VLSI DESIGN OF A MODULO-EXTRACTOR

R. Sivakumar, N. J. Dimopoulos, and K. F. Li

Department of Electrical & Computer Engineering
University of Victoria
Victoria, B.C, CANADA V8W 3P6

Abstract

In this paper, a VLSI design of a modulo-extractor based on the principles of Residue Arithmetic is discussed. A method for computing $\langle X\rangle_m$ for specific values of m is analyzed and the area-time complexity of the proposed structure is derived. The circuit has been implemented in 3μ CMOS3DLM technology and simulation results have yielded a propagation delay of less than 100ns.

1 Introduction

The rapid advancement in micro-electronics in recent years has opened new avenues in the field of Residue Number Systems (RNS). The RNS based processors which were so far considered impractical are now being investigated for implementation in VLSI [6]. Residue Number systems are characterized by a high degree of parallelism, regularity and modularity which can be exploited in VLSI to build high speed digital circuits and RNS processors [7]-[9]. These structures lend themselves to several applications in signal processing (digital filters, FFT, DFT, convolution), in the ALU of digital computers, in high speed arithmetic units such as fast adders, multipliers, random number generators [10]-[13].

Many real time systems which require high speed number crunching cannot afford to emulate in software. By migrating the design from software to hardware, system performance can be enhanced provided the problem is viable for hardware implementation. In the context of VLSI, the special purpose processors/digital circuits designed should have high throughput with reduced area-time complexity. In this paper, the hardware design of a modulo-extractor based on RNS for computing the function $X \bmod m$ for specific values of m is discussed. The paper is organized as follows. Section 2 introduces the basic definitions and terminology of residue arithmetic with respect to the application at hand. In section 3, the asymptotic area-time complexity of the proposed model is derived. Finally, the VLSI implementation of the modulo extractor and conclusion are discussed in the end.

2 Residue Arithmetic

Let X and m be any two integers with $m \geq 0$. Then we can write $X = mq + r$ where q is the quotient and r is the remainder such that $0 \leq r < m$. The remainder r, called the *residue* of X mod m is designated by $r = \langle X\rangle_m$. The solution to $\langle X\rangle_m$ is obtained from the remainder of the integer division X/m. However division is very expensive and does not lend itself to easy implementation. We are utilizing residue arithmetic to design a modulo extractor with reduced area-time complexity.

The modulo extractor discussed in this paper is capable of computing the moduli of the forms β^n,β^{n-1},β^{n+1}. In addition the moduli of other composite numbers which can be expressed as a product of prime factors can be determined through the application of the *Chinese Remainder Theorem* (CRT). In the following sections, the theory behind the use of residue arithmetic in the modulo-extractor and its formulation are discussed.

Let N be a number system defined in radix β. Then any number $X \epsilon N$ can be represented as an n-digit tuple $x_{n-1}, x_{n-2}, \ldots, x_1, x_0$ such that $X = \sum_{i=0}^{n-1} \beta^i x_i$.

2.1 Calculation of modulus β^k

For $k < n-1$, the number X and $\langle X\rangle_{\beta^k}$ are written as

$$\begin{aligned} X &= \sum_{i=0}^{k-1} \beta^i x_i + \beta^k[x_k + \beta x_{k+1} + \ldots + \beta^{n-1-k} x_{n-1}] \\ \langle X\rangle_{\beta^k} &= \langle \sum_{i=0}^{k-1} \beta^i x_i + \beta^k[x_k + \beta x_{k+1} + \ldots \beta^{n-1-k} x_{n-1}]\rangle_{\beta^k} \\ &= \sum_{i=0}^{k-1} \beta^i x_i \qquad (1) \end{aligned}$$

Therefore the number represented by the least significant k digits of the representation of X gives the modulus $\langle X\rangle_{\beta^k}$.

