A High Performance Modulo $2^{n+1}$ Squarer Design Based on Carbon Nanotube Technology

Weifu Li
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA
Email: lwf1987@gmail.com

Yong-Bin Kim
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA
Email: ybk@ece.neu.edu

Abstract — In this paper, a design of high performance modulo $2^{n+1}$ squarer is proposed. The primary improvement comes from the algorithm, circuit implementation, and technology. For the algorithm, the partial product matrix reconstruction is optimized to achieve a larger range of input and fewer operation steps. The modified Wallace tree is employed to compress the partial product in each column. For circuit implementation, full adders in traditional structure are replaced by 3:2 compressors. The spare-tree based inverted End-Carry-Around (ECA) modulo $2^n$ adder is utilized to implement the final addition stage, a much better performance in terms of delay, power, and area comparing with existing design. To assess the advantage of CNT device, the same circuit is designed using CNT technology. The proposed design shows that the critical path delay and rise time of modulo $2^{n+1}$ squarer on CNT technology is 13.6 times and 9.3 times better than that of CMOS technology, respectively. The power consumption is also improved about 4 times with much better tolerance to process, voltage, and temperature (PVT) variations than CMOS counterpart.

I. INTRODUCTION

In the past decades, modular arithmetic has been playing an important role in various digital computing systems and the residue number system (RNS) is considered as one of the most common application fields [1]. In RNS, each operand is represented as a sequence of residue, e.g., $(a_1, a_2, \ldots, a_n)$, so that the any two operand computation can be considered as a combination of multiple separate operations between $a_i$ and $b_j$ performing in parallel. As a result, the overall computation speed can be significantly improved. Due to the superior performance in applications with large width operand, the RNS is perfectly suitable for high-precision application such as Fast Fourier Transforms (FFT), Finite Impulse Response (FIR) filters [2], and convolution [3].

Although multiple efficient design of modulo adders and multipliers has already been presented for RNS, there is only a few dedicated design of modulo squarer circuit. In the work of H.T. Vergos, and C. Efstathiou [4], an efficient design of modulo $2^n+1$ squarer is proposed based on the diminished-one number system. In this system, each operand is represented as $X^* = X - 1$, so that the two operand computation can be considered as $X^n = X \cdot \bar{X^n}$, so that the n-bit modulo $2^n+1$ squarer can be realized. For the partial product compression process, a Dadda tree structure is employed with full adders and half adders. Then the final Sum and Carry Vector is added by diminished-1 modulo $2^n+1$ parallel adder.

There is no doubt that the diminished-1 operand implementation demonstrated a great advantage in the aspects of delay, power, and area for modular arithmetic, the conversion between diminsed-1 system and weighted number system unnecessarily adds complexity of the system and increases the error risk in VLSI implementation. Therefore, in the work of [1], a fast low-power modulo $2^n+1$ squarer is proposed based on the algorithm in [5]. The partial product matrix is divided into four groups there and then reconstructed as an $n \times n$ matrix. The equivalent pairs in each column are shifted to further reduce the number of partial product that needed to be compressed. In addition, the correction factor resulted from partial product repositioning in each process is summarized as a single number with the value of 3. Compressors with large number of input, such as 7:2 compressor, 5:2 compressor and 4:2 compressor, are utilized to compress the partial product to achieve a greater saving of power and delay. For the final addition staged, a novel sparse-tree based inverted EAC modulo $2^n+1$ adder is introduced.

In this paper, an improved design of modulo $2^n+1$ squarer is proposed. The initial partial product matrix here is reconstructed as $(n+1)$ by $n$ matrix. And then the modified Wallace tree structure is introduced for partial product compression process. For the circuit implementation, 3:2 compressors are utilized in the modified Wallace tree structure and the GP generators in modulo $2^n+1$ adder are implemented by novel And-Or-Inverter (AOI) and Or-And-Inverter (OAI) gate to achieve a higher saving in delay, power and area. Finally, the entire circuit is implemented in CNT technology and its performance is compared with the CMOS counterpart.

