ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.12, No.1, 1–6

An optimized architecture for modulo $(2^n - 2^p + 1)$ multipliers

Lei Li[1a)], **Hai Yan**[1], **Peng Yang**[1], **and Jianhao Hu**[2]

[1] *Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, No.2006, Xiyuan Avenue, High-Tech West Zone, Chengdu 611731, Sichuan, China*

[2] *National Key Lab. of Science and Technology on Communication, University of Electronic Science and Technology of China, No.2006, Xiyuan Avenue, High-Tech West Zone, Chengdu 611731, Sichuan, China*

a) *leilee@uestc.edu.cn*

Abstract: In this express, an optimized architecture for modulo $(2^n - 2^p + 1)$ multipliers on the condition $n \geq 2p$ is proposed. Compared with the state-of-art, synthesized results demonstrate that the proposed multipliers can achieve an average delay savings of about 7.5% with an average area savings of about 1.4%.

Keywords: residue number systems (RNS), multiplier

Classification: Integrated circuits

References

[1] M. Soderstrand, W. Jenkins, G. Jullien and G. Taylor: *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing* (IEEE Press, Piscataway, NJ, 1986).

[2] L. S. Didier and P. Y. Rivaille: IEEE Trans. Circuits Syst. II, Exp. Briefs **56** (2009) 46. DOI:10.1109/TCSII.2008.2010157

[3] L. Li, J. Hu and Y. Chen: IEICE Electron. Express **9** (2012) 352. DOI:10.1587/elex.9.352

[4] T.-B. Juang, C.-C. Chiu and M.-Y. Tsai: IEEE Trans. Circuits Syst. II, Exp. Briefs **57** (2010) 198. DOI:10.1109/TCSII.2010.2040302

[5] A. A. Hiasat: IEEE Trans. Comput. **49** (2000) 170. DOI:10.1109/12.833113

[6] L. Li, L. Zhou and W. Zhou: IEICE Electron. Express **9** (2012) 1141. DOI:10.1587/elex.9.1141

1 Introduction

Residue number systems (RNS), based on the Chinese Reminder Theorem (CRT), are different from the conventional binary number systems and non-weighted number systems. In RNS, concurrent computations can be performed among different moduli channels due to the carry free characteristics of RNS. In applications requiring intensive computation, such as digital signal processing [1], the characteristics of RNS is very useful. Over the last years, several modulus bases

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015

were proposed to achieve large dynamic range and many modular arithemic units were exploited [2, 3, 4, 5, 6], such as $(2^n \pm 1)$ and $(2^n - 2^p \pm 1)$. The multiplication units of modulo $(2^n - 2^p + 1)$, where n and p are integers, usually the most critical units of RNS implementations are very important for these processing systems with multiple RNS moduli channels in order to achieve more efficient and balanced systems, which often are used in large dynamic range applications and redundant residue number systems (RRNS) [6]. In [6], an improved algorithm and architecture for modulo $(2^n - 2^p + 1)$ multiplication on the condition $n \geq 2p$ was proposed.

In this paper, we extend the work in [6] and propose an optimized architecture for modulo $(2^n - 2^p + 1)$ multiplication on the condition $n \geq 2p$. In the optimized architecture, a new mergence scheme is employed to merge four intermediate terms and the produced term can be easily added to another term, which is shown in Section 2. Thus, the proposed optimized architecture can reduce the critical path by replacing a binary adder in the critical path with a CSA (Carry Save Adder) and area overhead by replacing two binary adders with a CSA, compared with the reference multipliers of [6]. As to the reference multipliers in [5], the optimized architecture avoids a binary multiplier and some other combinational logic with a binary adder and a CSA. Therefore, the optimized architecture can further improve the performance of modulo $(2^n - 2^p + 1)$ multiplication.