2.2 Calculation of modulus $\beta^k - 1$

Before proceeding further, we state the following relations which are important in this analysis.

$$\begin{aligned} \langle \beta^k\rangle_{\beta^k-1} &= \langle \beta^k - 1 + 1\rangle_{\beta^k-1} \\ &= \langle \langle \beta^k - 1\rangle_{\beta^k-1} + \langle 1\rangle_{\beta^k-1}\rangle_{\beta^k-1} = 1 \qquad (2) \end{aligned}$$

By a similar token we have

$$\langle \beta^{nk}\rangle_{\beta^k-1} = \langle [\beta^k]^n\rangle_{\beta^k-1} = \langle 1^n\rangle_{\beta^k-1} = 1 \qquad (3)$$

Let Y_0 denote the number represented by the k least significant digits. That is $Y_0 = \sum_{i=0}^{k-1} x_i\beta^i$. Y_1 is denoted by the next k significant bits given by $Y_1 = \sum_{i=0}^{k-1} x_{k+i}\beta^i$ and so on. Then

$$\begin{aligned} \langle X\rangle_{\beta^k-1} &= \langle Y_0 + \beta^k Y_1 + \beta^{2k} Y_2 + \ldots \beta^{(\rho_0-1)k} Y_{\rho_0-1}\rangle_{\beta^k-1} \\ &= \langle Y_0 + Y_1 + \ldots + Y_{\rho_0-1}\rangle_{\beta^k-1} \qquad (4) \end{aligned}$$

where $\rho_0 = \lceil \frac{n}{k} \rceil$[1]. The number ρ_0 represents the number of k-bit numbers to be added in the first step of the algorithm. After the sum $Z = \sum_{i=0}^{\rho_0-1} Y_i$ is determined in the first step, we apply the procedure recursively till the final result is less than β^k. The number of digits needed to represent Z is given by $\hat{Z} = \lceil \log_\beta Z \rceil$. Since $Z < \lceil \frac{n}{k} \rceil \beta^k$, the upper bound on the digit representation of Z is given by $\hat{Z}_{ub} = \left\lceil \log_\beta \lceil \frac{n}{k} \rceil + k \right\rceil$.

The problem therefore reduces to evaluating the much simpler $\langle Z\rangle_{\beta^k-1}$ due to a drastic reduction in its digit representation. The number of blocks ρ_1 required for addition in the second step is upper bound by

$$\rho_1 = \left\lceil \frac{\lceil \log_\beta \lceil \frac{n}{k} \rceil + k \rceil}{k} \right\rceil \qquad (5)$$

The procedure is repeated till ρ_i is equal to $k+1$ which implies that $\langle X\rangle_{2^k-1}$ is determined at that point. A decoding logic has to incorporated in the final stage in order to decipher the case $\langle \beta^k - 1\rangle_{\beta^k-1} = 0$.

[1] $\lceil x \rceil$: The smallest integer that is greater than or equal to x

A recurrence relation for the time complexity can be defined as follows.

$$T(n) = \begin{cases} T(\lceil \log\lceil \frac{n}{k} \rceil + k \rceil) + t(\lceil \frac{n}{k} \rceil) & \text{if } n > k+1 \\ 2t(2) + c & \text{if } n = k+1 \\ c & \text{if } n < k+1 \end{cases} \tag{6}$$

where $t(n)$ is the time taken to add n k-bit numbers. When $n = k + 1$, the number of levels of reduction needed is two at the most. The term c represents the constant time needed for decoding the *zero condition* given by $\langle \beta^k - 1 \rangle_{\beta^k - 1} = 0$.

We can summarize the above discussion to compute $\langle X \rangle_{\beta^k - 1}$ as follows.

Procedure: 2.2.1 *$\langle X \rangle_{\beta^k - 1}$ can be determined through the following procedure;*

1. *Represent the n-digit number as a weighted sum of powers of β^k as per eqn.(4);*
2. *Compute the sum of the $\lceil \frac{n}{k} \rceil$ k-digit factors;*
3. *Evaluate the modulus of the resultant sum represented by $\left\lceil \log_\beta \lceil \frac{n}{k} \rceil + k \right\rceil$ digits with respect to $\beta^k - 1$.*

In situations that warrant a simpler, modular and expandable structure from the design point of view, a variant of the recursive approach termed as the *bit-sliced* method can be used. In this method, the bit-slice s is chosen in such a manner that s divides n and the given number X can be partitioned into $\frac{n}{s}$ blocks such that the following relation is valid.

$$\begin{aligned} \langle X \rangle_m &= \langle Y_0 \pm \beta^s Y_1 \pm \ldots \pm \beta^{s(\lceil \frac{n}{s} \rceil - 1)} Y_{\lceil \frac{n}{s} \rceil} \rangle_m \\ &= \langle \langle Y_0 \rangle_m \pm \langle Y_1 \rangle_m \pm \ldots \pm \langle Y_{\lceil \frac{n}{s} \rceil} \rangle_m \rangle_m \end{aligned} \tag{7}$$

The residues of each of the $\frac{n}{s}$ sub-blocks are evaluated using the recursive approach outlined earlier and the outputs are passed to a cluster of residue adders. As shown in Fig. 1, the residues of the s-bit sub-blocks form the input to the residue adders which are arranged as a tree. Details on the structure of the residue adder is explained in the later section. This method is practical in cases where m is a small number.

