IMPLEMENTATION OF THE SINGLE MODULUS COMPLEX ALU

Rom-Shen Kao
Fred J. Taylor

University of Florida

ABSTRACT

Recently the complex residue number system, or RNS, has been a subject of intense study. One special embodiment of this theory is the single modulus complex RNS processor which suggests both implementation and performance advantages. In this paper these conjectures are tested in the context of a CMOS gate array design and are found to be valid.

This work was supported under an ARO grant.

I. INTRODUCTION

Over the past several years there has appeared numerous articles advocating the use of the quadratic residue number system or QRNS, for performing complex algebraic operations [1-6]. As a subset of the more general theory of the residue number system (RNS), it has been shown to offer low multiplication complexity but, like other RNS tools, suffers from an inability to efficiently service scaling, magnitude comparison, or number system conversion calls. In an attempt to overcome this objectionable feature, the single modulus QRNS, or SM-QRNS was proposed [6]. Based on Fermat primes $p = 2^n + 1$ for $n = 2^m$, the SM-QRNS was shown to offer several advantages over other QRNS embodiments. They were:

i compared to the general QRNS
- scaling greatly simplified
- magnitude comparison greatly simplified
- sign detection trivial
- capable of assimilating existing high performance radix-2 hardware into its design

ii compared to the RNS and conventional binary weighted systems
- reduced computational complexity
- reduced computational latency

II. THE SM-QRNS

Let $Z_p[i] = \{a+ib \mid a,b \in Z_p;\ i = \mathrm{SQRT}(-1)\}$. The ring $Z_p[i]$ is called the ring of Gaussian integers modulo p. A complex CRNS number of the form a + ib evaluates to i if a = 0 and b = 1. If $i \notin Z_p$, then it is considered to be **imaginary.** That is, there is no $i \in Z_p$ such that $i^2 = -1 \bmod p$ and i is historically called a nonquadratic root and -1 is a nonquadratic residue. However, if $i \in Z_p$, then i is a quadratic root and a + ib is real. For example, if p = 17 and $i = 4 \in Z_p$, then $i^2 = 16 \equiv -1 \bmod 17$. For such moduli the concept of an imaginary operation is void. Now define $Z_{(p,p)} = Z_p \times Z_p$ and $(a, b), (c, d) \in Z_{(p,p)}$ with rules of composition given by:

$$\begin{aligned} &\text{addition:} \\ &(a,b) + (c,d) = ((a + c) \bmod p,\ (b + d) \bmod p) \\ &\text{multiplication:} \\ &(a, b)(c, d) = (ac \bmod p,\ bd \bmod p) \end{aligned} \tag{1}$$

Even though addition requires two (2) real adds (as in the CRNS), multiplication requires but two (2) real multiplies (versus 4 products and 2 adds in the CRNS)! The isomorphism between $Z_p[i]$ and $Z(p,p)$ has been well studied and for p a Gaussian prime satisfying $p = 4k + 1$ (in fact, it can be a product of Gaussian primes). The isomorphism given by ϕ, maps a complex number $c = a + ib$ ($i = \mathrm{SQRT}(-1)$) into a two tuple (z, z^*) as follows:

$$\begin{aligned} &(a, b) \xleftrightarrow{\phi} (z, z^*) \\ &z = (a + jb) \bmod p; \\ &z^* = (a - jb) \bmod p; \\ &j^2 \equiv -1 \bmod p;\ j \in Z_p \\ &a = 2^{-1}(z + z^*) \bmod p; \\ &b = (2j)^{-1}(z - z^*) \bmod p \end{aligned} \tag{2}$$

In (2), j denotes a quadratic root. In fact there are two such roots (say j_1 and j_2) which are both additive and multiplicative inverses of each other.