2 The proposed modulo $(2^n - 2^p + 1)$ multipliers

Assume that $A[n-1:0] \times B[n-1:0] = P[2n-1:0]$ and the modulo $(2^n - 2^p + 1)$ multiplication can be given as:

$$\begin{aligned}
&\langle A[n-1:0] \times B[n-1:0] \rangle_{2^n-2^p+1} = \langle P[2n-1:0] \rangle_{2^n-2^p+1} \\
&= \langle 2^n \times P[2n-1:n] + P[n-1:0] \rangle_{2^n-2^p+1} \\
&= \left\langle \begin{array}{l} P[2n-p-1:n] \# \bar{P}[2n-1:2n-p] + \bar{P}[2n-1:n] + P[n-1:0] \\ + \overbrace{0 \cdots 0}^{(n-2p)bits} \# P[2n-1:2n-p] \# \overbrace{0 \cdots 0}^{pbits} + (-2^{p+1}+3) \end{array} \right\rangle_{2^n-2^p+1}
\end{aligned} \tag{1}$$

where $Z[w:v]$ represents bits of Z originally located in positions from v (less significant) to w (more significant), the symbol # is used to concatenate bits. The term $-2^{p+1}+3$ in (1) is reserved for mergence.

Two CSAs are used in the proposed architecture. $P[2n-p-1:n] \# \bar{P}[2n-1:2n-p]$, $\bar{P}[2n-1:n]$ and $P[n-1:0]$ in (1) are computed with one CSA.

$$\begin{aligned}
&P[2n-p-1:n] \# \bar{P}[2n-1:2n-p] + \bar{P}[2n-1:n] + P[n-1:0] \\
&\xrightarrow{CSA} L[n-1:0] + 2H[n-1:0]
\end{aligned} \tag{2}$$

where $L[n-1:0]$ and $H[n-1:0]$ are sum output data and carry output data of the first CSA, respectively.

$$\begin{aligned}
\langle 2H[n-1:0] \rangle_{2^n-2^p+1} &= \langle H[n-2:0] \# \bar{H}[n-1] + H[n-1] \times 2^p - 1 \rangle_{2^n-2^p+1} \\
&= \langle H[n-2:0] \# \bar{H}[n-1] + H[n-1] \times (2^p-1) + H[n-1] - 1 \rangle_{2^n-2^p+1}
\end{aligned} \tag{3}$$

The term $H[n-1]-1$ in (3) is reserved for mergence.

$\overbrace{0 \cdots 0}^{(n-2p)bits} \# P[2n-1:2n-p] \# \overbrace{0 \cdots 0}^{pbits}$ can be merged with $H[n-1] \times$

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015

(2^p-1) into $\overbrace{0\cdots\cdots\cdots\cdots0}^{(n-2p)bits}\#P[2n-1:2n-p]\#\overbrace{H[n-1]\cdots H[n-1]}^{pbits}$. The other CSA can be used to compress $L[n-1:0]$, $H[n-2:0]\#\bar{H}[n-1]$ and $\overbrace{0\cdots\cdots\cdots\cdots0}^{(n-2p)bits}\#P[2n-1:2n-p]\#\overbrace{H[n-1]\cdots H[n-1]}^{pbits}$.

$$\begin{aligned}&\left\langle \begin{array}{l} L[n-1:0]+H[n-2:0]\#\bar{H}[n-1] \\ +\overbrace{0\cdots\cdots\cdots\cdots0}^{(n-2p)bits}\#P[2n-1:2n-p]\#\overbrace{H[n-1]\cdots H[n-1]}^{pbits} \end{array} \right\rangle_{2^n-2^p+1} \\ &=\langle L1[n-1:0]+2H1[n-1:0]\rangle_{2^n-2^p+1} \\ &=\langle L1[n-1:0]+H1[n-2:0]\#\overline{H1}[n-1]+1+H1[n-1]\times 2^p-2\rangle_{2^n-2^p+1}\end{aligned} \tag{4}$$

The term $H1[n-1]\times 2^p-2$ in (4) is reserved for mergence. $L1[n-1:0]+H1[n-2:0]\#\overline{H1}[n-1]+1$ can be further computed with a binary adder and $R[n:0]$ is the sum:

$$L1[n-1:0]+H1[n-2:0]\#\overline{H1}[n-1]+1=R[n:0] \tag{5}$$

$$\langle R[n:0]\rangle_{2^n-2^p+1}=\langle R[n-1:0]+R[n]\times 2^p-R[n]\rangle_{2^n-2^p+1} \tag{6}$$

