module is not unique. It can have various options, each has different performance as discussed below.

Definition 1: A layout option L is is defined by 7 tuples of the form {Wb, Hb, W, H, A, T, AT} where AT is the area time-delay product, the other parameters as previously defined.

Definition 2: An option Y is called redundant, if there is another option X such that: $W_Y \geq W_X$, $H_Y \geq H_X$, and $T_Y \geq T_X$ so, $X \supseteq Y$ means Y is redundant and X offers either better performance or at least the same performance as Y.

Redundant options can be cancelled, since they do not offer better performance. Table 1 shows some layout options of a memory module for modulo 7. We notice the redundancy included in the table, e.g., $X \supseteq Y$; $P \supseteq O, R, Q$; $G \supseteq J$. Fig. 1 shows the relations among the module's parameters.

In [12] it has been shown that Wb is bounded by m^2. The proof is straight forward as the number of bits of a look-up table modulo m is $m^2 \lceil \log (m-1) \rceil$ and the minimum number of bits in any dimension is $\lceil \log (m-1) \rceil$ (A3). Table 1 shows that all the options following option Q (for N = 1) are redundant.

To get all the viable options of the memory layout required for storing a look-up table modulo m algorithm 1 has been developed shown in Fig.2. Table 2 shows the viable layout options for a look-up table modulo 7.

TABLE 1 Some Layout Options for Modulus 7 Without Any Constraints

	Wb	Hb	W	H	A	T	AT
for N = 1							
	3	49	27	54.1	1461.96	49	7163.608
	6	25	20.4	32.4	661.41	25	1653.517
	9	17	22.1	26.8	592.32	17	1006.936
	12	13	22.2	23.1	513.41	13	667.433
	15	10	24.9	20.5	511.30	15	766.950
	18	9	17.8	22.1	613.30	18	1103.938
X	21	7	28.5	20.8	593.11	21	1245.540
Y	24	7	31.4	20.8	654.14	24	1569.926
	27	6	74.4	20.4	701.54	27	1894.151
	30	5	37.3	20.3	758.64	30	2275.918
	33	5	40.3	23.1	932.30	33	3076.575
	36	5	43.3	23.1	1001.04	36	3602.742
P	39	4	44.2	24	1060.45	39	4135.772
Q	42	4	47.2	24	1131.13	42	6754.931
R	45	4	50.2	24	1203.84	45	5417.293
Q	48	4	53.1	24	1275.60	48	6122.862
	51	3	56.1	27	1515.80	51	7730.585
	54	3	59.1	27	1596.59	54	8621.556
	57	3	62.1	27	1677.39	57	9561.133
	60	3	65.1	27	1758.22	60	10549.310
	63	3	68.1	27	1839.06	63	11586.080
	66	3	71.1	30.8	2186.57	66	14431.380
	69	3	74.1	30.8	2278.68	69	15722.870
	72	3	77.1	30.8	2370.79	72	17089.700
N = 2							
	9	25	47.3	59.4	2805.47	10	28054.72
	36	9	74.9	47.3	3538.08	12	42457
G	81	4	112.6	56	6307.87	18	113541.7
J	144	4	184.5	56	10329.91	24	247917.9
N = 3							
	27	8	74.2	64	4752	9	42767.96
N = 4							
	81	16	222.7	128	28511.98	12	342143.8

4. SELECTING THE OPTIMAL LAYOUT

It is a key problem to minimize both the time and area of a VLSI layout; which is very difficult to minimize both of them simultaneously. The designer selects the optimal solution for a certain design according to the imposed constraints. If there is no constraints the minimum are time-delay product is selected as a measure of performance.