A specific choice of Gaussian primes are those which are also Fermat primes of the form $p = 2^n + 1$, $n = 2^m$, n = 2, 4, 8, 16. For the case where n = 32, p is a composite Gaussian prime and therefore also an admissable modulus. As such, the critical parameters found in equation 2 take the form:

$$\begin{aligned} &2^{-1} = 2^{n-1} + 1\ ;\ \ j = 2^{n/2}; \\ &j^{-1} = 2^n + 1 - 2^{n/2} \end{aligned} \tag{3}$$

which can be seen to be the essentially radix 2 scaling operations. Based on this result, a SM-QRNS of the type suggested in Figure 1 can be designed. The functional elements will be summarized in the next section.

III. SM-QRNS ARCHITECTURE

The SM-QRNS unit suggested in Figure 1 consists of the following function modules.

i) Negator (see Figure 2): Note, $-\underline{x} = (p - |x|) \bmod p = 2^n + 2 + \bar{x}) \bmod p$ where $\bar{x}$ denotes bitwise complement of x. Then if $x = \sum a_i 2^i$ it follows that $-x = \sum b_i 2^i$ where:

$$b_0 = \bar{a}_0$$

$$b_1 = \bar{a}_1$$

$$b_2 = \bar{a}_1 a_2 + a_1 \bar{a}_2$$

$$\vdots$$

$$b_n = \bar{a}_1 \bar{a}_2 \cdots a_n + (a_1 + a_2 + \cdots + a_{n-1})\bar{a}_n$$

ii) Modulo p Adder (see Figure 3): The mod p adder consists of an n-bit fast carry-lookahead adder, a modulo p mapping unit, and control logic (compare network). The mapping unit is designed to transfer $S = A + B$ into $V = (A + B) \bmod p$ or $V = S - p$ if $S \geq p$, using the following procedure [7]:

a. Starting with the LSB of S, complement all "0's" up to the first encountered "1."
b. Complement the first encountered "1."
c. Leave all other bits of greater significancy unchanged.

More specifically,

$$v_0 = \bar{s}_0$$

$$v_1 = s_0 s_1 + \bar{s}_0 \bar{s}_1$$

$$v_2 = \bar{s}_0 \bar{s}_1 \bar{s}_2 + (s_0 + s_1)s_2$$

$$\vdots$$

$$v_{n-1} = \bar{s}_0 \bar{s}_1 \cdots \bar{s}_{n-1} + (s_0 + s_1 + \cdots + s_{n-2})s_{n-1}$$

$$v_n = 0$$

iii) Modulo p Multiplier (see Figure 4): The modulo p multiplier consists of an unsigned multiplier and a network to process some exceptions:

a. If both inputs equal 2^n, then set the product to 1.
b. If only one input equals 2^n (say A) then set the product to -B

IV. SM-QRNS SUPPORT

The SM-QRNS architecture displayed in Figures 1 through 4 consists of a number of common macrocells called modules. They are summarized below and shown in Figure 5.

a. Scaling by j (eq. 3)
$jX \bmod p = 2^{n/2}X_{LO} - X_{HI}$,
$X = 2^{n/2}X_{HI} + X_{LO}$

b. Scaling by 2^{-1} (eq. 3)
$2^{-1}X \bmod p = (2^{n-1} + 1)X_{LO} + X_{HI}$;
$x = 2X_{HI} + X_{LO}$

c. Scaling by $(2j)^{-1}$ (eq. 3)
$(2j)^{-1}X \bmod p = X_{HI} - 2^{(n/2-1)}X_{LO}$;
$X = 2^{(n/2+1)}X_{HI} + X_{LO}$

V. SM-QRNS GATE ARRAY DESIGN

The SM-QRNS architecture reported in the previous section were tested using TEGAS (TEst Generation And Simulation language) and the IGC20000 Macro Library which are parts of the GE Semiconductor Gate Array design tool kit. Each array cell consists of three CMOS devices using 2 micron gate-length and the array itself may consist of up to 54,000 internal transistors. With these building blocks, the following modules were designed:

NEG:	Negates modulo p	MUTT:	Product modulo p
SUM:	Carry lookahead adder	JX:	scale by j
MDL:	Sum modulo p	ITWO:	scale by 2^{-1}
MUL:	Carry-save unsigned multiplier	ITWOJ:	scale by $(2j)^{-1}$

For expository reasons, the MDL module will be developed more fully. The MDL accepts data from the adder and modifies the received data modulo p if $S \geq p$ or passes it unaltered otherwise. A section of this logic is reported in Figure 6. Such data is used to define the gate array.