2.3 Calculation of modulus $\beta^k + 1$

Making use of the properties that $-1 \equiv \beta^k \bmod \beta^k + 1$ and $-1^n \equiv \beta^{nk} \bmod \beta^k + 1$, we get

$$\begin{aligned} \langle X \rangle_{\beta^k+1} &= \langle Y_0 + \beta^k Y_1 + \beta^{2k} Y_2 + \ldots \beta^{\rho_0 k} \rangle_{\beta^k+1} \\ &= \langle Y_0 - Y_1 + Y_2 - Y_3 + \ldots \pm Y_{\rho_0} \rangle_{\beta^k+1} \end{aligned} \tag{8}$$

Evaluation of $\langle X \rangle_{\beta^k+1}$ is effectively reduced to addition and subtraction and the recursive method can be used to determine the result.

Having explained the principles for evaluating the modulus for a number system defined in radix-β, we consider the special case of the binary system for which $\beta = 2$ which is a tenable proposition for digital implementation. In the following sections, the application of the Chinese Remainder Theorem to the X mod m problem and the theory of Relational Calculus will be elucidated.

2.4 Calculation of modulus of composite numbers

The CRT provides an elegant method to evaluate $\langle X \rangle_M$ when M is a composite number [3, 4]. If M can be expressed as a product of mutually prime[2] numbers $m_1, m_2, \ldots, m_k$ for which the moduli are known, then X mod M can be generated in terms of the prime numbers. Let $M = \prod_{i=1}^{k} m_i$ and $\langle X \rangle_{m_i} = r_i$ where r_i is the residue of X with respect to m_i. Since the $m_i s$ are mutually prime, we can write $M_1 = M/m_1$, $M_2 = M/m_2, \ldots$, $M_k = M/m_k$ such that GCD $(M_i, m_i) = 1$. We can find numbers N_i such that $\langle N_i M_i \rangle_{m_i} = 1$ and

$$\langle X \rangle_M = \langle r_1 N_1 M_1 + r_2 N_2 M_2 + r_3 N_3 M_3 + \ldots + r_k N_k M_k \rangle_M \tag{9}$$

If we take the modulus of eqn.(9) with respect to m_1 then

$$\langle \langle X \rangle_M \rangle_{m_1} = \langle X \rangle_{m_1} = \langle r_1 N_1 M_1 \rangle_{m_1} \tag{10}$$

The other terms in equation(9) will be zero as they contain m_1. Since $\langle N_i M_i \rangle_{m_i} = 1$, we get $\langle X \rangle_{m_1} = \langle r_1 \rangle_{m_1}$. From the above relation, it can be concluded that $\langle X \rangle_{m_i} = r_i$. By determining the coefficients N_i and M_i, $\langle X \rangle_M$ can be computed in terms of moduli m_i. When the $m_i s$ are not mutually prime, the above procedure can be used to determine X mod $\bar{M}$ rather than X mod M and $\bar{M}$ is represented by the least common multiple of $m_1, m_2, \ldots m_k$.

The inherent disadvantage of mapping the CRT directly in hardware is the additional overhead incurred in the form of multipliers, decoders and residue converters. To circumvent this problem, a residue mapping function (RMF) given by $F(x) : P \rightarrow Q$ is defined as follows.

$$\begin{aligned} P &= \{\text{The set of all residue classes generated by } \langle X \rangle_M\} \\ Q &= \{n_1, n_2, \ldots, n_r \mid n_1, n_2, \ldots, n_k \text{ is a } k\text{-tuple string and} \\ &\qquad n_i = x \bmod m_i, \forall\, x\colon 0 \le x \le M-1,\ i = 1 \ldots k, \\ &\qquad m_1.m_2, \ldots m_k = M\} \\ F &= \{(x, n_1, n_2, \ldots, n_k) \mid x \epsilon P, n_1, n_2, \ldots, n_k \epsilon \text{ Q}\} \end{aligned} \tag{11}$$

It can be shown that the RMF is homomorphically related to the CRT. For the sake of brevity. the proof is not presented here. From the implementation and hardware complexity point of view, the RMF is a better candidate than eqn.(9) since the required boolean logic for computing the modulus of M can be derived easily from the mapping tables of the residues of its prime factors. The need for multipliers and other residue decoders can therefore be avoided in the RMF.