TABLE 2 The Viable Layout Options of a Memory Module for Modulus 7 by Applying Algorithm 1

	Wb	Hb	W	H	T	A	AT
N = 1	3	49	27	54.1	49	1461.96	71636.08
	6	25	20	32.4	25	661.41	16545.18
	9	17	22	26.8	17	592.32	10069.36
	12	13	22.2	23.1	13	513.41	6674.33
	15	10	24.9	20.5	15	511.30	7669.50
	18	9	27.8	22.1	18	613.30	11039.38
	21	7	28.5	20.8	21	593.11	12455.40
	27	6	34.4	20.4	27	701.51	18941.51
	30	5	37.3	20.3	30	758.84	22759.19
	39	4	44.2	24	39	1060.45	41357.72
N = 2	9	25	47.3	59.4	10	2805.47	28054.72
	36	9	74.9	47.3	12	3538.09	42457.01
	81	4	112.6	56	18	6307.87	113541.70
N = 3	27	8	74.2	64	9	4752	42767.96

5. MULTI-LOOK-UP TABLES MODULES

In VLSI environment, the designer has control on the chip function, geometry, and topology, so, some modules are proposed in which several independent look-up tables are stored in the same chip with appropriate interconnections. This approach has the advantage of saving in the cost of interconnections among the different look-up tables. The topology of these modules are selected according to the requirements of most DSP algorithms, these modules are: the joint module (J), fork module (F), and joint-fork module (JF) as discussed below:

5.1. Joint Module

Fig. 4 shows a J-module having three look-up tables, the area and time-delay of this module is calculated (depending on the selected modules) as follows:

1. Calculate H_1, W_1, H_2 and W_2 of Table 1 and Table 2 by algorithm 1 and selecting the minimum AT.
2. The channel dimensions are calculated based on the worst case and the techniques in [10] as follows:

$$\text{channel width } W_{ch} = 1 + 2\lceil \log (m-1) \rceil \quad (6)$$

$$\text{channel height } H_{ch} = 3\lceil \log (m-1) \rceil \quad (7)$$

Assuming that the time-delay is linear with the wire length [10, 11], then:

$$\text{channel time-delay } T_{ch} = K\left(1 + 5\lceil \log (m-1) \rceil\right) + 1 \quad (8)$$

where K

$$= \frac{\text{Cap. of Unit Length}}{\text{Cg of the Tr}} \left(\text{consider it } \frac{1}{4} \text{ as [10]} \right)$$

TABLE 3 The Optinal Options for Modulus 7

Measure of Performance	N	Wb	Hb	W	H	A	T	A.T
A	1	15	10	24.94	20.49	511.3	15	7669.501
T	3	27	8	74.25	64.0	4752.0	9	42768
A.T	1	12	13	22.2149	23.111	513.4104		667.4334

Table 3 shows the optimal options according to the selected measure of performance. Figure 3 shows these optimal solutions.

3. Calculate the H_3, W_3 using algorithm 1 and selecting H_3 as close to $(H_1 + H_2 + 1)$ to save in area as it was shown in Fig. 3, the constraint here is that the corresponding time T_3 should be within a certain percentage with respect to the optimal time.

4. Calculate the overall dimensions of the module

$$H = \text{Max} \{ (H_1 + H_2 + 1), H_3 \} \quad (9)$$

$$W = W_2 + (1 + \lceil \log(m_1 - 1) \rceil + \lceil \log (m_2 - 1) \rceil) + W_3 \quad (10)$$

$$(W_2 \geq W_1)$$

$$T = T_1 + T_{ch} + T_3 \quad (11)$$

$$(T_1 \geq T_2)$$

If $m_1 = m_2 = m$, the best results are achieved when:

$$H = \text{Max} \{ (2H + 1), H_3 \}$$

$$W = W_1 + 2 \lceil \log (m - 1) \rceil + W_3$$

5.2. Fork Module

Fig. 5 shows an F-module, it has the same topology as J-module, but the data flow is in the reverse direction. To calculate the module's parameters, a procedure similar to the J-module is followed:

1. Calculate H_1, W_1 for Table 1 by algorithm 1 and selecting H_1 larger than W_1 within a certain limits for the corresponding area and time-delay.