VI. ARRAY ANALYSIS

The macros used to configure the SM-QRNS are 2, 3, 4, and 5 input NAND gates, 2 input XOR and XNOR gates, 2, 3, and 4 input NOR gates, 2-2 and 2-2-2 AND-OR- inverters. The NEG, SUM, MDL, MUL, MUTT, JX, ITWO, and ITWOJ units were designed and integrated into a SM-QRNS machine. An eight bit design, for example, consists of 4187 1/3 array cells where each array cell consisted of 3 p-n channel transistors pairs.

The main purpose for this study was to develop a peformance database for SM-QRNS and conventional CRNS designs. To establish a common denominator for comparison, the QRNS and CRNS units are configured so as to have a similar hardware budget with both operating dual modulo p multipliers (see Figure 1). The basic database used to support the comparative analysis is reported in Table 1.

Assume T_c is the carry propagation time and T_s is the sum propagation time for a one-bit adder. The timing analysis for the module MUL unit, in general case is

$$T_d = \begin{cases} (n-1)T_s + (n-1)T_c & \text{if } nT_c < (n-1)T_s \\ (2^n-2)T_c & \text{if } nT_c > (n-1)T_s \end{cases}$$

where n is the bit number of the multiplicands. Based on the equivalent hardware budget assumption, the timing analysis for the QRNS and CRNS systems are stated as follows:

a. Because of the number of gate levels are the same for the case where n = 8 to 16, assume that the gate delay time for the non-wordlength dependent modules (i.e., non-multiplier) is fixed.

b. Assume $T_c = T_s$, the delay time T_m for the single modulus multiplier MUTT is
$T_m = (2n - 2)T_c + 230ns$

c. The delay time for the QRNS system under worse case is
$T_q = T_m$ + (delay time of non-processing units)

$= (2n - 2)T_c + 230ns + 650ns$

$= 2nT_c - 2T_c + 880ns$

while the delay time for CRNS system under worse case is
$T_x = 2T_m$ + (delay time of non-multiplier units)

$= 2((2n - 2)T_c + 230ns) + 200ns$

$= 4nT_c - 4T_c + 660ns$

The "break even" wordlength is given by
$4nT_c - 4T_c + 660ns = 2nT_c - 2T_c + 880ns$
$2nT_c - 2T_c = 220ns$

From the performed simulation, T_c approximates 15ns. Therefore, the equation shown above becomes reduced to 30n = 190 or n = 6.3.

Based on the timing studies and data reported in Table 1, the performance of a CRNS and SM-QRNS can be compared. The data represents temporal bounds on the performance of each individual module. For both the SM-QRNS and CRNS it can be seen that the principal bottleneck is at the multiplier level. The table also reports summary data for the integrated SM-QRNS and CRNS designs. This data was produced using TEGAS applied to a complete design rather than simply summing the individual module delays (in accordance to Figure 1). The result is a superior SM-QRNS design.

