Sensors & Transducers, Vol. 155, Issue 8, August 2013, pp. 247-254

www.sensorsportal.com

Sensors & Transducers

http://www.sensorsportal.com

The Low Power Design Model Based on RNS-FIR

Zijun Zhang, Guofeng Qin

The Department of Computer Science and Technology, Tongji University,
4800 Caoan Road, Shanghai, 201804, China
Tel.: 8621-65981423, fax: 8621-65981423
E-mail: jerry.7.good@163.com, gfqing@yahoo.com.cn

Received: 16 May 2013 /Accepted: 12 August 2013 /Published: 20 August 2013

Abstract: High power problem become one of the most important bottleneck of the IC industry. FIR filter is a most basic element in a digital signal processing system. In this paper, we managed to design a new FIR filter that can have low power. We use a residue number system to design a FIR filter in order to reduce power. In our model, we designed a serial modulus adder and a new way to compute modulus bit by bit. We use a way of looking up table in place of multiplication. By experiment verification, these methods in each module can effectively cut down the power and area. *Copyright © 2013 IFSA.*

Keywords: Low power, Filter, RNS, FIR, Modulus, Parallel, Prefix, Adder.

1. Introduction

1.1. Background

With the rapid development of IC technology, high power problem become one of the most important bottleneck of the IC industry. It is imperative to carry out a low power design technology [1].

The main factors drive low power researches are the wide application of mobile communication equipment and its sluggish progress on battery [2]. Besides, low power can not only reduce the costs on packaging and cooling but also the cost of high performance cluster system. Through dropping the chip temperature, a lower power can make the system have a less fault rate so that improve the reliability. In terms of environmental protection, a lower power decreases noise pollution and cuts down the radiation on human body [3].

For the moment, there are several primary strategies for low power. Firstly, is to reduce the capacitance of the chip and encapsulation [2]. This method is very effectively, just expensive. Secondly is to decrease the supply voltage. This can work well, too. But as a consequence, it causes a decline in performance. Then is strategy on power management. The technology has been used widely [4]. Its efficiency depends on the application of statistical features and optimization on strategies. Finally, to use various novel circuit structures. It has a low cost and extensive application. There is a large space on optimization [5]. Above all, it is one of the most efficient ways of depressing power consumption. In this paper, I adopt the method of using various novel circuit structures. In this way, the power consumption of the FIR filter can be dropped significantly.

1.2. FIR Filter

FIR filter is a most basic element in a digital signal processing system. It has any amplitude frequency and strict linear phase frequency. At the same time, its unit sample response is limited, thus filter is a stable system [6]. Therefore, FIR filter has a

range of applications in communication, image processing, pattern recognition and other fields. The model in this paper takes a 24 shaping filter as an example, do some research on how to reduce the power consumption of FIR filter using a residue number system design. First we designed a serial modulus adder and a parallel prefix modulus adder. Then I chose the serial modulus adder as a better one after compared. Second, we have a new way to compute modulus bit by bit. Third, we use a way of looking up table in place of multiplication. All these methods have effective functions so that they can cut down the filter's power and area a lot.

2. Model and Method

2.1. Theory of Algorithms

The principle of a decimal general filter is:

$$y(n)=\sum_{i=0}^{23} c_i x_i(n)\,, \tag{1}$$

where $x_i(n)$ is the filter inputs with 16 bits wide, $c_0, c_1 \ldots c_{23}$ are filter coefficients with 12 bits wide. $y(n)$ is filter output with 16 bits wide.

Below is the calculation formula of FIR filter based on a cardinal M. Y is the filter output, c_i is the filter coefficients, x_i is filter input, N is the number of filter and M is a cardinal of RNS [7].

$$Y=\left|\sum_{i=0}^{N-1} C_i X_i\right|_M \tag{2}$$

Define a discriminant:

$$\varsigma_i(g)=\begin{cases}1 & if\ \left|C_i\right|_M=g, g=0,1,2,\ldots..M-1\\ 0 & otherwise\end{cases} \tag{3}$$

Only when $\left|C_i\right|_M=g$, $\varsigma_i(g)$ =1, otherwise $\varsigma_i(g)$ =0.