3 Area-time complexity

In this section, the modulo-extractor is evaluated as an asymptotic VLSI design, that is the order of complexity in terms of silicon area A and the response time T of the circuit. Expressions are derived for both the recursive and bit-sliced methods. To analyze the area-time complexity of the circuit, we consider the cases when $m = 2^k - 1, 2^k + 1$.

3.1 Recursive Method

3.1.1 Area

The recurrence relation for the recursive method given by eqn.(6) suggests that maximum complexity occurs when the addition of $(\lceil \frac{n}{k} \rceil)$ k-bit numbers is performed in the first stage of the algorithm. The subsequent steps are, by and large, less involved since the representative size of the number is reduced logarithmically given by eqn.(5). It can be stated that the area and time complexity of the circuit is therefore dominated by the addition in the first step.

Assuming that each k-bit adder has an area $\mathcal{O}(k \log k)$ [2], the number of adders needed in the first step of the algorithm is $\lceil \frac{n}{k} \rceil$. The area complexity of the circuit is given by

$$\begin{aligned} A_I &= k \log k \left\lceil \frac{n}{k} \right\rceil + k \log k \left\lceil \frac{\lceil \log\lceil \frac{n}{k} \rceil + k \rceil}{k} \right\rceil + \\ &\quad + \mathcal{O}(k \log k \log\log \left\lceil \frac{n}{k} \right\rceil) \end{aligned} \tag{12}$$

$$\begin{aligned} &\approx k \log k \left\lceil \frac{n}{k} \right\rceil + \log k \log \left\lceil \frac{n}{k} \right\rceil \\ &= \mathcal{O}(\left\lceil \frac{n}{k} \right\rceil) \qquad \text{if } n \gg k \end{aligned} \tag{13}$$

[2]Two numbers x,y are said to be mutually prime if x and y have no common divisors. GCD $(x, y) = 1$

3.1.2 Time

If the adders are arranged in the form of a tree the number of steps to add $\lceil \frac{n}{k} \rceil$ k-bit numbers is $\log\lceil \frac{n}{k} \rceil$. If each k-bit adder has a time complexity $\mathcal{O}(\log k)$ [2], then the total time complexity of the circuit is given by

$$\begin{aligned} T_I &= \log k \log \left\lceil \tfrac{n}{k} \right\rceil + \log k \log \left\lceil \tfrac{\lceil \log\lceil \frac{n}{k} \rceil + k \rceil}{k} \right\rceil + \\ &\quad + \mathcal{O}\left(\log k \log\log \left\lceil \tfrac{n}{k} \right\rceil\right) \qquad (14) \end{aligned}$$

When $n \gg k$, we have

$$T_I = \mathcal{O}\left(\log \left\lceil \tfrac{n}{k} \right\rceil\right) = \mathcal{O}(\rho_0) \qquad (15)$$

Hence the area-time complexity for the structure is obtained by combining eqns. (13) and (15) to get

$$AT_I = \mathcal{O}(\left\lceil \tfrac{n}{k} \right\rceil \log \left\lceil \tfrac{n}{k} \right\rceil) = \mathcal{O}(\rho_0 \log \rho_0) \qquad (16)$$

where $\rho_0 = \lceil \frac{n}{k} \rceil$

3.2 Bit-sliced Method

3.2.1 Area

As was explained earlier, the crux of the bit-sliced method lies in evaluating mod m for each bit-slice of length of s using the recursive method and the output is processed through the tree of residue adders as given in Fig.1. The residue adders are improvised versions of conventional adders in that they accept the residues from the previous blocks and compute the resultant modulus. In other