A new mergence scheme is used to merge four reserved intermediate terms: $-2^{p+1}+3$ in (1), $H[n-1]-1$ in (3), $H1[n-1]\times 2^p-2$ in (4) and $R[n]\times 2^p-R[n]$ in (6).

$$\begin{aligned}&(-2^{p+1}+3)+(H[n-1]-1)+(H1[n-1]\times 2^p-2)+(R[n]\times 2^p-R[n]) \\ &=-\overline{H1}[n-1]\#\overbrace{\bar{R}[n]\cdots\bar{R}[n]}^{pbits}-\bar{H}[n-1] \\ &=\overbrace{1\cdots1}^{(n-p-1)bits}\#H1[n-1]\#\overbrace{R[n]\cdots R[n]}^{pbits}+H[n-1]-2^n \\ &\triangleq\overbrace{1\cdots1}^{(n-p-1)bits}\#H1[n-1]\#\overbrace{R[n]\cdots R[n]}^{pbits}+H[n-1]\end{aligned} \tag{7}$$

Thus,

$$\begin{aligned}&\langle A[n-1:0]\times B[n-1:0]\rangle_{2^n-2^p+1} \\ &\triangleq\left\langle R[n-1:0]+\overbrace{1\cdots1}^{(n-p-1)bits}\#H1[n-1]\#\overbrace{R[n]\cdots R[n]}^{pbits}+H[n-1]\right\rangle_{2^n-2^p+1} \\ &=\langle T[n:0]\rangle_{2^n-2^p+1}\end{aligned} \tag{8}$$

From (7) and (8), it can be concluded that

$$\begin{aligned}&\langle A[n-1:0]\times B[n-1:0]\rangle_{2^n-2^p+1} \\ &=\bar{T}[n]\cdot M+T[n-1] \\ &=Y[n-1:0]\end{aligned} \tag{9}$$

where $M=2^n-2^p+1$.

Fig. 1 shows the proposed architecture. The proposed modulo (2^n-2^p+1) multipliers are composed of a n-bit $\times$ n-bit binary multiplier, two n-bit CSAs, and