II. ALGORITHM OF MODULO $2^n+1$ SQUARER

Let $X$ be a $(n+1)$ bit unsigned input denoted as $X = x_nx_{n-1} \ldots x_0$, then the square of $X$ modulo $2^n+1$ can be represented as flow:

$$Q = [X^2]_{2^{n+1}} = [\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_i x_j 2^{i+j}]_{2^{n+1}} \quad (1)$$

Different from the multiplier, the two inputs of modulo $2^n+1$ squarer are identical, so that the value of partial products $p_{ij}$ and $p_{ji}$ is always equal and these equivalent pairs can be simply replaced by shifting either $p_{ij}$ or $p_{ji}$ to next left column and removing the
other one. Therefore, the most significant bit (MSB) of each vector can be shifted to left for one bit through combining the corresponding partial product in the vector \([p_{n,n}, p_{n-1,n}, \ldots, p_{0,n}]\) respectively, so that the \((n+1) \times (n+1)\) matrix is modified to a \(n \times (n+1)\) one. Each partial product terms with weight greater than \(2^n\) can be inverted and repositioned to the corresponding position in the right part of the matrix with a correction factor as shown in equation (2). Due to \(2^{2n} \mid 2^{n+1} = 1\), terms with a weight of \(2^n\) can be repositioned to the most left column without any correction.

\[
|s2^n|_{2^{n+1}} = | - s2[\mid n]_{2^{n+1}} = |s2[\mid n] + 2^n[\mid n]|_{2^{n+1}} \tag{2}
\]

Then, as mentioned above, there are still some equivalent terms appearing twice as \(p_{i,j}\) or \(\overline{p}_{i,j}\) in the same column. Therefore, the number of vector that needed to be compressed can be further reduced and the terms in the most left column should be inverted and repositioned as well. The final partial product matrix after shifting is shown in Table-1.

<table>
<thead>
<tr>
<th>Table-1 Final Partial Product Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2^{n-1})</td>
</tr>
<tr>
<td>(p_{n,0})</td>
</tr>
<tr>
<td>(p_{n-1,1})</td>
</tr>
<tr>
<td>(\ldots)</td>
</tr>
<tr>
<td>(P_{n-3,n})</td>
</tr>
<tr>
<td>(0)</td>
</tr>
</tbody>
</table>

In addition to the partial product, the correction factor results from repositioning in each process should be also considered. The overall correction factor comes from two parts, the repositioning of partial product and the feedback of MSB during shifting and compression. For the first part, each repositioned term results in a correction factor with the value of \(2^n\) and the number of bit, needed to be repositioned, in each vector is incremented from 0 to \(n-1\). Hence, the sum of correction factor resulted from repositioning is given as:

\[
CF_1 = 2^n[(2^4 - 1) + \ldots + (2^n - 1)]
\]

\[
= 2^n[2^{n+1} - n - 3] \tag{3}
\]

For the second part, the total number of MSB needed to be fed back is \(n\), and the value of second correction factor is \(2^n\). The overall correction factor can be calculated by summing the two parts up and is given as:

\[
CF_{all} = CF_1 + CF_2 = [2^{n+1} - 3]2^n|_{2^{n+1}} = 5
\]

where \(n \geq 3\).

Comparing with the existing algorithm, the process of reconstruction is simplified to reduce the gate count on critical path. Though there is one more partial product vector after reconstruction, the vector number needed to be compressed will be still the same because there is one more equivalent pair in each column as well. In addition, since there isn’t any bitwise operation before partial product compression, the correctness of final computation result is no longer influenced by the value of input. As a result, the range of input is extended from \([0, 2^{n-1}]\) to \([0, 2^{n-1}+1]\).

III. CIRCUIT IMPLEMENTATION ON CMISTECHOLOGY

Based on the performance comparison between various possible implementation of each module, modulo \(2^{n+1}\) squarer is implemented on 32nm CMOS technology in this section. The entire circuit is divided into three parts: partial product generation and repositioning module, partial product compression module, and final addition module.