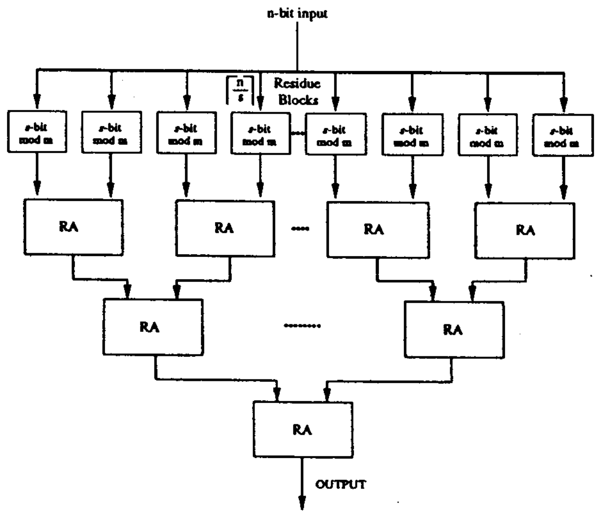

Figure 1: Hierarchical diagram of bit-sliced modulo extractor

words, if $\langle X_1 \rangle_m$ and $\langle X_2 \rangle_m$ are known, then the residue adder can determine $\langle \langle X_1 \rangle_m + \langle X_2 \rangle_m \rangle_m$. For example, when $m = 2^k - 1$, the residue adder determines the result according to the following conditions.

1. If $\langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1} < 2^k - 1$, then the solution is given by the result $\langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1}$.

2. If $\langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1} = 2^k - 1$, then we have a string of k 1's and the result is decoded to be zero since $\langle 2^k - 1 \rangle_{2^k-1} = 0$.

3. If $\langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1} > 2^k - 1$, then the addition will result in a $k+1$ bit number, then

$$\begin{aligned} \langle \langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle \rangle_{2^k-1} &= \langle 2^n + \langle \langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1} \rangle_{2^k} \rangle_{2^k-1} \\ &= 1 + \langle \langle X_1 \rangle_{2^k-1} + \langle X_2 \rangle_{2^k-1} \rangle_{2^k} \qquad (17) \end{aligned}$$

We get the required solution by adding the carry bit to the least significant k-bits after the first addition is performed. In effect, two levels of addition is performed. The residue adder (RA) is composed of two adders, the first one for adding the two k bit numbers and the second for absorbing the resultant carry, if any, to the previous sum. The adders can be constructed to have a area and time complexity of $\mathcal{O}(k \log k)$ and $\mathcal{O}(\log k)$ respectively while the decoding circuit for the zero condition can be taken to have constant delay. A similar structure is used for the case $m = 2^k + 1$ but with subtractors instead of adders.

The area of the circuit using the bit-sliced approach is composed of two parts, namely, the area occupied by the bit-sliced stages and the residue adder tree. Since there are $\frac{n}{s}$ mod m blocks, the area for the bit-sliced stage is given by

$$\begin{aligned} A_s &= \left(\frac{n}{s}\right) \left[k \log k \left\lceil \tfrac{s}{k} \right\rceil + k \log k \left\lceil \tfrac{\lceil \log\lceil \frac{s}{k} \rceil + k \rceil}{k} \right\rceil + \right. \\ &\quad \left. + \mathcal{O}(k \log k \log\log \left\lceil \tfrac{s}{k} \right\rceil)\right] \qquad (18) \\ \Rightarrow A_s &< \frac{n}{s}\left[\gamma k \log k \left\lceil \tfrac{s}{k} \right\rceil\right] < \gamma k \log k \frac{n}{s}\left(\frac{s}{k} + 1\right) \\ \Rightarrow A_s &< \gamma k \log k \left(\frac{n}{k} + \frac{n}{s}\right) < \gamma k \log k \left(\left\lceil \tfrac{n}{k} \right\rceil + \frac{n}{s}\right) \\ \Rightarrow A_s &< \mathcal{O}\left(\left\lceil \tfrac{n}{k} \right\rceil + \frac{n}{s}\right) \qquad (19) \end{aligned}$$

and γ is a constant.

The area for the residue adder tree is given by

$$A_r = \left(\frac{n}{s} - 1\right) k \log k \qquad (20)$$

When $n \gg k$, the total area (A_S) of the bit-sliced approach is obtained from eqns.(19) and (20) and on simplification we get

$$A_S = \mathcal{O}\left(\left\lceil \tfrac{n}{k} \right\rceil + \frac{n}{s}\right) = \mathcal{O}(\mathrm{MAX}(\rho_0, N)) \qquad (21)$$

where $N = \frac{n}{s}$. Thus the area of the bit-sliced approach is dominated by either ρ_0 or N which ever is greater of the two.