2. Calculate the channel parameters:

$$W_{ch} = \lceil \log (m_1 - 1) \rceil + 1$$

$$H_{ch} = 3 \lceil \log (m_1 - 1) \rceil - 1$$

$$T_{ch} = \frac{1}{4} (3 \lceil \log (m_1 - 1) \rceil) + 1$$

4. Calculate the overall dimensions of the module as in J-module.

5.3. Joint-Fork Module

Fig. 6 shows a Joint-Fork module. This module combines the features of both J and F modules topologically and functionally. To calculate the module's parameters, we follow the same procedures developed before:

1. Tables 1, 2, and 3 parameters are calculated as in J-module.

2. Tables 4 and 5 parameters are calculated as in F-module.

3. Calculate the overall dimensions of the module:

$$H = \text{Max} \{ (H_1 + H_2 + 1), H_{ch_1}, H_{ch_2}, H_3, (H_4 + H_5 + 1) \} \quad (12)$$

$$W = W_1 + W_{ch_1} + W_3 + W_{ch_2} + W_4 \quad (13)$$

$$W_1 \geq W_2 \quad \text{and} \quad W_4 \geq W_5$$

$$T = T_1 + T_{ch_1} + T_3 + T_{ch_2} + T_2 \quad (14)$$

6. APPLICATIONS

The modules discussed in the previous section are able to be used as building blocks for many different DSP operations. In order to keep this paper brief, we present one typical use in the following sub-section. Other application can be found in [12].

6.1. NTT Multiplication

In [13] a scheme has been proposed to implement Number Theoretic Transforms (NTT's) using arrays of look-up tables. The transforms are computed in a ring which is isomorphic to a direct sum of Galois fields where the multiplication is computed modulo a prime number. That scheme is based on using the isomorphism between a multiplicative group g having elements $\{g_n\} = \{1, 2, 3, .., m_i - 1\}$ with multiplication modulo m_i, and the additive group K having elements $\{K_n\} = \{0, 1, 2,$

..., $m_i - 2\}$ with addition modulo $m_i - 1$, m_i being restricted to primes.

For the large moduli a submodular approach has been considered, where the modulus is decomposed into two relatively prime moduli, and the addition is carried out within these two moduli system. The final result is reconstructed using a look-up table. Fig. 7 illustrates the procedure with a modulo 19 multiplier.

The developed multiplication algorithm can be implemented by the proposed J-module in one chip as shown in Fig. 8. Tables I, II, III of Fig. 7 are combined in Table (1) of 361 x 3 = 1083 bits, also, Tables IV, V, VI are combined in Table (2) of 1083 bits. This composition cannot be done in an economical way using commercially available LSI modules. Table (3) is 42 x 5 = 210 bits. The dimensions of the module are 94.2 and 76.4 (unit length) for H and W, respectively. The time delay is 33 + 13 + 12 = 67 unit times.

7. CONCLUSIONS

The paper has presented a model for the VLSI layout of RNS arithmetic modules using NMOS technology. The modules are based on interconnected ROM's and the nucleus of the paper has concerned itself with a memory model for allowing optimization of a look-up table layout. The memory model proves flexible, in that it allows dimensions other than power of 2, which proves useful for optimal RNS look-up table storage.

The model can also be used, in conjunction with multi-look-up table modules, to efficiently layout useful RNS processing elements. Algorithms have been presented to perform the optimization, and an example illustrates the use of one of the modules in a finite field multiplier using a large prime modulus.

REFERENCES

[1] N. Ohmada et al, "LSI's for Digital Signal Processing", IEEE Journal Solid-State Circuits, Vol. SC-14, No.2, 1979.

[2] J.S. Thompson and J.R. Boddie, "An LSI Digital Signal Processor", ICASSP 80.