V. SUMMARY AND CONCLUSIONS

The principal objective of this work was to test the recently published conjecture that an integrated QRNS based unit could outperform an CRNS counterpart. Based on the analysis provided, using relatively slow gate array technology, the conjecture has been verified. While the results will vary somewhat as a function of technology and macro-cell differences, the reported results do represent an unbiased exploration of SM-QRNS question using accepted analysis tools. The foregone analysis indicates that in order to increase the potential speed advantage of SM architecture, the non-wordlength dependent overhead factors (i.e. scaling, negation and mod p) should be reduced to a minimum. It is noted that using gate-level constructs to implement these modules will present penalties. For example, moving from one fan-out to three could easily add another 1.06ns delay for the ND5 macro (5-input NAND gate) under 27°C and 5 V operating conditions. Instead, it may be advantageous to use a high-performance commerically available multiplier chip (used as an attached coprocessor) or high speed semiconductor table lookup (for n < 12 bits). For example, a 45ns 16 x 16 multiplier chip could be interfaced to a SM-QRNS controller/engine. It would accept operands from the engine, compute the binary weighted product, and return the results to the engine for modulo p reduction. The table-lookup method would also be a suitable forum in which to develop a faster architecture. For example, using high speed memory device (e.g., 40ns, 64kx1 static RAM chips) to implement the mapping logic in a n = 16 QRNS system, the following latencies have been determined:

	Table-Lookup	Gate Array
NEG	40ns	60ns
SUM + MDL	250ns	250ns
JX	40ns	130ns
ITWO	40ns	100ns
ITWOJ	40ns	120ns
MUTT	$(2n-2)T_c$ + 230ns	$(2n-2)T_c$ + 230ns

Using the fast lookup method, the non-wordlength overhead bias to T_q can be decreased from 880ns to 670ns. The "break even" wordlength then becomes n = 1.3.

Some large and varied macrocell libraries, composed of input more pin gates, multiplexers, register files, and multipliers, are available in the market place. These predefined LSI/VLSI functions are optimized such as the C20000 CMOS gate array from Fujitsu Microelectronics mixes high-speed static memory with random logic. This configuration, which consists of 15,000 uncommitted gates with 6K or 12K bits of RAM, provides both small lookup table and discrete logic design capabilities. The bipolar array is another choice for modular design although it has fewer gates within a single chip (up to 8,000 gates versus 20,000 for CMOS counterparts). Higher speed with fewer problems in driving large fan-out (as many as eight loads) can be found in bipolar arrays which have advantages in designing high throughput signal processing systems.

REFERENCES

1. M.C. Vanwormhoudt, "Structural properties of complex residue rings applied to number theoretic Fourier transforms," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-26, Feb. 1978.

2. S.H. Leung, "Application of residue number system to complex digital filters," in Proc. 15th Asilomar Conf. Circuits Syst. Comput., Pacific Grove, CA, Nov. 1981, pp. 70-74.

3. J.B. Krogmeier and W.K. Jenkins, "Error detection and correction in quadratic residue number system," in Proc. IEEE 26th Midwest Symp. Circuits Syst., Pueblo, MX, Aug. 1983.

4. M.A. Soderstrand and G.D. Poe, "Applications of a quadratic-like complex residue arithmetic system to ultrasonics," in Proc. 1984 ICASSP, San Diego, CA, Mar. 1984.

5. F. Taylor et al., "A radix-4 FFT using complex residue arithmetic," IEEE Trans. Comput., June 1985.

6. F Taylor, "Single modulus complex ALU," IEEE Trans. on ASSP, Oct. 1985, pp. 1302-1315.

7. F. Taylor, "Residue arithmetic: A tutorial with examples," IEEE Computer Magazine, May 1984.

Acknowledgement: To JoEllen Wilbur for her assistance.

This work was supported under an ARO grant.