3.2.2 Time

The time complexity for the bit-sliced approach is dictated by the delay of the s-bit modulo blocks and the delay of the residue adder tree. The delay of the s-bit blocks, T_s follows eqn.(14) except that n is substituted by s. That is

$$\begin{aligned} T_s &= \log k \log \left\lceil \tfrac{s}{k} \right\rceil + \log k \log \left\lceil \tfrac{\lceil \log\lceil \frac{s}{k} \rceil + k \rceil}{k} \right\rceil + \\ &\quad + \mathcal{O}\left(\log k \log\log \left\lceil \tfrac{s}{k} \right\rceil\right) \qquad (22) \end{aligned}$$

The residue adder tree has a delay given by

$$T_r = \log k \log \frac{n}{s} \qquad (23)$$

Thus the time complexity, T_S, of the bit-sliced approach is given by combining eqns.(22) and (23) to get

$$\begin{aligned} T_S &= \log k \log \left\lceil \tfrac{n}{k} \right\rceil + \log k \log \left\lceil \tfrac{\lceil \log\lceil \frac{s}{k} \rceil + k \rceil}{k} \right\rceil \\ &\quad + \mathcal{O}\left(\log k \log\log \left\lceil \tfrac{s}{k} \right\rceil\right) \qquad (24) \end{aligned}$$

When $n \gg k$ and for small s, we have

$$T_S = \mathcal{O}\left(\log \left\lceil \tfrac{n}{k} \right\rceil\right) \qquad (25)$$

The above equation suggests that the time complexity of both methods considered in this analysis are comparable. If we take into account the second order terms, the bit-sliced approach will be faster than the normal recursive method depending on the ratio n/s,

The area-time complexity of the bit-sliced approach is given by

$$\begin{aligned} AT_B &= \mathcal{O}\left(\left\lceil \tfrac{n}{k} \right\rceil \log \left\lceil \tfrac{n}{k} \right\rceil, \frac{n}{s} \log \left\lceil \tfrac{n}{k} \right\rceil\right) \\ &= \mathcal{O}(\mathrm{MAX}(\rho_0 \log \rho_0, N \log \rho_0)) \qquad (26) \end{aligned}$$

Table 1: Chip statistics of the implemented 16-bit modulo extractor

16-bit Modulo Extractor	
Technology	3μ NTE CMOS3DLM
Area of the chip	4545.1x4545.1 $\mu^2 m$
Area of the core	3681.1x3681.1 $\mu^2 m$
No. of active pins	38
No. of standard cells	1392
No. of transistors	5156
Propagation delay (worst case)	95ns
Area*time	1.2873e09 ns$\mu^2 m$

4 Implementation of Modulo-Extractor

A 16-bit modulo-extractor with moduli $2, 3, \ldots, 10$ has been implemented in 3μ CMOS Double Layer Metal technology. The schematic capture of the circuit was made using the Cadence design tool. The statistics of the chip are given in Table 1. The VLSI layout of the modulo extractor is shown in Fig. (2).

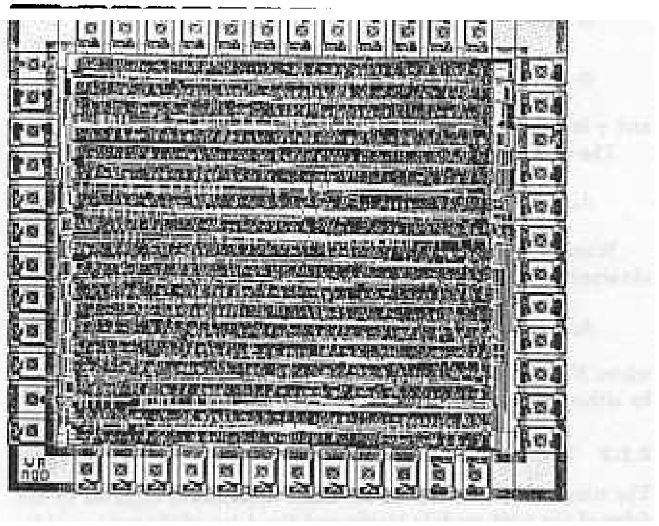

Figure 2: VLSI layout of implemented 16-bit modulo extractor (3μ CMOS3DLM)