[3] J.L. Buie and T.A. Zimmerman, "Very Large Scale Integrated Circuits for Digital Signal Processing", Circuits and Systems, pp.2-8, Feb.1977 and pp.2-7, April 1977.

[4] S.S. Patil and T. Welch, "An Approach to Using VLSI in Digital Systems", Proc. 5th Symp. on Computer Architecture, pp.139-143, 1978.

[5] W.K. Jenkins and B.J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters", IEEE Trans. Circuits and Systems, Vol.CAS-24, pp.191-201, April 1977.

[6] M.A. Soderstrand, "A High Speed Low-cost Recursive Filter Using Residue Arithmetic", Proc. IEEE, Vol. 65, July 1977.

[7] G.A. Jullien, "Residue Number Scaling and Other Operations Using ROM Arrays", IEEE Trans. Comput., Vol.C-27, pp.325-337, April 1978.

[8] M. Bayoumi, et al, "A VLSI Implementation of Memory Intensive Residue Number System Architecture", 20th Annual Allerton Conf., Oct. 6-8, 1982.

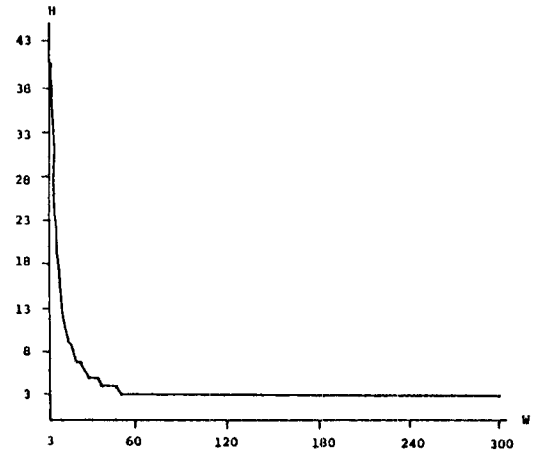
[9] C.A. Mead and M. Remi, "Cost and Performance of VLSI Computing Structures", IEEE Trans. on Electron Devices, ED-26, pp.533-540, April 1979.

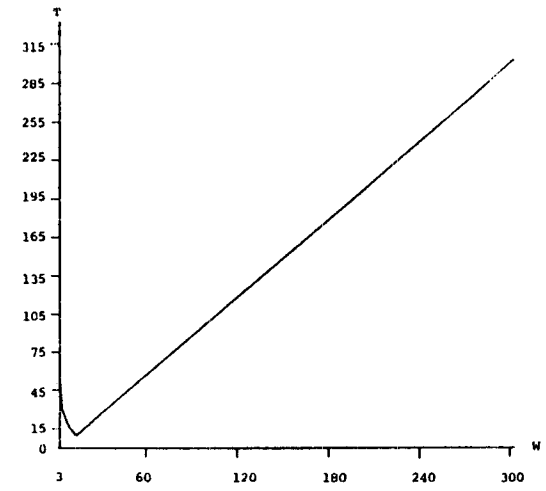
[10] C.A. Mead and L.A. Conway, 'Introduction to VLSI Systems', Addison Wesley, 1979.

[11] A.M. Mohsen and C.A. Mead, "Delay-Time Optimization for Driving and Sensing of Signals on High-Capacitance Paths of VLSI Systems", IEEE Trans. on Electron Devices, ED-26, pp.542-548, April 1979.

[12] M. Bayoumi and G.A. Jullien, "A VLSI Modular Implementation of RNS Systems", submitted for publication.

[13] G.A. Jullien, "Implementation of Multiplication, Modulo, a Prime Number, With Applications to Number Theoretic Transforms", IEEE Trans. Comput., Vol.C-29, pp.899-905, October 1980.