Using the discriminant, we continue to simplify the expression of Y

$$Y=\left|\sum_{g=0}^{M-1} g\sum_{i=0}^{N-1}\varsigma_i(g)X_i\right|_M \tag{4}$$

$$Y=\left|\sum_{g=1}^{M-1} g\sum_{i=0}^{N-1}\varsigma_i(g)X_i\right|_M \tag{5}$$

Divide x_i into single bits

$$Y=\left|\sum_{g=1}^{M-1} g\sum_{i=0}^{N-1}\sum_{j=0}^{L-1}2^j\varsigma_i(g)x_{i,j}\right|_M \tag{6}$$

Transform the order of sum

$$Y=\left|\sum_{j=0}^{L-1}2^j\sum_{g=1}^{M-1}\sum_{i=0}^{N-1} g\varsigma_i(g)x_{i,j}\right|_M \tag{7}$$

Divide (g=1 to M-1) into g=1 to (M-1)/2 and g=(M+1)/2 to M-1

$$Y=\left|\sum_{j=0}^{L-1}2^j\sum_{i=0}^{N-1}\left[\sum_{g=1}^{(M-1)/2} g\varsigma_i(g)x_{i,j}+\sum_{g=(M+1)/2}^{M-1} g\varsigma_i(g)x_{i,j}\right]\right|_M \tag{8}$$

Using the following formula, we go on to split from g=(M+1)/2 to M-1

$$\left|\beta X\right|_M=\left|\beta+(M-\beta)\bar{X}\right|_M,\ X\in(0,1) \tag{9}$$

Here $\bar{X}$ is a reverse of X so that $\left\langle \bar{X}*X\right\rangle_M=1$

$$\begin{aligned}&\left|\sum_{g=(M+1)/2}^{M-1} g\varsigma_i(g)x_{i,j}\right|_M=\\ &=\left|\sum_{g=(M+1)/2}^{M-1}\left[g+(M-g)\overline{x_{i,j}}\right]\varsigma_i(g)\right|_M\\ &=\left|\sum_{g=(M+1)/2}^{M-1} g\varsigma_i(g)+\sum_{g=(M+1)/2}^{M-1}(M-g)\overline{x_{i,j}}\varsigma_i(g)\right|_M\end{aligned} \tag{10}$$

Thus we abstracted a constant coefficient which can be searched in a table when implemented.

$$\alpha=\left|\sum_{g=(M+1)/2}^{M-1} g\varsigma_i(g)\right|_M \tag{11}$$

By DA algorithm, we can transport Y into following formula, and then make coefficients coalesced.

$$Y=\left|\sum_{j=0}^{L-1}2^j\sum_{i=0}^{N-1}\left[\alpha_i+\sum_{g=1}^{(M-1)/2} g\varsigma_i(g)x_{i,j}+\sum_{g=1}^{(M-1)/2} g\overline{x_{i,j}}\varsigma_i(M-g)\right]\right|_M \tag{12}$$

$$=\left|\sum_{j=0}^{L-1}2^j\sum_{i=0}^{N-1}\left[\alpha_i+\sum_{g=1}^{(M-1)/2} g\left(\varsigma_i(g)x_{i,j}+\overline{x_{i,j}}\varsigma_i(M-g)\right)\right]\right|_M \tag{13}$$

S_j is actually a convolution sum of single bits.

$$S_j=\left|\sum_{i=0}^{N-1}\left[\alpha_i+\sum_{g=1}^{(M-1)/2} g\left(\varsigma_i(g)x_{i,j}+\overline{x_{i,j}}\varsigma_i(M-g)\right)\right]\right|_M \tag{14}$$

Below is a bit combination, then we finally found out the results.

$$Y = \left| \sum_{j=0}^{L-1} 2^j S_j \right|_M \tag{15}$$

Next step is scaling.

$Y = X / K$, K is the scaling constants, Y is what we need.

In the end, transport the data of residue number system to decimal ones by CRT [6] (Chinese Remainder Theorem).

$$\langle X \rangle_M = \left\langle \hat{m}_1 \left\langle \hat{m}_1^{-1} \right\rangle_{m1} r_1 + \hat{m}_2 \left\langle \hat{m}_2^{-1} \right\rangle_{m2} r_2 \right\rangle_M \tag{16}$$

In this formula, $\langle X \rangle_M$ represents the results of X mod M, $M=m_1m_2$, $\hat{m}_i = M/m_i$, $\hat{m}_i^{-1}$ is reverse of $\hat{m}_i$ so that $\left\langle \hat{m}_i^{-1} * \hat{m}_i \right\rangle_{m_i} = 1$. Besides, $r_1=<X>m_1$, $r_2=<X>m_2$, $m_1=M/m_2$, $m_2=M/m_1$.