A. Partial Product Generation and Repositioning Module

In this module, the partial product matrix is generated by simple NAND gate. Instead of simply adding inverters at the output ports of nand gate, the output of nand gate can be used in compression module directly with the assistance of modified 3:2 compressors in our design, and a large amount of inverters can be saved.

B. partial product compression module

The modified Wallace tree structure is employed in this module. Different from the traditional one, 3:2 compressor are utilized to replace full adders and the schematic of 3:2 compressor [6] is shown in Figure 1 (a). As mentioned previously, based on the particular character of XOR operation, the extra inverters used as a part of the AND gate can be eliminated from critical path by moving it into the bypass of some 3:2 compressors as shown in Figure 1 (b).

Figure 1: Schematic of 3:2 Compressor

Comparing the performance of single full adder and 3:2 compressor, the critical path delay of 3:2 compressor is improved 49% with a 21.9% lower power consumption. Considering the composition of critical path delay in this module, the performance of entire module can be significantly improved as shown in equation (5).

\[
T_{Com,M} = n \times T_{S,C} \tag{5}
\]

where \(n\) is number of Wallace tree stage and \(T_{S,C}\) is delay of single compression stage.

Simulation result of two partial product compression structures in modulo \(2^{n+1}\) is shown in Table-2.
Comparing with existing structure in [1], the delay of proposed structure is improved about 17% with a 29% power and 8% area improvement.

Table- 2 Compression Structure Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Existing structure</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (ps)</td>
<td>457.2</td>
<td>392.1</td>
</tr>
<tr>
<td>Power (μW)</td>
<td>12.2</td>
<td>8.64</td>
</tr>
<tr>
<td># of transistors</td>
<td>262</td>
<td>240</td>
</tr>
</tbody>
</table>

Generally, the superior performance of Wallace Tree structure in regular multiplier and squarer comes at a high expense of none regularity interconnection and wasted area. In modulo \(2^{n+1}\) squarer, the shape of partial product matrix is always rectangular and the number of partial products in each column is always equal, which means the waste of area and irregular interconnection can be dramatically improved.

C. Final addition module

For the final addition stage, modulo \(2^{n+1}\) addition can be implemented by the modulo \(2^n\) adder which is more suitable for VLSI implementation as shown in (6) and the constant “1” here is a part of the correction factor.

\[
|A + B + 1|_{2^n+1} = |A + B + \bar{c}_{out}|_{2^n}
\]

(6)

In [7], the performance of various existing modulo \(2^n\) adder is comprehensively analyzed and the proposed fast parallel-prefix (FPP) modulo \(2^n\) adder has been considered as the fastest possible implementation. However, the Proposed FPP modulo \(2^n\) adder with wide operand has to suffer from the area and power issues due to the large amount of operator. Furthermore, the complex wire routing of the proposed FPP will further influence its performance in practice and increase the implementation difficulty. Therefore, a sparse-tree based inverted EAC modulo \(2^n\) adder is implemented in this paper.

The sparse tree modulo \(2^n\) adder combines the advantages of parallel prefix adder and conditional sum generator. It has the minimum logic depth of \(\log_2 n\) and its maximum fanout is 3. Comparing with proposed FPP, the sparse tree adder computes the carryout into each 4-bit group using a valency-2 tree structure similar to Sklansky, and the amount of operator can be dramatically reduced to achieve a lower power and area efficiency implementation with much simpler wire routing [8]. In addition, since the critical path of sparse tree adder comes from the gates used to compute carryout, the output delay skew of final sum vector can be improved. The schematic of 16-bit sparse tree modulo \(2^n\) adder and 4-bit conditional sum generator is shown in Figure 2 and Figure 3, respectively.

The simulation result of 8-bit proposed FPP and sparse-tree modulo \(2^n\) adder is shown in Table-3. Although the critical path delay of sparse tree adder is slightly larger than that of the proposed FPP one, the power consumption is significantly reduced and results in an almost 4 times better PDP. In addition, since the transistor count in sparse tree is 20% fewer than that in the proposed FPP and this percent value will be further increased for large width operand, the sparse-tree based modulo \(2^n\) adder is more area efficient.