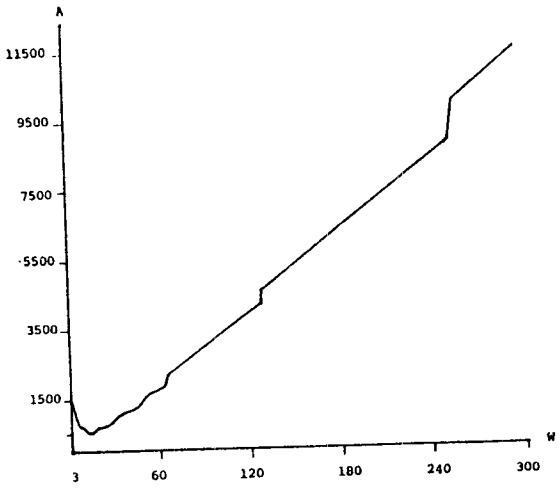


Fig. 1

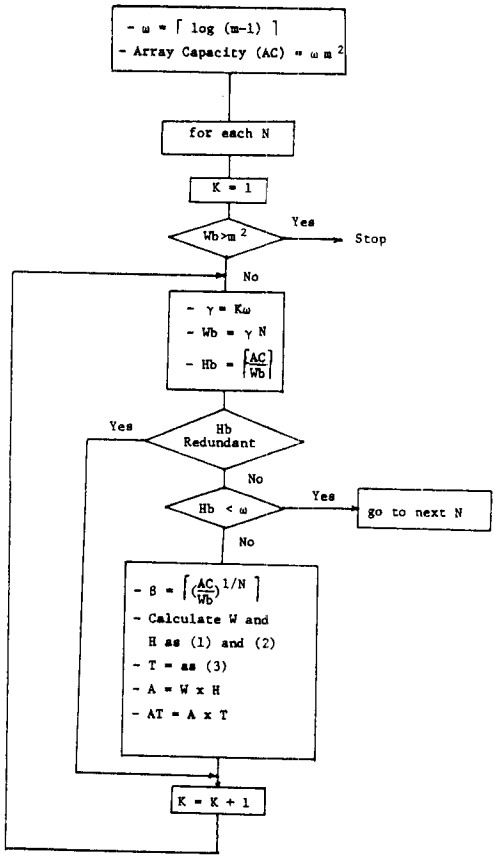


Fig. 2. Algorithm 1

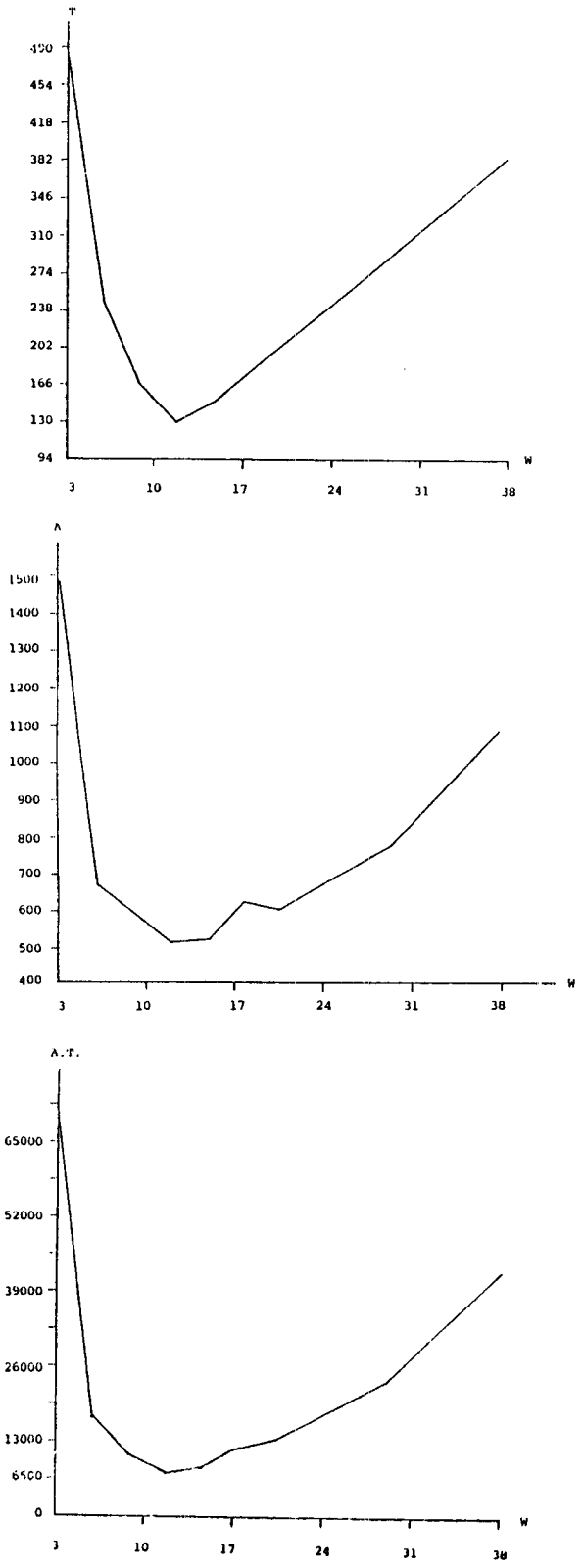