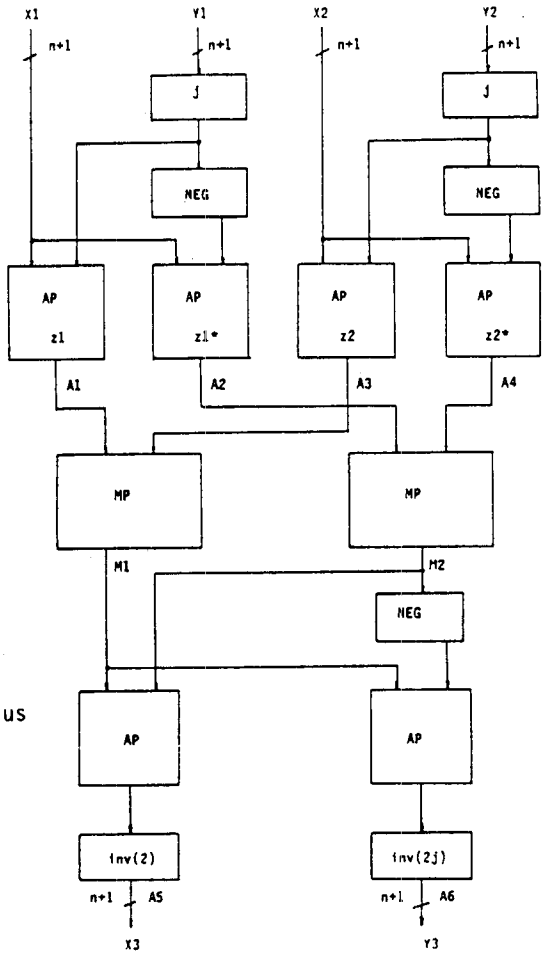


Figure 1 Functional block of the Single Modulus QRNS (SM-QRNS) unit

Table I
Listing of propagation delay of the CRNS, QRNS system and their modules

Modulus $P = 2^n + 1$ Delay time	n = 4		n = 8		n = 16	
Module	min	max	min	max	min	max
NEG	30ns	50ns	40ns	60ns	40ns	60ns
SUM	60ns	70ns	120ns	130ns	120ns	130ns
MDL	50ns	80ns	70ns	120ns	70ns	120ns
MUL	-	80ns	230ns	240ns	540ns	560ns
MUTT	-	250ns	300ns	470ns	620ns	790ns
JX	70ns	110ns	90ns	130ns	90ns	130ns
ITWO	50ns	60ns	80ns	100ns	80ns	100ns
ITWOJ	70ns	100ns	90ns	120ns	90ns	120ns
QRNS	480ns	740ns	900ns	1100ns	1220ns	1420ns
CRNS	230ns	560ns	910ns	1200ns	1550ns	1840ns

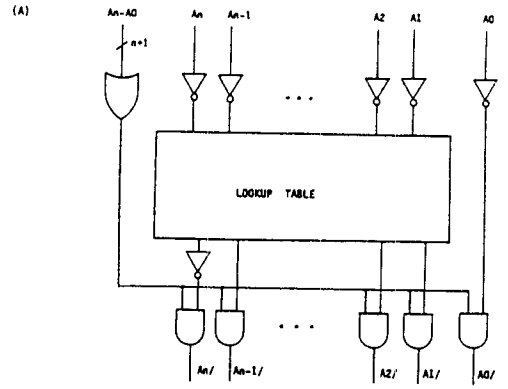
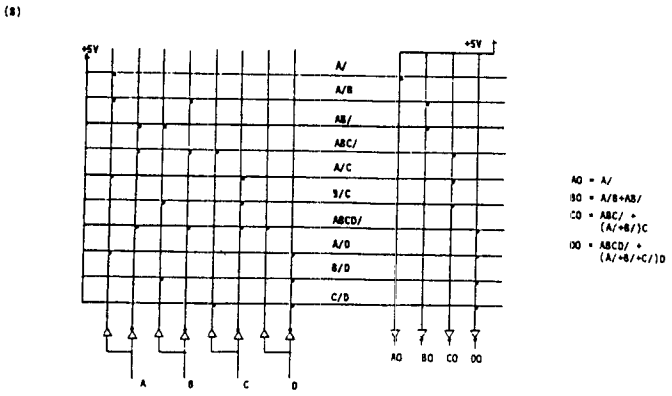


Figure 2 (A) Circuit diagram of the NEGATOR module
(B) PLA implementation of a 4-bit lookup table for the NEGATOR module



