نظام الأرقام المتبقي (RNS)

د. السيد عثمان شعرون[1] (*) د. الطاهر عبد الله قلعوز[2]

1 قسم الحاسوب، كلية التربية بالزاوية – جامعة الزاوية

2 قسم الهندسة الكهربائية والإلكترونية، كلية الهندسة صبراته، جامعة صبراته

الملخص:

في نظام الأرقام المتبقية ، يتم تقديم مجموعة من المعاملات المستقلة عن بعضها البعض يتم تمثيل العدد الصحيح بالمتبقي من كل معامل والعمليات الحسابية بناءً على المخلفات بشكل فردى ، العمليات الحسابية على أساس عدد المخلفات يمكن أن يتم تنفيذ النظام على وحدات مختلفة بشكل مستقل لتجنب الحمل بالإضافة إلى ذلك، الطرح والضرب ، والذى عادة ما يكون مضيعة للوقت ، ومع ذلك ، فإن المقارنة بعملية القسمة أكثر تعقيداً وحساب رقم الكسر غير كامل، ولذلك فإن نظام الأرقام المتبقية ليس شائعاً في أجهزة الكمبيوتر للأغراض العامة، بالرغم أنه مفيد للغاية لتطبيقات معالجة الإشارات الرقمية.

نتناول في هذا البحث التصميم ، المحاكاة وتنفيذ الميكرو كونترولر لبعض (نظام رقم البقايا) واللبنات الأساسية للتطبيقات في مجال معالجة الإشارات الرقمية ، لبنات البناء التي تم دراستها عبارة عن محمول ثنائي إلى محمول بقايا، بقايا إلى محمول ثنائي ،والبقايا المضاعفة ، كما تم إدخال الخوارزميات الجديدة في نظام البواقي.

(*) Email: asyed.sharoon@zu.edu.ly

نظام الأرقام المتبقي(RNS)

Dr.Assaid Othman Sharoun[1] , dr. Taher A . Galouz[2]

1 Department of Computer, College of Education- zawia, Zawia University

2 Department of Electrical and Electronic Eng, College of Engineering, Sabratha University

Abstract:

In the residue number system, a set of moduli which are independent of each other is given. An integer is represented by the residue of each modulus and the arithmetic operations are based on the residues individually. The arithmetic operations based on residue number system can be performed on various moduli independently to avoid the carry in addition, subtraction and multiplication, which is usually time consuming. However, the comparison and division are more complicated and the fraction number computation is immatured. Due to this, a residue number system is not yet popular in general-purpose computers, though it is extremely useful for digital-signal-processing applications. This thesis deals with the design, simulation and microcontroller implementation of some (residue number system) based building blocks for applications in the field of digital signal processing. The build¬ing blocks which have been studied are binary to residue converter, residue to binary converter, residue adder and residue multiplier. And the New algorithms were also introduced.

aims :

Why RNS?

- Residue Number System (RNS) has been found to be very efficient in doing certain arithmetic operations.
- Researchers have constructed ASIC architectures for DSP units, like FIR filters, based on RNS, which are faster and

consume less power than 2's complement binary number system.

- Practical results of RNS
- Jenkins & Leon, 1977 – Used RNS to design FIR filters
- Jenkins, Soderstrand, Jullien, Taylor, 1986 – Residue Number System Arithmetic: modern applications in Digital Signal Processing.
- H. T. Vergos, 2001 – Proposed 200 MHz RNS Core
  - Supposed to be used as a building block in ASIC design.
- Chaves & Sousa, 2003 – Design RISC DSP based on RNS
- Chaves & Sousa, 2007 – Moduli sets for balanced processing.

Advantages and disadvantages:

- Advantages
  - It is a "carry-free"system that performs addition, subtraction, and multiplication as **parallel operations.**
  - Additions, subtractions, multiplications can be done faster
- Disadvantages
  - Overhead of conversions.
  - Residue to Binary, based on Chinese Remainder Theorem, is especially expensive.
  - Magnitude comparison not possible.

