8-Bit NCL Asynchronous Multiplier based on Radix-4 Booth Algorithm

Weixiang Deng, a, Wenxiang Cheng, b, Jie Cheng, c, Leibin Ni, d and Anping He, *

1 School of Information Science and Engineering, Lanzhou University, Lanzhou, China
2 Data Center Technology Laboratory, Huawei Technologies Co., Ltd., Shenzhen, China

*Corresponding author e-mail: heap@lzu.edu.cn, a594766943@qq.com, b36399029@qq.com, cchengjie8@huawei.com, dniebin@huawei.com,

Abstract. Multiplier is one of the key modules in signal processing circuit and processor, and its robustness is especially important in the complex environment. Due to the lack of robustness of single-rail logic, a Null Convention Logic (NCL) Multiplier based on the Radix-4 Booth 8-bit fast parallel structure is proposed in this paper. The Radix-4 Booth algorithm reduces the number of partial products to lower the computing time, and the "sign generate" algorithm simplifies sign extension bit computation. In order to demonstrate the effectiveness of the proposed hardware implementation scheme, we implement an improved 8-bit NCL multiplier on vivado platform, the speed of which is 14% higher than that of the traditional Wallace NCL 8-bit multiplier.

1. Introduction

After clock frequency turned to GHz's order of magnitude, the design method of the synchronous circuit will face more and more challenges in the deep sub-micron field, such as, clock skew, clock jitter, and power consumption, etc [1]. To mitigate these problems, the asynchronous clock-less circuit methodology was designed and becoming prevalent in recent years. The clock-less feature of the asynchronous circuit shows the following merits: low power dissipation, quasi-delay-insensitivity, low noise, high reliability, etc [2]. Therefore, the asynchronous circuit is propitious to Systems-on-a-Chip (SoCs) design which is insensitive to delay while sensitive to power dissipation and performance. Among asynchronous design approaches, Null Convention Logic (NCL) [3] is the most robust method with a low-power advantage, greater optimization potential and better integration ability, and is gaining more and more attention. Therefore, this paper discusses the design of asynchronous multiplier based on NCL with the purpose of high reliability.

The multiplier is a vital component in a signal processing circuit and its performance will greatly affect the chip's performance. Asynchronous circuit has been used in multiplier design, and the current research focuses on the speed and area of multiplier. To increase the multiplier's speed, many scholars have proposed plenty of high-efficiency algorithms and the corresponding circuit structure. Two of the most famous algorithm and structures are the Booth algorithm [4] and the Wallace tree structure [5]. By separating the multiplier into several multiplier segments, the classical Booth algorithm transforms the multiplication problem into the sum of the partial product of multiplicand and multiplier segments.
Specifically, in the Booth algorithm, based on the data character of binary system multiplier section, each multiplicand's multiplication can be mapped to the equivalent shifting and subtraction operation. In [6], the authors improved the Booth encoding, employing a more optimized shift strategy, to simplify the multiple sum (difference) of the traditional Booth algorithm to a single time, which shows higher performance for the multiplier with smaller bit width. Ohkubo et al uses MUX to implement Radix-4 Booth coding circuit, and uses NAND gate structure instead of and-or structure in the process of partial product generation[7], which reduces the delay of coding circuit. Moreover, in [8], the author improved the encoding circuit by reducing the total operation time of the encoding circuit under the same scenario, and has more advantages in fast generating of partial product.

And for the signed multiplication, one needs to extend the sign of the partial product, and the "sign generate" algorithm [9] can be used to reduce the number of compressors used in the multiplier. Furthermore, by making use of Wallace tree structure, the submission progression can be compressed into only two operands, and the final result can be obtained by adding the two operands through a ripple carry adder (RCA).

This paper is organised as follows. The theory of NCL is provided in Section 2 and it is necessary to appreciate the structure of NCL circuit. Section 3 introduces our optimized design of Radix-4 Booth based on NCL. Section 4 presents the implementation of NCL multiplier design. In Section 5, we discuss our results and compare the proposed NCL multiplier and NCL multiplier based on Wallace tree in terms of delay. And Section 6 concludes the paper.

2. Null Convention Logic

NCL is a quasi-delay-insensitive asynchronous circuit design method which embeds the status into the data itself in order to eliminate timing dependence [10]. Through this method the circuit will operate correctly no matter when the input of the NCL circuit inputs is available; therefore, it only needs very few timing analysis for the NCL circuit to ensure correct operation. For example, dual-rail signal A consists of two wires, A₀ and A₁, where A₀=1, A₁=0 represent logic 0, denoted as DATA0; A₀=0, A₁=1 represent logic 1, denoted as DATA1; A₀=0, A₁=0 represent empty status, denoted as NULL which meaning that the value of A is not yet available; A₀=1, A₁=1 represent illegal status which will not appear while the circuit is regular working. In addition, the valid data is any combination of DATA0 and DATA1. During the circuit working, when the circuit output is null, the input is open to allow the valid data to enter the circuit from the input; on the other hand, when the output is valid data, the input is open to allow the Null signal to enter the circuit from the input. After that, the circuit is ready for the next valid data (Null signal).