2.2. Architecture of Model

First of all, convert input data to binary residue number. In the process of the filter will be a single bit split step. Then do a convolution operation of a single bit. After that, merge single bit together and scaling them. Finally, switch the data from residue number system to decimal system using CRT (Chinese Remainder Theorem). The details of RNS-FIR frame can be seen in Fig. 1.

In Fig. 1, the subscript stands for the i^{th} remainder branch; the superscript stands for the i^{th} bit. For the 1^{st} and 2^{nd} branches i=0, 1, ... 4. For the rest branches i=0, 1, ... 5.

The model in this paper picks six cardinals (25-3, 25-1, 25+1, 25+3, 26-3, 26-1). The range of computation covers all bit wide of residue addition and residue multiplication, so the calculation will not overflow.

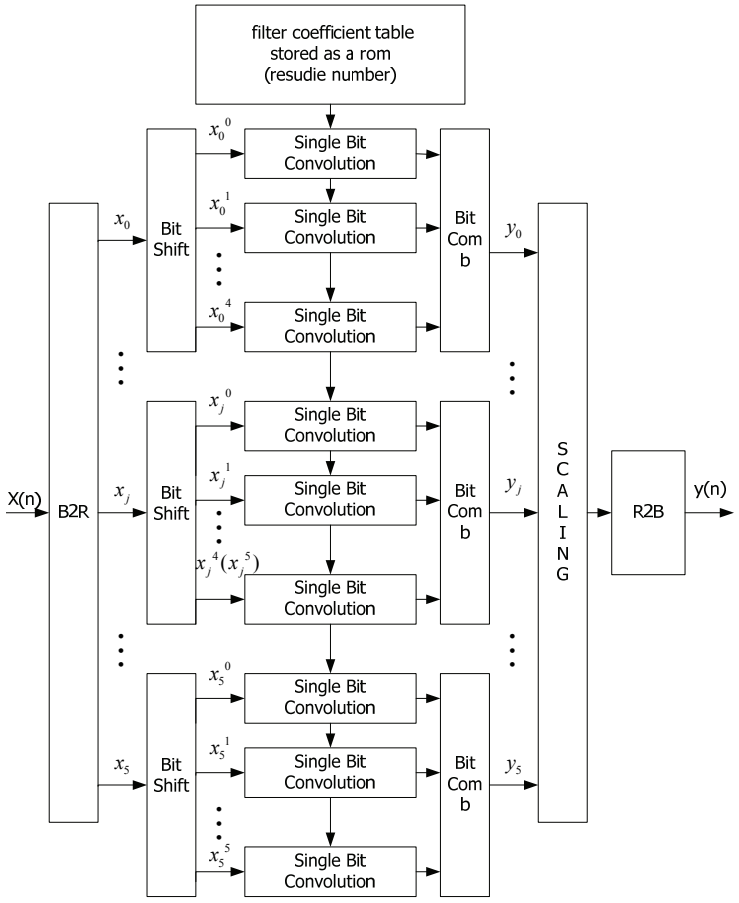

Fig. 1. RNS-FIR frame.

3. Low Power Method in Each Module

3.1. Low Power in Residue Adder

A RNS needs a number of addition operations. Traditional adder has to extend bits wide while residue adder can keep the bits scale [8] stably by modulus operation. Thus, the complexity of computing can be cut down. As a residue adder finishes a summation operator and a modulus operation at the same time, the complexity can be reduced further.

In this paper, I design two modules of residue adder. One is a serial modulus adder and the other is a parallel prefix one.

3.1.1. Serial Modulus Adder

The first scheme combined modulus principle and the ordinary adder in QUARTUS platform.

Residue adder theory is:

Respectively set up the two numbers A, M, M is cardinal, $T = 2^n - M$, S is result [9].

$$when \quad A + B + T > 2^n \tag{17}$$

$$S = A + B + T \tag{18}$$

$$otherwise \quad S = A + B \tag{19}$$

That is if output carries of $(A + B + T)$ is "1", the result will pick low-order *n* bits of $(A + B + T)$.

If output is not "1" the result will pick high-order *n* bits of A+B. The details on the RTL circuit can be seen as Fig. 2.