Introduction to RNS:

- Non-positional (non-weighted) number system
- Characterized by set of relatively prime numbers $(P_1,P_2,\ldots,P_k)$ – **moduli (modulus)**
- A binary number N is represented as a *k*-tuple $(R_1,R_2,\ldots,R_k)$, where $R_i=N \bmod P_i$ - **residue**
- Any number in [0,M), $M= P_1 x P_2 x \ldots x P_k$ can be uniquely represented. M is called the **dynamic range.**

Definitions (1) :

- The RNS is defined in terms of a set of **relatively prime moduli**.
- If ***P*** denotes the moduli set, then

$$\boldsymbol{P} = \{p_1, p_2, \cdots, p_L\}$$
$$\mathrm{GCD}(p_i, p_j) = 1,\ for\ i \neq j$$

- The dynamic range ***M*** is

$$M = p_1 p_2 \cdots p_L$$

Definitions (2):

- Any integer in the residue class Z_M has a unique ***<u>L</u>*****<u>-tuple</u>**

$$X \xrightarrow{RNS} (X_1, X_2, \cdots, X_L)$$

<u>representation</u> given by

- **where Xi=X mod pi and is called the i-th residue**

Decimal to RNS conversion examples :

Example

$$\boldsymbol{P} = \{p_1, p_2\} = \{3,5\} \qquad \boldsymbol{M} = 3 \times 5 = 15$$

X	→	X_1	X_2		X	→	X_1	X_2		X	→	X_1	X_2
0	→	0	0		5	→	2	0		10	→	1	0
1	→	1	1		6	→	0	1		11	→	2	1
2	→	2	2		7	→	1	2		12	→	0	2
3	→	0	3		8	→	2	3		13	→	1	3
4	→	1	4		9	→	0	4		14	→	2	4

Decimal to RNS conversion examples (2)

$$\boldsymbol{P} = \{p_1, p_2, p_3\} = \{3,4,5\} \qquad \boldsymbol{M} = 3 \times 4 \times 5 = 60$$

X	→	X_1	X_2	X_3		X	→	X_1	X_2	X_3		X	→	X_1	X_2	X_3		X	→	X_1	X_2	X_3		X	→	X_1	X_2	X_3
0	→	0	0	0		12	→	0	0	2		24	→	0	0	4		36	→	0	0	1		48	→	0	0	3
1	→	1	1	1		13	→	1	1	3		25	→	1	1	0		37	→	1	1	2		49	→	1	1	4
2	→	2	2	2		14	→	2	2	4		26	→	2	2	1		38	→	2	2	3		50	→	2	2	0
3	→	0	3	3		15	→	0	3	0		27	→	0	3	2		39	→	0	3	4		51	→	0	3	1
4	→	1	0	4		16	→	1	0	1		28	→	1	0	3		40	→	1	0	0		52	→	1	0	2
5	→	2	1	0		17	→	2	1	2		29	→	2	1	4		41	→	2	1	1		53	→	2	1	3
6	→	0	2	1		18	→	0	2	3		30	→	0	2	0		42	→	0	2	2		54	→	0	2	4
7	→	1	3	2		19	→	1	3	4		31	→	1	3	1		43	→	1	3	3		55	→	1	3	0
8	→	2	0	3		20	→	2	0	0		32	→	2	0	2		44	→	2	0	4		56	→	2	0	1
9	→	0	1	4		21	→	0	1	1		33	→	0	1	3		45	→	0	1	0		57	→	0	1	2
10	→	1	2	0		22	→	1	2	2		34	→	1	2	4		46	→	1	2	1		58	→	1	2	3
11	→	2	3	1		23	→	2	3	3		35	→	2	3	0		47	→	2	3	2		59	→	2	3	4

RNS arithmetic operations

$$Z = X \circ Y$$

$$\begin{cases} Y \xrightarrow{RNS} (Y_1, Y_2, \cdots, Y_L) \\ X \xrightarrow{RNS} (X_1, X_2, \cdots, X_L) \end{cases}$$

presents addition, subtraction and multiplication

$$Z \xrightarrow{RNS} (Z_1, Z_2, \cdots, Z_L)$$

$$= \left((X_1 \circ Y_1) \bmod p_1, (X_2 \circ Y_2) \bmod p_2, \cdots, (X_L \circ Y_L) \bmod p_L \right)$$