The NCL circuit consists of 27 fundamental gates, as shown in Table 1, which constitute all function sets consisting of four or less variables, thus providing better optimization ability. As shown by Fig.1 (a), THmn gate is the primary type of threshold gate, in which, n is the input number of THmn gate, m is the threshold gate value which means that the output will be asserted when at least m inputs are asserted and deasserted when all inputs are deasserted; otherwise, it will keep its previous state.

Another kind of threshold gate is called weighted threshold gate in which one input with multiple wires may connect to this gate and threshold value of four is illustrated in Fig.1(b) described the TH44w2 gate. Note that, the first input consists of two connective wires, and hence its corresponding weight is two and the Boolean logic function is \( Z=ABC+ABD+ACD \). Usually, in THmnw₁w₂...wk gate, the first k inputs are weighted as w₁, w₂, ..., wk, and the default is 1.

![Figure 1](image-url)

**Figure 1.** (a): THmn threshold gate[10]; (b): TH44w2: \( Z=ABC+ABD+ACD \).
3. The Proposed NCL Based Improved Booth Algorithm

In this section, for the convenience of the following description we first introduce the classic Radix-4 Booth algorithm [11] which was devised by O.L.MacSorley. When we define $A=a_7a_6...a_0$ and $B=b_7b_6...b_0$ which represent multiplier and multiplicand, respectively, our aim is to compute the product $PP=A\cdot B$. The calculation formula can be written as follows:

$$PP = A \times B = A \times (b_7b_6...b_0)$$

$$= A \times (-b_7 \times 2^7 + \sum_{i=0}^{6} b_i \cdot 2^i)$$

$$= A \times \left[ \sum_{i=0}^{3} (b_{2i-1} + b_{2i} - 2b_{2i+1})4^i \right]$$

$$= \sum_{i=0}^{3} A(b_{2i-1} + b_{2i} - 2b_{2i+1})4^i$$

(1)

Each item $A (b_{2i-1}+b_{2i}+2b_{2i+1})4^i$ so-called partial product, and the $PP$ is the sum of four partial products. Moreover, the encoding scheme is designed as in Table 2, and the coefficient of the partial product is $b_{2i-1}+b_{2i}-2b_{2i+1}$.

The Radix-4 Booth coding circuit is given in [9], but there are more gate delay and circuit area when we mapping these circuit to NCL circuit directly. Therefore we redesign NCL Booth coding circuit based on Radix-4 coding rules of Booth algorithm which was first proposed by O.L.MacSorley. The Fig.2 (a) shows the result and in this figure $X$, $Neg$ represent the coefficient and sign of the partial product respectively and $a_{j-1}$ is the two consecutive bits of the multiplicand. Furthermore, $PP_j=0$ corresponds to $X=0$; $PP_j=a_0$ corresponds to $X=1$; $PP_j=a_{j-1}$ corresponds to $X=2$; and $PP_j$ is negative when $M_{2i+1}=1$, $Neg=1$. The Karnaugh map(K-map) for partial product generation component is shown in Fig.2(b).
Table 2. Radix-4 Booth algorithm encoding rule.

| $M_{2i}$ | $M_{2i+1}$ | PP       |
|----------|------------|----------|
| 000      | 0          | 0        |
| 001      | +A         | 0        |
| 010      | +A         | 0        |
| 011      | +2A        | 0        |
| 100      | -2A        | 0        |
| 101      | -A         | 0        |
| 110      | -A         | 0        |
| 111      | 0          | 0        |

![Figure 2. (a): Improved Booth encoding rule; (b): K-map for improved Booth PP generation component.](image)

And the simplified sum-of-products expressions of each output can be obtained as follows:

$$PP_j^1 = A_j^0 M_{2i+1}^0 M_{2i}^1 + A_j^0 M_{2i+1}^1 M_{2i}^0 + M_{2i+1}^1 M_{2i}^1 M_{2i-1} + A_j^1 M_{2i+1}^1 M_{2i-1} + A_j^1 M_{2i+1}^0 M_{2i-1}$$

$$PP_j^0 = A_j^0 M_{2i+1}^0 M_{2i}^0 + A_j^1 M_{2i+1}^0 M_{2i}^0 + M_{2i+1}^0 M_{2i}^1 M_{2i-1} + A_j^0 M_{2i+1}^0 M_{2i-1}$$