<table>
<thead>
<tr>
<th></th>
<th>Proposed FPP</th>
<th>Sparse Tree</th>
<th>Performance Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (ps)</td>
<td>96.01</td>
<td>100.66</td>
<td>-2.5%</td>
</tr>
<tr>
<td>Power (μW)</td>
<td>60.35</td>
<td>12.54</td>
<td>3.8x</td>
</tr>
<tr>
<td>PDP(joule)</td>
<td>57.94e-16</td>
<td>12.62e-16</td>
<td>3.6x</td>
</tr>
<tr>
<td># of transistor</td>
<td>119</td>
<td>99</td>
<td>20.2%</td>
</tr>
</tbody>
</table>

In addition, the general GP generator implemented by the combination of AND and OR gate is replaced by the AOI and OAI gate in this paper. Taking an adder with logic depth of 3, the delay and power of the proposed GP generator is improved 80.9% and 39.3%, respectively comparing with general one. The number of transistors is also reduced to half in the proposed one.

IV. CIRCUIT IMPLEMENTATION ON CNT TECHNOLOGY

Nowadays, the CNT technology becomes a very competitive candidate for high-speed low-power application due to its excellent performance in the
aspects of delay, power, frequency response, and stability comparing. In this section, various simulation of modulo $2^n+1$ squarer are performed on both CNT and CMOS technology implementation.

A. PDP

The performance comparison of delay, power and PDP between modulo $2^n+1$ squarer implemented on different technologies is summarized in Table-4.

<table>
<thead>
<tr>
<th></th>
<th>CMOS</th>
<th>CNTFET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (ps)</td>
<td>401.81</td>
<td>29.63</td>
</tr>
<tr>
<td>Rise-Time (ps)</td>
<td>35.84</td>
<td>3.84</td>
</tr>
<tr>
<td>Power (μw)</td>
<td>27.42</td>
<td>11.74</td>
</tr>
<tr>
<td>PDP (joule)</td>
<td>11.02e-15</td>
<td>0.35e-15</td>
</tr>
</tbody>
</table>

It can be clearly observed that the critical path delay and rise time of modulo $2^n+1$ squarer on CNT technology is 13.6 times and 9.3 times better than that of CMOS technology respectively. The power is also improved about 4 times. Finally, a nearly 32 times better PDP is achieved by CNT technology.

B. Stability

Considering that the variation of PVT always comes together in practice, a Monte Carlo simulation is performed for both CNT and CMOS implementation of modulo $2^n+1$ squarer to see the PVT variation effects. In each Monte Carlo simulation, the threshold voltage, temperature and supply voltage of both implementations is randomly selected within the range of ±3% of reference value at the same time. For the CMOS implementation, one thousand samples are collected as shown in Figure 4 and one hundred twenty samples are collected for the CNT implementation as shown in Figure 5.

Comparing with the 21.95% maximum variation of CMOS implementation in the Figure 4, the maximum variation of CNT implementation is only 5.7% as shown in Figure 5. As shown in Figure 4 and 5, the CNT implementation of modulo $2^n+1$ squarer performs a considerable improvement of stability in the aspect of critical path delay under PVT variations.

V. CONCLUSION

In this paper, a novel modulo $2^n+1$ squarer is implemented based on the improved algorithm. Comparing with existing algorithm, the input range of modulo $2^n+1$ squarer can be extended without any extra cost, and the number of gate on critical path can be further reduced. In partial product compression module, employment of 3:2 compressor-base Wallace tree structure resulted in a considerable improvement in terms of delay, power, and area. For the final addition stage, a sparse tree inverted EAC adder is introduced to further optimize the delay and power with fewer gates and simpler wire routing. Finally, the improved modulo $2^n+1$ squarer is implemented on both CMOS technology and CNT technology. The simulation result proves that the CNT implementation is superior to CMOS counterpart in terms of power and delay. A Monte Carlo simulation is also performed to demonstrate the better PVT variation tolerance that CMOS counterpart.

REFERENCE