Addition example

P={3, 4, 5}, X = 7, Y = 3, Z = X + Y

$$\begin{array}{lcllll} x & \longleftrightarrow & (|x|_{m_1}, & |x|_{m_2}, & \cdots, & |x|_{m_n}) \\ y & \longleftrightarrow & (|y|_{m_1}, & |y|_{m_2}, & \cdots, & |y|_{m_n}) \\ \hline |x \pm y|_M & \longleftrightarrow & (\left| |x|_{m_1} \pm |y|_{m_1} \right|_{m_1}, & \left| |x|_{m_2} \pm |y|_{m_2} \right|_{m_2}, & \cdots, & \left| |x|_{m_n} \pm |y|_{m_n} \right|_{m_n}) \end{array}$$

$$\begin{array}{lllll} 7\rightarrow(1, & 3, & 2 &) \\ +3\rightarrow(0, & 3, & 3 &) \\ \hline 10\rightarrow(1 \bmod 3, & 6 \bmod 4, & 5 \bmod 5) = (1,2,0) \end{array}$$

$$\begin{array}{ccl} x & \longleftrightarrow & \left(|x|_{m_1}, \quad |x|_{m_2}, \quad \cdots, \quad |x|_{m_n}\right) \\ y & \longleftrightarrow & \left(|y|_{m_1}, \quad |y|_{m_2}, \quad \cdots, \quad |y|_{m_n}\right) \\ \hline |x \pm y|_M & \longleftrightarrow & \left(\left||x|_{m_1} \pm |y|_{m_1}\right|_{m_1}, \left||x|_{m_2} \pm |y|_{m_2}\right|_{m_2}, \cdots, \left||x|_{m_n} \pm |y|_{m_n}\right|_{m_n}\right) \end{array}$$

$$\begin{array}{llll} 7\rightarrow(1, & 3, & 2 &) \\ -\ 3\rightarrow(0, & 3, & 3 &) \\ \hline 4\rightarrow(1 \bmod 3, & 0 \bmod 4, & -1 \bmod 5) = (1,0,4) \end{array}$$

- **Subtraction example**

$\boldsymbol{P}$={3, 4, 5}, X = 7, Y = 3, Z = X · Y

$$\begin{array}{llll} 7\rightarrow(1, & 3, & 2 &) \\ *3\rightarrow(0, & 3, & 3 &) \\ \hline 21\rightarrow(0 \bmod 3, & 9 \bmod 4, & 6 \bmod 5) = (0,1,1) \end{array}$$

Multiplication example

$\boldsymbol{P}$={3, 4, 5}, X = 7, Y = 3, Z = X · Y

$$\begin{array}{llll} 7\rightarrow(1, & 3, & 2 &) \\ *3\rightarrow(0, & 3, & 3 &) \\ \hline 21\rightarrow(0 \bmod 3, & 9 \bmod 4, & 6 \bmod 5) = (0,1,1) \end{array}$$

RNS to decimal conversion

- A little bit complicated than decimal to RNS conversion
- We have two algorithms:
  - ◘ Chinese Remainder Theorem (CRT)
  - ◘ Mixed radix conversion (MRC)
- Both algorithm uses Multiplicative inverse

Multiplicative inverse and example

- The multiplicative inverse of g of modulo p_i is denoted as g_i^{-1} and satisfies:

$$\left(g_i^{-1}g\right) \bmod p_i = 1$$

$P_i=5$	g	0	1	2	3	4
	g_i^{-1}	X	1	3	2	4

$P_i=6$	g	0	1	2	3	4	5
	g_i^{-1}	X	1	X	X	X	5

$P_i=7$	g	0	1	2	3	4	5	6
	g_i^{-1}	X	1	4	5	2	3	6

Chinese Remainder Theorem

$$|x|_M = \left| \sum_{j=1}^{n} \hat{m}_j \left| \frac{x}{\hat{m}_j} \right|_{m_j} \right|_M$$

$$\hat{m}_j = \frac{M}{m_j}$$

$$M = \prod_{j=1}^{n} m_j$$

Chinese Remainder Theorem example:

$m_1 = 7$, $m_2 = 8$, $m_3 = 9$

$x_{RNS} = (6, 4, 2)$ 1)

$$M = m_1 \cdot m_2 \cdot m_3 = 7 \cdot 8 \cdot 9 = 504$$

2)

$$|\hat{m}_1|_{m_1} = |8 \cdot 9|_7 = 2 \quad \Rightarrow \quad \left| \frac{1}{\hat{m}_1} \right|_7 = 4$$