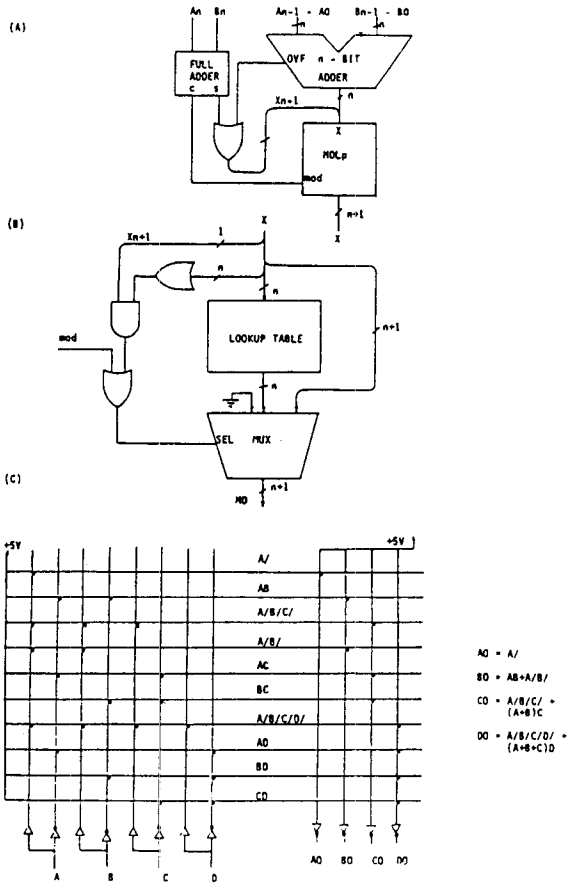


Figure 3 (A) Modulo P Adder
(B) MDL_p unit implemented by using the table lookup method
(C) PLA implementation of a 4-bit lookup table for the MDL_p