5 Discussion

The proposed structure has significant advantages in terms of speed and area over the model proposed in [7]. It has been claimed that the VLSI structure for computing X mod m has a response time of less than 200 ns for 32-bit numbers. In our model, the propagation delay for 16-bit operands is less than 100ns for the 3μ CMOS3DLM technology. It has been shown that the increase in time complexity for the bit sliced method is proportional to $\log_2 \rho_0$ as the operand length increases. The implemented model is totally devoid of any multipliers and uses only high speed adders. This accounts for a drastic reduction in silicon real estate and propagation delay. The area-time complexity of the circuit is shown to be $\mathcal{O}(\rho_0 \log \rho_0)$. Besides the structure proposed is amenable to pipeline implementation. The demerits of our model is that it cannot be used for prime numbers which are not of the form 2^{k-1} or 2^{k+1} and the mapping of CRT through RMF for higher order moduli could be complicated.

6 Conclusion

The regular, modular structure of residue arithmetic coupled with the silicon revolution has given a vibrant dimension to the implementation of RNS. The technological advantages of VLSI has made RNS a viable proposition in terms of speed, cost, power dissipation and chip density. In this paper, a structure for the VLSI design of $\langle X \rangle_m$ operation built using residue arithmetic has been discussed. Expressions for the asymptotic area-time complexity of the model has been derived. Technology scaling should provide a quantum increase in performance and throughput. The circuit was simulated in the 1.2 μ CMOS technology and it yielded a worst case propagation delay of less than 20 ns. The above modulo-extractor has been used in the design of of a 16-bit mixed congruential random number generator. It also constitutes an integral part the Hypercycle routing engine [14] which has been implemented in 1.2μ CMOS4S technology.

Acknowledgement

The authors wish to thank the Canadian Microelectronics Corporation for its assistance in fabricating the above VLSI chip. They also acknowledge the support of the Natural Sciences and Engineering Research Council Canada and the Institute for Robotics and Intelligent Systems of the National Network of Centres of Excellence.

References

[1] C.A. Mead and L.A. Conway, *Introduction to VLSI systems*, Reading, MA: Addison-Wesley, 1980.

[2] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, Maryland, 1984.

[3] James H. McClellan, Charles M. Rader, *Number Theory in Digital signal processing*, Prentice Hall, 1979.

[4] Szabo N.S. and R.I. Tanaka, Residue Arithmetic and its Applications to Computer technology, McGraw Hill, 1967.

[5] Tremblay and Manohar, *Discrete Mathematical Structures with Application to Computer Science*, McGraw-Hill,1985.

[6] F.J. Taylor, "Residue Arithmetic: A tutorial with examples", *IEEE Computer Mag.*, pp. 50-62, May 1984.

[7] Giuseppe Alia, Enrico Martinelli, A VLSI Structure for X(mod m) Operation, *Journal for VLSI Signal Processing*, 1, pp. 257-264, 1990.

[8] M.A. Bayoumi, G.A. Jullien, W.C. Miller, A VLSI model for Residue Number System Architectures, *INTEGRATION, the VLSI journal* 2, pp. 191-211, 1984.

[9] F.J. Taylor, "A VLSI residue arithmetic multiplier", *IEEE Trans. Computers.*, vol C-31, 1982, pp. 540-546.

[10] 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, 1977, pp. 191-201.

[11] ——, "Modes for VLSI implementation of residue number system arithmetic modulus", *Proc. IEEE 6th Symp. on Comp. Arithmetic*, pp. 174-182, June 1983.

[12] M.A.Soderstrand, "A high speed low-cost recursive digital filtering using Residue Arithmetic", *Proc. IEEE*, vol. 65, July 1977, pp. 1065-1067.

[13] C.H. Huang, D.G.Patterson *et al*, "Implementation of a fast digital processor using the residue number system", *IEEE Trans. Circuits and Systems.*, vol CAS-28, 1981, pp.32-38.

[14] N.J.Dimopoulos, D. Radvan, R. Sivakumar, "Implementation of the Back-Track-to-the Origin-and-Retry routing of the Hypercycle based Interconnection Networks", *Proc. Canadian Conference on Elect. & Comp. Eng*, 1990,