More importantly, in order to ensure that the PP generation component is input complete[10], each product term of an output must contain any of the rails of $M_{2i}$. Therefore, to make the circuit input-complete with respect to $M_{2i}$, $M_{2i}$ must be added to all product terms in which it is missing in either $PP_j^0$ or $PP_j^1$, but not both. So we add $M_{2i}$ to $PP_j^0$ by ANDing the product term with logic 1 which yields the following equations:

$$PP_j^0 = A_j^1 M_{2i}^0 M_{2i+1}^0 + M_{2i+1}^0 M_{2i}^0 M_{2i-1} + A_j^0 M_{2i+1}^0 M_{2i}^0 M_{2i-1}$$

4. Design implementation

Based on the output logic equation, $PP_j^1$ and $PP_j^0$ can be mapped to seven TH44 gates, three TH54w22 gates, three TH13 gates, one TH33 gate and one TH14 gate. After mapping, the proposed NCL circuit still only needs three gate delays and fifteen NCL logic gates, which reduces 3 NCL gates and 2 gate delays.
delay when compared with direct mapping, and the resulting optimized NCL improved Booth PP generation circuit is shown in Fig. 3.

Since adding 1 only occurs when \( \text{Neg} = 1 \), we can directly incorporate four sign bits \( s_0, s_1, s_2, s_3 \) into the empty low bit calculation of partial product in the compression process. Because all the partial product need to extend the sign to the highest bit of the array, which leads to the increase of the number of the compressors, the raising of the area and power consumption of the multiplier. Therefore the "sign generate" algorithm is used in the design.

Figure 3. NCL improved Booth PP generation circuit.

According to the algorithm, all sign extension bits are added to get the sum, "Sgn", and the calculation formula can be written as follows:

\[
Sgn = (s_0 \sum_{i=9}^{15} 2^i) \cdot 2^0 + (s_1 \sum_{i=9}^{11} 2^i) \cdot 2^2 + (s_2 \sum_{i=9}^{11} 2^i) \cdot 2^4 + (s_3 \sum_{i=9}^{9} 2^i) \cdot 2^6
\]  

(5)

We use \( \sum_{i=j}^{k} 2^i = 2^{k+1} - 2^j \), \( s_i = 1 - \overline{s_i} \) in the derivation above. Therefore we can simplify it as follows:

\[
Sgn = [s_0(2^{16} - 2^9)] \cdot 2^0 + [s_1(2^{14} - 2^9)] \cdot 2^2 + [s_2(2^{12} - 2^9)] \cdot 2^4 + [s_3(2^{10} - 2^9)] \cdot 2^6
\]

\[= -(s_0 \cdot 2^9 + s_1 \cdot 2^{11} + s_2 \cdot 2^{13} + s_3 \cdot 2^{15}) + 2^{16} \cdot (s_0 + s_1 + s_2 + s_3)\]

(6)

Since the second term of formula (5) only affects the seventeenth bit and the above, it can be omitted, therefore, we can get:
After coding, a total of 4 partial products (n/2=4) are generated, and we can get the final product by adding these partial products. To accelerate the operation speed and reduce the area of NCL multiplier, we adopt the Wallace tree structure composed of 4:2 compressors [12] which is made up of two full adders.

After generation the partial products, we use Wallace tree to sum the final product PP, as shown in Figure 8, which is composed of NCL full adders and half adders [13]. Where W, X, Y, Z are the four partial products, s0, s1, s2, s3 are the four signs of the partial products. After compression, the four partial products are summed by the RCA to produce the 16 bit product. It should be noted that a RCA was used instead of a carry look-ahead adder (CLA) because the performance of the asynchronous circuit depends on average-case delay but the worst-case delay in the synchronous circuit. As for the N-bit adder, the average delay of CLA and RCA is O (logN) [14]. Additionally, NCL RCA circuit has more advantages in area than CLA under the same conditions, Therefore, RCA is the preferred choice [15].

![Wallace tree and RCA](image)

**Figure 4.** Wallace tree and RCA.
5. The Simulation and Performance Analysis of NCL Multiplier

In this paper we design and implement an 8-bit NCL multiplier on the Vivado platform and the hardware description language is Verilog-1995. Furthermore, we compare the delay of the designed NCL multiplier with that of the traditional 8-bit NCL multiplier based on Wallace. As shown in Table 2, the worst delay time of the two multipliers is given. It should be noted that when the partial product is compressed, the two final operands after compression are generated serially from the lowest to the highest bit, so the final sum operation and compression operation will overlap in the operation time to a certain extent, hence, we combine the delay of the two operations. According to the Table 2, it can be concluded that although the delay of the NCL multiplier designed in this paper is about three times that of the NCL multiplier based on Wallace while generating the partial product, the compression and adder delay of the former are much less than that of the latter. The comparison results show that the delay of the multiplier designed in this paper is reduced by 14%. Moreover, it's parallel computing when the partial product is generated, and there is no connection between delay and bit-width of multiplier. Therefore, when the number of multiplier bits is larger, the advantages of the NCL multiplier designed in this paper will be more significant.