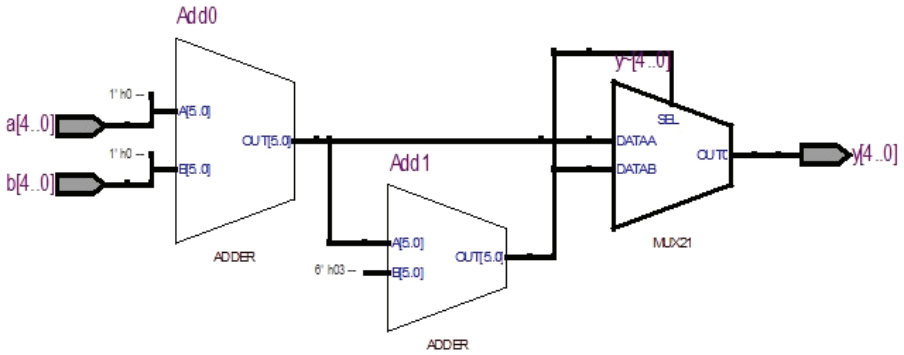

Fig. 2. Serial modulus adder's RTL.

3.1.2. Parallel Prefix Modulus Adder

The second scheme uses all basic logic units. It can achieve a faster computing speed. But it needs more logic resources.

Its principles are:

First of all, it is based on the principle of parallel prefix addition.

Generate prefix pairs used for prefix operation of $(A + B)$

$$(g_i, p_i) = (a_i b_i, a_i \oplus b_i), \tag{20}$$

g_i stands for the i[th] carry generation.

p_i stands for the i[th] carry transmission [10].

Then we have carry information of each bit.

$$\begin{cases} c_1 = g_0 \\ c_i = g_{i-1} + \sum_{j=0}^{i-2} \left(\prod_{l=j+1}^{i-1} p_l g_j \right) \quad i \geq 2 \end{cases} \tag{21}$$

This is each bit prefix operation in its group. After that, with every bit's carry information, we obtain summation result of each bit.

$$s_i = c_i \oplus p_i \tag{22}$$

On the basis of front information and the modulus theory, we revise the results according to the top bit carry value. Then real carry value of each bit is generated. In this way, we avoid to compute respectively carry values of $(A + B + T)$ and $(A + B)$. So the delay is decreased.

Firstly, get pairs of carry generation and carry transmission.

$$\begin{cases} (g_0, p_0) = (a_0 b_0, \overline{a_0 \oplus b_0}) & i = 0 \\ (g_i, p_i) = (a_i b_i, a_i \oplus b_i) & i = 1, 2, ..., k-1 \end{cases} \tag{23}$$

By observation, you can found that $M = 2^k + 1$, $T = 2^n - 2^k - 1$ so its binary representation is $\underbrace{000...001}_{n-k\,bits}\underbrace{00...001}_{k\,bits}$, therefore, we regard them as an add operation with a default carry both at the 0th bit and the k[th] bit.

When i=k, k+1, …. n-1, calculate reserve carry (g', p')

$$(g_i^{'}, p_i^{'}) = (a_i b_i, a_i \oplus b_i) \quad i = k, k+1, ..., n-1 \tag{24}$$

Add the pair $(g_i^{'}, p_i^{'})$, get pairs of carry generation and carry transmission eventually.

$$\begin{cases} (g_k, p_k) = (a_k + b_k, \overline{p_k^{'}}) & i = k \\ (g_{k+1}, p_{k+1}) = (p_{k+1}^{'} g_k^{'}, (\overline{g_k^{'}} + g_0)(p_{k+1}^{'} \oplus g_k^{'}) + g_k^{'} \overline{g_0} p_{k+1}^{'}) & i = k+1 \\ (g_i, p_i) = (p_i^{'} g_{i-1}^{'}, p_i^{'} \oplus g_{i-1}^{'}) & i = k+2, ..., n-2 \\ (g_{n-1}, p_{n-1}) = (a_{n-1} b_{n-1} + p_{n-1}^{'} g_{n-2}^{'}, p_{n-1}^{'} \oplus g_{n-2}^{'}) & i = n-1 \end{cases} \tag{25}$$

The output of highest bit is the bottom carry of A+B+T

$$c_{out} = g_{n-1} + \sum_{i=0}^{n-2}\left(\prod_{j=i+1}^{n-1} p_j g_i\right) \tag{26}$$

Do a revise once.

$$\begin{cases} c_1^{'} = g_0 \\ c_i^{'} = g_{i-1} + \sum_{j=0}^{i-2}(\prod_{l=j+1}^{i-1} p_l g_j) \quad i = k+1,...,n-1 \end{cases} \tag{27}$$