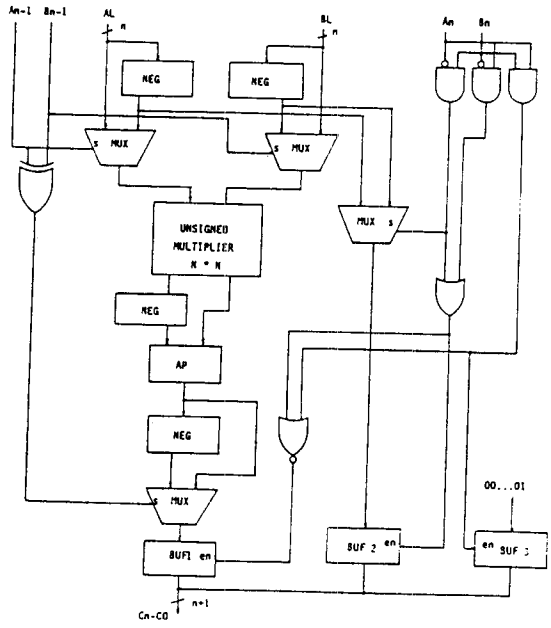


Figure 4 Modulo P Multiplier

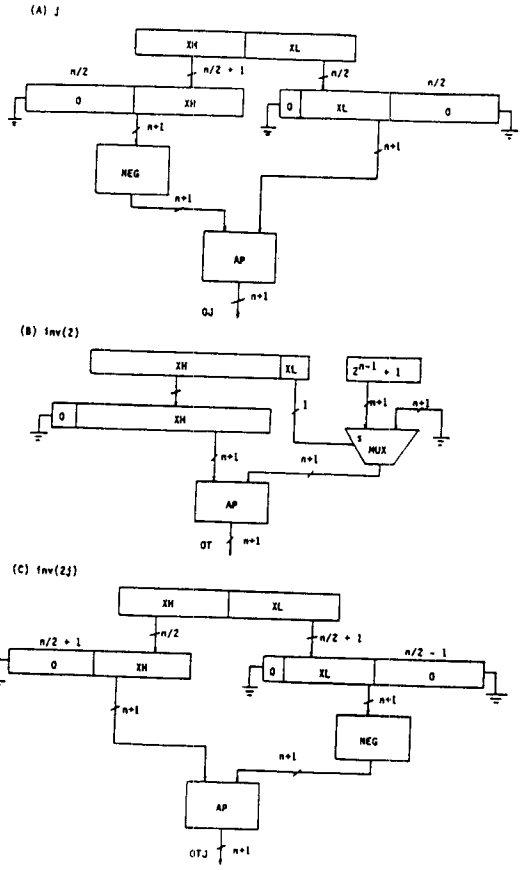


Figure 5 (A) Scaling module j
(B) Scaling module inv(2)
(C) Scaling module inv(2j)

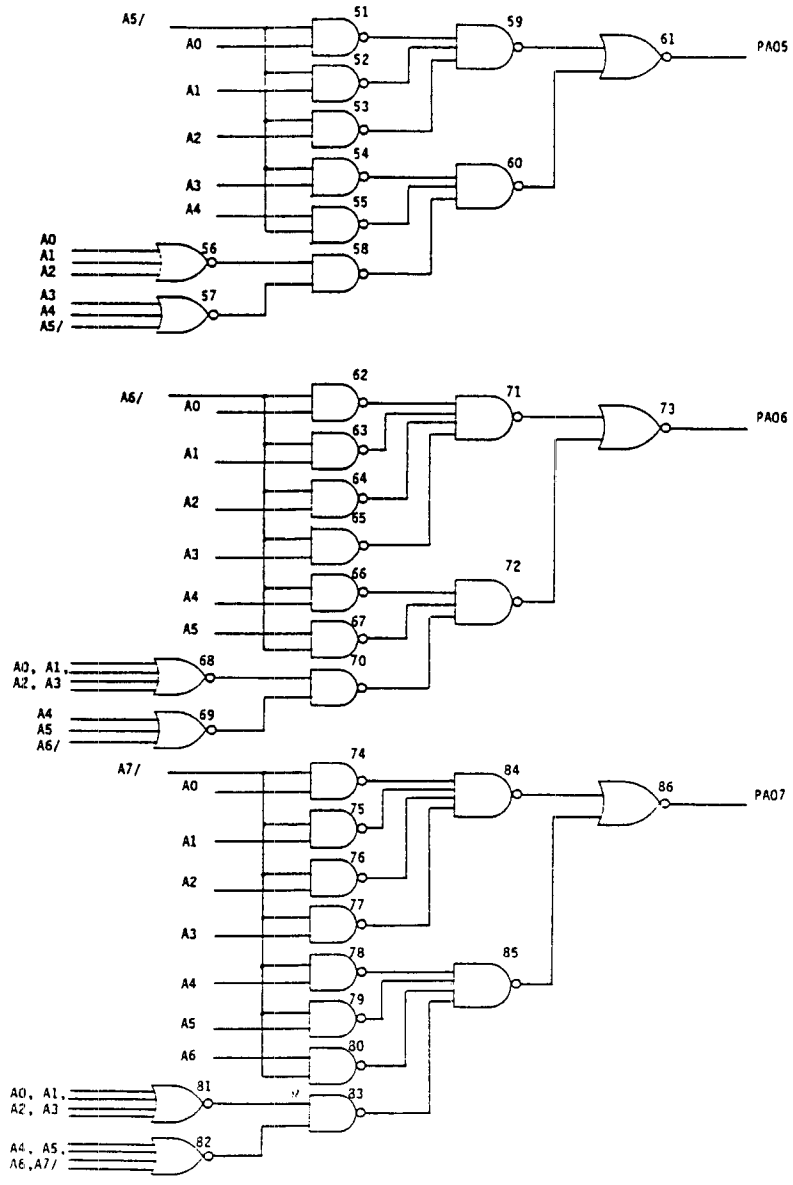


Figure 6 Circuit diagram of the MDL module (a submodule of Modulo P Adder)