3)

$$\left|\hat{m}_2\right|_{m_2} = |7 \cdot 9|_8 = 7 \quad \Rightarrow \quad \left|\frac{1}{\hat{m}_2}\right|_8 = 7 \qquad \text{4)}$$

$$(6, 4, 2)_{\text{RNS}(7, 8, 9)} = (20)_{\text{base 10}}$$

$$|x|_M = \left|\sum_{j=1}^{n} \hat{m}_j \left|\frac{x_j}{\hat{m}_j}\right|_{m_j}\right|_M = \left|\sum_{j=1}^{3} \hat{m}_j \left|\frac{x_j}{\hat{m}_j}\right|_{m_j}\right|_{504} =$$

$$= \left|72\left||6|_7 \cdot 4\right|_7 + 63\left||4|_8 \cdot 7\right|_8 + 56\left||2|_9 \cdot 5\right|_9\right|_{504} =$$

$$= |72 \cdot 3 + 63 \cdot 4 + 56 \cdot 1|_{504} =$$

$$= |524|_{504}$$

$$= 20$$

1. Discussion

It is clear that RNS arithmetic provides parallel arithmetic operation for addition, subtraction and multiplication. Other operations like division, magnitude comparison, algebraic-sign determination and overflow detection are relatively difficult and complex with RNS. Therefore, RNS arithmetic finds extensive applications in the field of signal processing where addition and multiplication are the predominant operations. The CRT and the M-RC process are the basic methods for conversion of residue number into weighted number system. However, subsequent chapter will also describe another algorithm for conversion of residue number into weighted number system.

$$\left|\hat{m}_3\right|_{m_3} = |7 \cdot 8|_7 = 2 \quad \Rightarrow \quad \left|\frac{1}{\hat{m}_1}\right|_7 = 5$$

microcontroller 8051 Instruction Set

The instruction set of 8051 microcontroller is divided into five groups. These are:

a) Arithmetic Instructions
b) Logical Instructions
Data Transfer Instructions
Boolean or Bit Processing Instructions
Program and Machine Control Instructions

Arithmetic Instructions

Arithmetic operations include addition, subtraction, multiplication, division, increment, and decrement operations. For 12 MHz clock frequency, all arithmetic instructions are executed in 1 µS except the INC DPTR instruction, which takes 2 µS, and the Multiply and Divide instructions, which take 4 µS. From the instruction set, it is clear that any byte in the internal Data Memory space can be incremented or decremented without going through the accumulator. One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory. The MUL A B instruction multiplies the accumulator by the data in the B register and puts the 16-bit product into the concatenated B and accumulator registers. The DIV A B instruction divides the accumulator by the data in the B register and leaves the 8-bit quotient in the accumulator, and the 8-bit remainder in the B register. The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD.

Logical Instructions

Logical instructions include AND, OR, Exclusive-OR, Complement, Clear, Rotate and Swapping operations. All accumulator based logical instructions take 1 µS time and others take 2 µS for 12 MHz clock frequency. Here, Boolean operations can be performed on any byte in the lower 128 internal Data Memory space or the SFR space, through direct addressing, without using the accumulator. The Rotate instructions shift the accumulator 1 bit to the left or right. The SWAP A instruction interchanges the high and low nibbles within the accumulator. This is a useful operation in BCD manipulations.

Data transfer Instructions

Data Transfer Instructions are used for moving data from accumulator to internal or external RAM, from one byte address of internal RAM to another byte address of internal RAM. All data transfer instructions related to internal RAM take 1 µS or 2 µS, but in case of external RAM, they take 2 µS. In data transfer operations, there are PUSH and POP instructions. The PUSH instruction first increments the Stack Pointer, and then copies the byte into the stack. The POP instruction reads data from the stack and then decrements SP. The MOVX instructions transfer data between accumulator and external data memory locations or external I/O ports. The MOVC instructions are used for reading lookup tables in Program Memory.