Do a second revise
For i=1, 2, …, k-1, the carry keeps the same
For i=k, k+1, …, n-1, set another variable Z

$$\begin{cases} z_k = p_{k-1}(p_k \oplus c_k^{'}) \\ z_i = \overline{\prod_{j=k+1}^{i} p_j} + p_0(p_k \oplus c_k) \quad i = k+1,...,n-1 \end{cases} \tag{28}$$

$$\begin{cases} z_k = p_{k-1}(p_k \oplus c_k^{'}) \\ z_i = \overline{\prod_{j=k+1}^{i} p_j} + p_0(p_k \oplus c_k) \quad i = k+1,...,n-1 \end{cases} \tag{29}$$

The corrected results are as follows via the variable Z.

$$\begin{cases} c_i = c_i^{'} \quad i = 1,...,k-1 \\ c_k = c_k^{'}(c_{out} + p_{k-1}) \quad i = k \\ c_i = c_i^{'}(c_{out} + z_i) \quad i = k+1,...,n-1 \end{cases} \tag{30}$$

Set the output sum is S, finally is a sum operation.

$$\begin{cases} s_0 = c_{out} + \overline{p_0} \\ s_k = c_k \oplus c_{out} \oplus \overline{p_k} \\ s_i = c_i \oplus p_i \quad i \neq 0,k \end{cases} \tag{31}$$

The comparing resources of our parallel prefix modulus adder and serial modulus adder can be seen in Table 1.

Table 1. Logic cost of two modulus adders.

Total logic elements	Parallel prefix modulus adder	Serial modulus adder
Total combinational functions	22	17
Dedicated logic registers	0	0

The simulation results of parallel prefix modulus adder and serial modulus adder can be seen in Fig. 3.

In the Fig. 3, x1, x2 is input data, y1 stands for parallel prefix modulus adder outputs, y2 represents serial modulus adder output. It is obvious in the diagram that there is no big advantage for parallel prefix modulus adder on time of computing. But it cost more logic units. Consequently, application of a serial modulus adder can effectively reduce the power consumption of the whole system.

3.2. Low Power Strategy in B2R

3.2.1 Modulus Operation Bit by Bit

This part is to transmit binary data to residue data. The input is computed by modulus operation under each cardinal. Its remainder is divided into six roads to be passed to the next module. Our cardinals are (25-3, 25-1, 25+1, 25+3, 26-3, 26-1). After calculation, this group of cardinals can cover whole computing wide, so that we can solve the problem of overflow.

A modulus operation involves division And the division isn't belong to basic computing operation. For this reason, if we directly use modulus operator, it will occupy more logic and slow down the speed. In this paper, we take a look-up table method instead of modulus operation which has a division. The details of the process can be seen in Fig. 4.

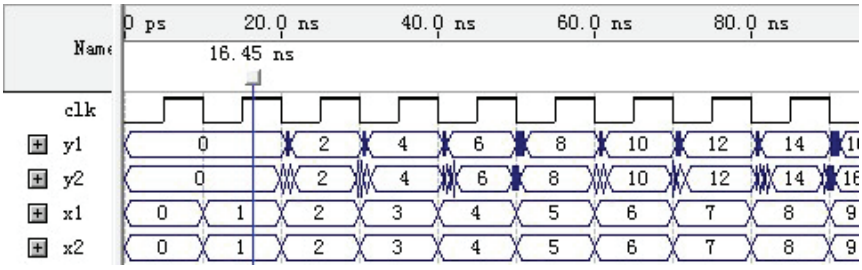

Fig. 3. Simulation results of two modulus adders.

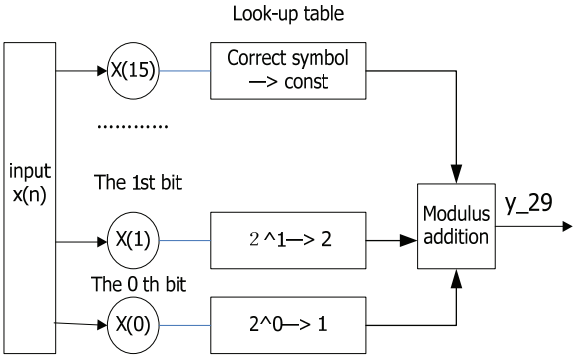

Fig. 4. Modulus process.

$x(n)$ is filter input with 16 bits wide and it's a signed data. The table lists all binary remainder of 16 bits $2^0, 2^1, 2^2 \cdots 2^{15}$ under a cardinal. There are six tables in total. For each bit of $x(n)$, if that is "1", look up the remainder in the table, if that is "0", the default remainder is "0". Add all the 16 remainders using our modulus adder. In this way, we get a remainder under a cardinal. In order to enhance speed, computing of six look-up tables is parallel.