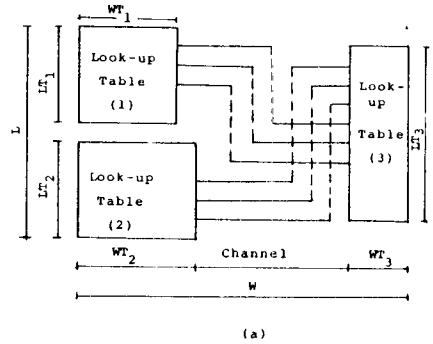


Fig. 3.



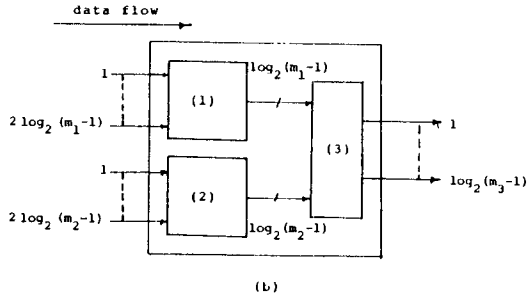
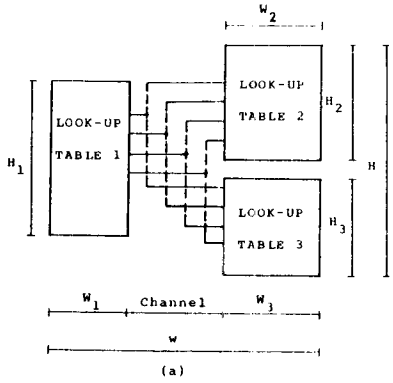