Microcontroller Program:

a microcontroller program has been written in Assembly Language. The program gives the output for different value of *n*. The inputs X and Y are stored in accumulator and B register respectively. The number *n* is stored in R_0 register. The complete program is presented below:

Label	Mnemonic	Comment
START	MOV A , # X	Move the input X to accumulator
	MOV F0, # Y	Move the input Y to B register
	MOV R_0 , # n	Move the input n to R_0 register
	MUL A, B	Multiply the content of accumulator
	MOV 20, A	Move the content of accumulator to location 20
	MOV 21, F0	Move the content of B register to location 21
	MOV A, # 00	Clear the accumulator
REL1	RLA	Rotate accumulator left
	ADD A, # 01	Add the immediate data 01 to accumulator
	DJNZ R_0, REL1	Decrement R_0 register and jump to the label REL1 if not zero
	ANL A, 20	Logic AND the content of 20 with accumulator
	MOV 90, A	Move the content of accumulator to Port 1 of address 90
HLT	SJMP HLT	Short jump to the label HLT
		The end

1 Summary and Conclusion

RNS results in carry-free arithmetic operation and supports high speed concurrent computations. It is thereby, extremely useful for DSP applications. This thesis is concerned with design, simulation and microcontroller implementation of some RNS based building blocks for applications in the field of DSP. The building blocks those have been studied are binary to residue converter, residue to binary converter, residue adder and residue multiplier .The underlying algorithms, based on which the above RNS modules operate, are also described. In a specific case, a new algorithm is also introduced.

all building blocks have been implemented using digital ICs and logic gates, and also using 8051 microcontroller kit. The advantage of using microcontroller is that it has very powerful instruction set and the implementation is quite easy. Moreover, same microcontroller kit may be employed for higher modulo without any additional hardware.

a new algorithm for conversion of negative binary number to residue form for moduli set $(2^n - 1, 2^n, 2^n + 1)$ is proposed. Separate circuits of positive and negative binary to residue converters of moduli $(2^n - 1)$ and $(2^n + 1)$,

a memoryless residue to binary converter circuit for moduli set $(2^n + 1, 2^n, 2^n - 1)$ is proposed. The advantage of using memoryless circuit is that the converter can be designed easily for large dynamic range. But in memory based converter, the requirement for memory size of memory increases for higher value of *n*, making it difficult to have large dynamic range.

References:

1. *Jenkins & Leon, 1977 – Used RNS to design FIR filters*
2. *Jenkins, Soderstrand, Jullien, Taylor, 1986 – Residue Number System Arithmetic: modern applications in Digital Signal Processing.*
3. *H. T. Vergos, 2001 – Proposed 200 MHz RNS Core Supposed to be used as a building block in ASIC design.*
4. *Chaves & Sousa, 2003 – Design RISC DSP based on RNS.*
5. *Chaves & Sousa, 2007 – Moduli sets for balanced processing.*
6. *Wei Wang, M. N. S. Swamy, M. O. Ahmad, a.nd Yuke Wang," A High-Speed Residue-to- Binary Converter for Three-Moduli (2-, 2k-1, 2--1-1) RNS and a Scheme for Its VLSI Implementation," IEEE Transactions On Circuits and Systems-II: Analog and Digital Signal Processing, vol. 47, No. 12, pp. 1576-1581, December 2000.*
7. *Yuke Wang, Xiaoyu Song, Mostapha Aboulhamid,, and Hong Shen "Adder Based Residue to Binary Number Converters for (2- - 1, 2n, 2n + 1) ,"IEEE Transactions On Signal Processing, vol. 50, No. 7, pp. 1772-1779, July 2002.*
8. *S. J. Piestrak," Design of High-Speed Residue-to-Binary Number System Converter Based on Chinese Remainder Theorem," Proceedings ICCD Int. Conf. Computer Design: VLSI in Computers and Processors, Cambridge, MA, pp. 508-511, October 1994.*
9. *Neil Burgess," Scaled and Unscaled Residue Number System to Binary Conversion Techniques Using the Core Function," Proceeding of the 13TH IEEE Symposium on Computer Arithmetic, pp. 250-257, July 1997.*
10. *G. Cardarilli, M. Re, and R. Lojacono," A Residue to Binary Conversion Algorithm for Signed Numbers," European Conference on Circuit Theory and Design, ECCTD'97, vol. 3, pp. 1456-1459, 1997.*

11.*A. Skavantzos," An Efficient Residue to Weighted Converter for a New Residue Number System," Proceedings of the 8th Great Lakes Symposium VLSI, LA, no.*