3.2.2. Correct Symbol

As the input is a signed number, a negative input's modulus operation is quite different from positive inputs [11]. So we find a way of correcting symbol. This method can share the same logic circuit with negative modulus operation and positive modulus operation. There is no need to do another modulus operation for negative inputs as usual.

Principle is as follows:

Set the input data is x, when x is negative, there is a relationship between the true code of x's absolute value and complement code of x. In the following formulas, the subscript t stands for true code and the subscript c stands for complement code.

$$\left[|x|\right]_t + \left[x\right]_c = 2^{16} \tag{32}$$

Do a transposition:

$$\left[x\right]_c - 2^{16} = -\left[|x|\right]_t \tag{33}$$

The right of the equation $-\left[|x|\right]_t$ is exactly our input x, set out cardinal is r, do a modulus operation on both sides of equation.

$$\left\langle \left[x\right]_c - 2^{16} \right\rangle_r = \left\langle -\left[|x|\right]_t \right\rangle_r \tag{34}$$

The right side of the equation $\left\langle -\left[|x|\right]_t \right\rangle_r$ is just our output remainder.

In order to simplify the logic, we add an r in left side of equation according to the remainder principle so that we can get rid of subtraction.

$$\left\langle \left[x\right]_c + \left(r - 2^{16}\right) \right\rangle_r = \left\langle -\left[|x|\right]_t \right\rangle_r \tag{35}$$

Decompose the modulus in the left side.

$$\left\langle \left[x\right]_c \right\rangle_r + \left\langle r - \left\langle 2^{16} \right\rangle_r \right\rangle_r = \left\langle -\left[|x|\right]_t \right\rangle_r \tag{36}$$

Since r is known, $\left\langle 2^{16} \right\rangle_r$ is also a definite number, so you can get value of $\left\langle r - \left\langle 2^{16} \right\rangle_r \right\rangle_r$ then store in the table.

And $\left\langle \left[x\right]_c \right\rangle_r$ can be used as a positive number to do modulus operation.

For any input data, what need is only to detect the top bit. If it's "0", then it's positive, the remainder is "0". If it's "1", then it's negative, we look up to the table to fetch its remainder. Thus, this symbol correct operation is the same as other bits' modulus operation. So they can be done in one table.

3.3. Look-up Table Replace of Multiplication

3.3.1. Single Bit Convolution Section

Our cardinals are {27, 29, 31, 35, 59, 61}. Four of them are 5 bits while two of them are 6 bits. In the process of convolution, there will be a large number of multiplications. In order to reduce complexity of computing, we use a look-up table instead of multiplier.

In the session of single bit convolution, each bit of x will be combined to original data after it is convoluted. These operations are

$$y = \left\langle \sum_{i=0}^{5} x_i \times 2^i \right\rangle_r \tag{37}$$

where x_i is the convolution result of each bit, it has 6 bits wide. The details can be seen in Fig. 5.

This left shift is equivalent of a multiplier of 5×5, the final data wide will be extended to 10 bits. This is a huge cost. Here we have a improve method. As the result of single bit addition is less then 2 bits. So we can make an exhaustion. Every case of sum which multiplied 2^i is listed in a table. So just look at the sum of single bit and then look up the table. The details can be seen in Fig. 6.

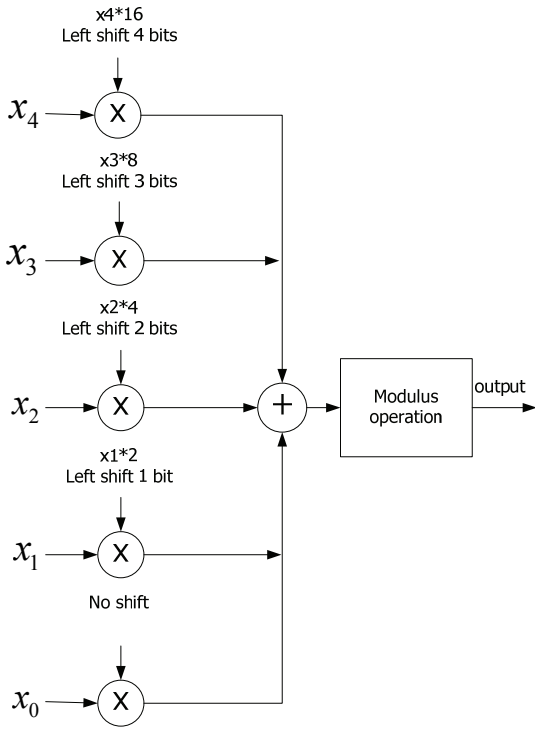

Fig. 5. Convolution process.