Fig. 4. A J-module
(a) Layout
(b) Input/Output



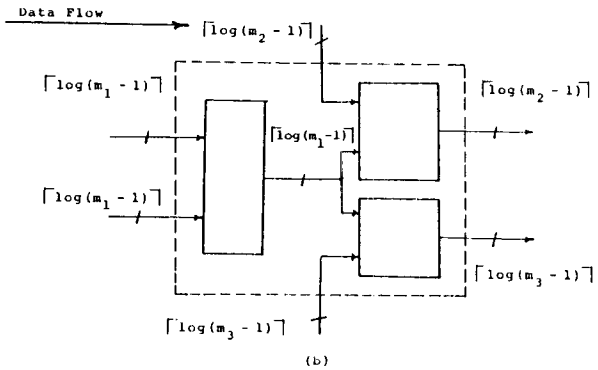


Fig. 5 An F-module

(a) Layout
(b) Input/Output

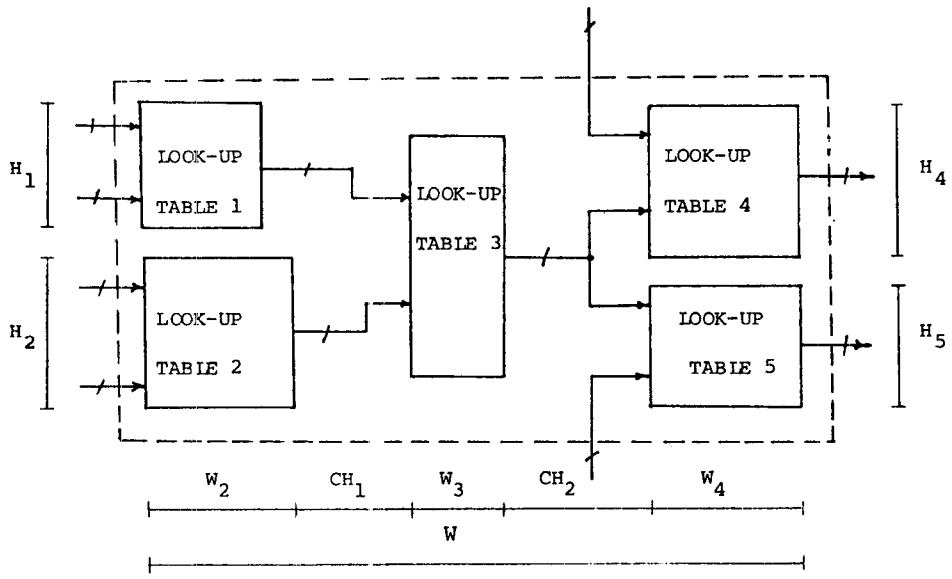


FIG. 6 The Layout of a JF-Module

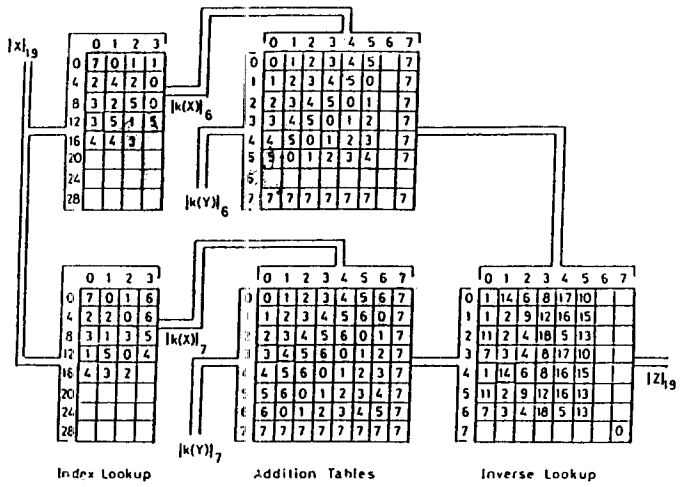
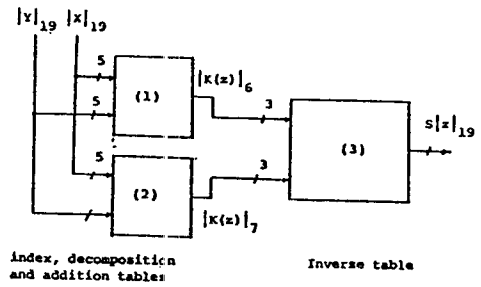
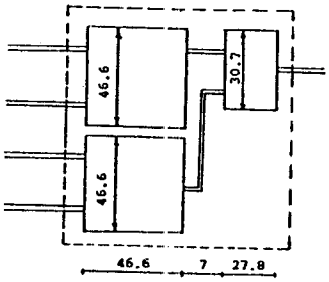


Fig. 7 A modulo 19 ROM array multiplier



(a)



(b)

Fig. 8 A modulo 19 multiplier

(a) Functional Diagram

(b) Floor Plan