| NCL Multiplier                                                                 | Delay of PP generation | Delay of compression and RCA | Total delay |
|--------------------------------------------------------------------------------|------------------------|------------------------------|-------------|
| The proposed NCL multiplier                                                   | 7.6ns                  | 15.2ns                       | 22.8ns      |
| The NCL multiplier based on Wallace                                           | 2.5ns                  | 24ns                         | 26.5ns      |

6. Conclusion

In this paper we designed and implemented an 8-bit NCL multiplier based on Radix-4 Booth algorithm, and this NCL improved Booth circuit reduced coding time and area through the optimization of K-map. The Radix-4 Booth algorithm will reduce a half the number of the partial product, and the sign-generate algorithm will simplify the sign extension of the partial product. The final product is obtained by using the 4:2 compressor and the RCA. Compared with the 8-bit NCL multiplier based on Wallace tree, the calculation speed is reduced by 14%, and the larger the number of multiplier bits, the more significant the advantage will be.

However, compared with the multipliers currently used, 8-bit NCL multiplier has limited processing capacity. Therefore, the later work in this paper has carried out the research and development of 64-bit NCL multiplier based on this designed Booth algorithm which improves the calculation efficiency by clearly dividing the module function and increasing the circuit throughput by establishing a scientific pipeline under the condition of ensuring the minimum area and delay.

Acknowledgments

This work is supported by Huawei Technologies Co., Ltd, partly supported by Guangxi Science Technology Project (AB17129012) and the Fundamental Research Funds for the Central Universities (Grant No. Izujbky-2017-193) and by National Natural Science Foundation of China (No. 61772006) and the special fund for Bagui Scholars of Guangxi.

References

[1] Bandapati S K, Smith S C. Design and characterization of NULL convention arithmetic logic units[J]. Microelectronic Engineering, 2007, 84(2):280-287.
[2] S.C. Smith S C, Demara R F, Yuan J S, et al. Optimization of NULL convention self-timed circuits[J]. Integration the VLSI Journal, 2004, 37(3):135-165.
[3] Fant K M, Brandt S A. NULL Convention LogicTM: a complete and consistent logic for asynchronous digital circuit synthesis[C]// International Conference on Application Specific Systems. IEEE, 2002.
[4] BOOTH A.D.A signed binary multiplication technique[J]. Quarterly Journal of Mechanics and Applied Mathematics, 1951, 4(2): 236-240.

[5] Wallace C S . A Suggestion for a Fast Multiplier[J]. IEEE Trans. Electron. Comp. 1964, 13.

[6] HE An- ping, LIU Xiao-qing, CHEN Hong. Study of 8-bit Booth Asynchronous Multiplier based on two-phase handshake protocol with bounded bundle data[J]. Acta Electronica Sinica, 2018, 46(04): 961-968. (in Chinese)

[7] Ohkubo N , Suzuki M , Shinbo T , et al. A 4.4 ns CMOS 54×54-b multiplier using pass-transistor multiplexer[J]. 1995, 30(3): 251-257.

[8] Cui Xiaoping. 32×32 bit fast multiplier based on modified booth algorithm[J]. Electronic Measurement Technology, 2007(01): 82-85. (in Chinese)

[9] Millar B , Madrid P E , Jr E E S . A fast hybrid multiplier combining Booth and Wallace/Dadda algorithms[C]// Symposium on Circuits & Systems. IEEE, 1992.

[10] Smith S C , Di J . Designing Asynchronous Circuits using NULL Convention Logic (NCL)[J]. Synthesis Lectures on Digital Circuits and Systems, 2009, 4(1): 1-96.

[11] MACSORLEY O.L. High-speed arithmetic in binary computers[J]. Proc. IRE, 1961, 49: 67-91.

[12] Weinberger A . 4:2 carry-save adder module[C], 1981.

[13] Smith S C , Demara R F , Yuan J S , et al. Optimization of NULL convention self-timed circuits[J]. Integration the VLSI Journal, 2004, 37(3): 135-165.

[14] Joshi M V , Gosavi S , Jegadeesan V , et al. NCL Implementation of Dual-Rail 2S Complement 8Å 8 Booth2 Multiplier using Static and Semi-Static Primitives[C]// Region 5 Technical Conference, 2007 IEEE. IEEE, 2007.

[15] S.C. Smith. Development of a large word-width high-speed asynchronous multiply and accumulate unit[J]. Integration, the VLSI Journal, 2004, 39(1).