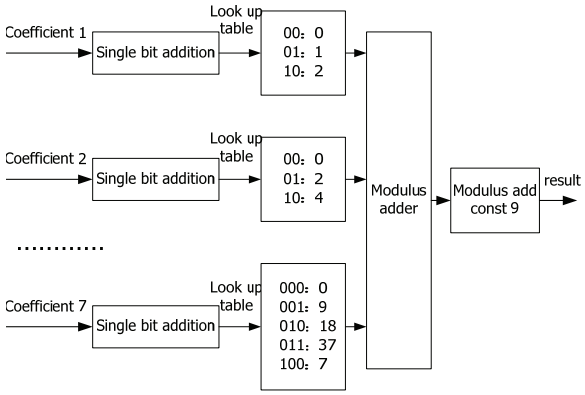

Fig. 6. Look-up table convolution.

3.3.2. The Session of Scaling

$Y=X / K$, K is the scaling constant, Y is what we need [7]. Set X is a number before scaling, Y is X's result after scaling. K is an inverse element in cardinal M_i. Set X is a number before scaling, Y is X's result after scaling. K is an inverse element in cardinal M_i. The details on the RTL scaling circuit is shown in Fig. 7.

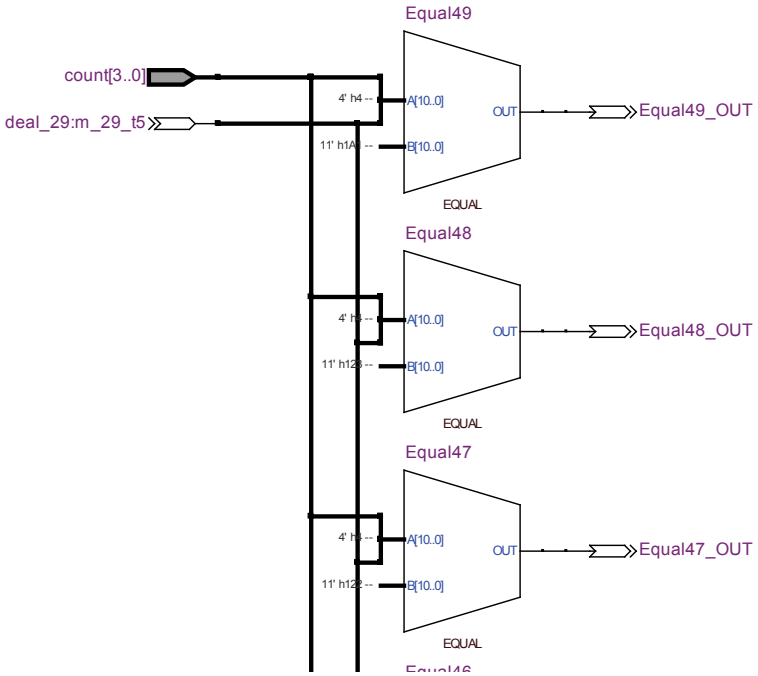

Fig. 7. Scaling RTL.

4. Experimental Verification

Based on the Design Compiler, we take respectively the SMIC 0.13 um and 0.18 um standard library under 100 MHz clock. We do a synthesis both on RNS FIR and decimal FIR. In order to compare more accurately, the decimal FIR has an identical function as RNS-FIR. It is also a 24 shaping filter and its 24 coefficients are the same as our RNS-FIR's. Under the same 100 MHz clock, they have totally the same input data and output data. Their area and power consumption are shown in Fig. 8 and Fig. 9.

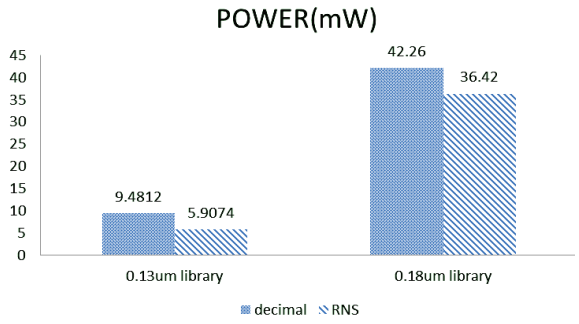

Fig. 8. Power comparisons of RNS and decimal FIR filters.

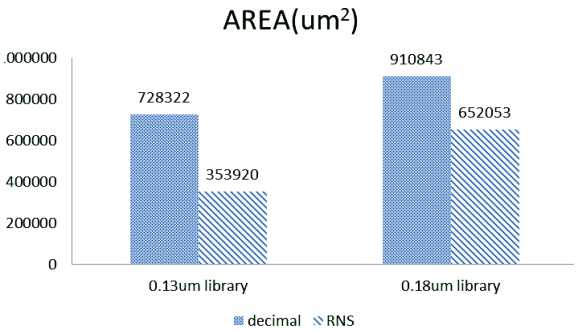

Fig. 9. Area comparisons of RNS and decimal FIR filters.

From Fig. 8 and Fig. 9, the methods in this paper can reduce effectively the power cost by 37.69 % under 0.13 um library and 13.82 % under 0.18 um library Compared to decimal FIR's power. On power area, we decrease by 51.4 % under 0.13 um library and 28.41 % under 0.18 um library compared to decimal FIR's power area.

After computing, on every area (*um*), they can cut down the power by 22 % less than decimal FIR filter under 0.13 um library and 16.94 % under 0.18 um library.

5. Conclusions

A residue number system is used to design a FIR filter to reduce power by verification from experience. The serial modulus adder and a look-up table are useful to cut down the power and area.

However, there is still some disadvantage in our design. As we have large amount of single bit computing and convolutions, so the delay in our system is longer than the decimal filter. Nonetheless, we have great throughput. So this model will be more suitable for large data computing system.

References

[1]. Hojun Kim, Jin-Gyun Chung, Minimizing switching activity in input word by offset and its low power applications for FIR filters, in *Proceedings of the International Symposium of ISCAS' 2003*, Vol. 5, 2003, pp. 297-300.

[2]. Nerurkar S. B., Abed K. H., Low-power decimator design using approximated linear-phase N-band IIR filter, *IEEE Transactions on Signal Processing,* Vol. 54, 2006, pp. 1550-1553.

[3]. Nadeem M., Wong S., Kuzmanov G., Shabbir A. Nadeem, Anjam F., Low-power, high-throughput deblocking filter for H.264/AVC, in *Proceedings of the International Symposium on SOC,* 2010, pp. 93-98.

[4]. Disney, Don, Shen, Z. John, Review of silicon power semiconductor technologies for power supply on chip and power supply in package applications, *IEEE Transactions on Power Electronics,* Vol. 28, Issue 9, 2012, pp. 4168-4181.

[5]. Woo Hyun Kim, Soohee Han, Jang Gyu Lee, An optimal FIR Filter With Fading Memory, *Signal Processing Letters, IEEE,* Vol.18, 2011, pp. 327-330.

[6]. Yu-Chi Tsao, Ken Choi, Area-Efficient Parallel FIR Digital Filter Structures for Symmetric convolutions Based on Fast FIR Algorithm, *IEEE Transactions on Very Large Scale Integration Systems,* Vol. 20, 2012, pp. 366-371.

[7]. Ben-Dau Tseng, G. A. Jullien and William C. Miller, Implementation of FFT Structures Using the Residue Number System, *IEEE Transactions on Computers,* Vol. C-28, Issue 11, November 1979, pp. 831-845.

[8]. M. Anantha, Padmanabh Shenoy and Ramdas Kumaresan, A Fast and Accurate RNS Scaling Technique for High Speed Signal Processing, *IEEE Transactions on Acoustics, Speech and Signal Processing,* Vol. 37, Issue 6, June 1989, pp. 929-938.

[9]. Ma Shang, Ye Yanlong and Hu Jianhao, Efficient Design and Implement of VLSI of Modulus Adder based on 2^n-2^k-1, *Microelectronics and Computers,* Vol. 27, Issue 10, October 2010.

[10]. Ma Shang, Researches on Key Techniques of VLSI Implementation for Digital Signal Processing Based on Residue Number System, Ph.D. Dissertation, *Dept. Elect. Eng. Xi'an Electonic Technology Univ,* Xi'an, 2010.

[11]. Chen JIan-Wen, Yao Ruo-He, Effective switcher design of 5-cardinal resuide to binary, *Journal of South China University of Technology (Natural Sciense Edition),* Vol. 38, Issue 5, 2010, pp. 55-63.