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015

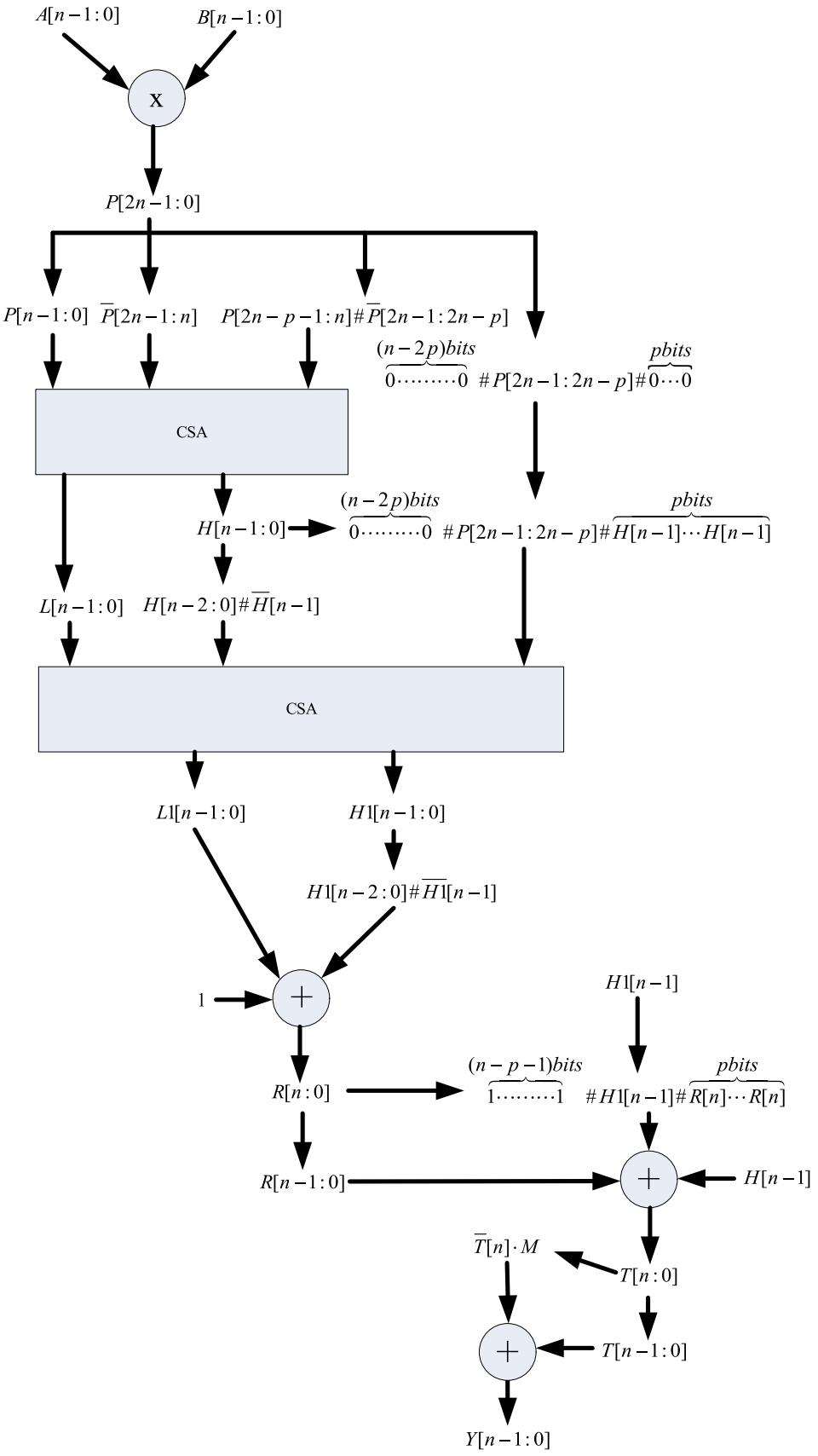

Fig. 1. The proposed architecture for modulo $(2^n - 2^p + 1)$ multiplication.

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015

three n-bit binary adders, while the reference multipliers in [5] are composed of a n-bit × n-bit binary multiplier, a $(n-p-2)$-bit × $(p+1)$-bit binary multiplier, a n-bit CSA, two n-bit binary adders and some other combinational logic and the reference modulo $(2^n - 2^p + 1)$ multipliers in [6] are composed of a n-bit × n-bit binary multiplier, a n-bit CSA, a $(n+1)$-bit binary adder, a $(n+2)$-bit binary adder, a $(n-p)$-bit binary adder, and two n-bit binary adders.

From the analytical method in [6], the proposed multipliers have a total delay of $6\lceil\log_2(n)\rceil + 5$, while the reference multipliers in [5, 6] have a delay of $8\lceil\log_2 n\rceil + 2$ and $\lceil\log_2(n+2)\rceil + \lceil\log_2(n+1)\rceil + 5\lceil\log_2 n\rceil + 4$, respectively, where $\lceil * \rceil$ is the integer greater than or equal to $*$, which is consistent with the results shown in Fig. 2.

3 Analysis

The proposed modulo $(2^n - 2^p + 1)$ multipliers in this paper were designed with VerilogHDL and exhaustively verified. The Synopsys Design Compiler tool version D-2010.03-SP5-2 for linux was used to get the synthesized results using a 90 nm CMOS process technology. The obtained results are plotted in Fig. 2 and Fig. 3. Compared with that in [5], the proposed modulo $(2^n - 2^p + 1)$ multipliers in this paper can achieve an average delay savings of 35.6% with an average area savings of 23.5% for $p = 3$, an average delay savings of 39% with an average area savings of 29.7% for $p = 4$ and an average delay savings of 44.7% with an average area savings of 23.4% for $p = 5$. When $p = 3$, compared with that in [6], the proposed modulo $(2^n - 2^p + 1)$ multipliers in this paper can achieve an average delay savings of 8.2% with an average area savings of 1.9%. When $p = 4$, compared with that in [6], the proposed modulo $(2^n - 2^p + 1)$ multipliers in this

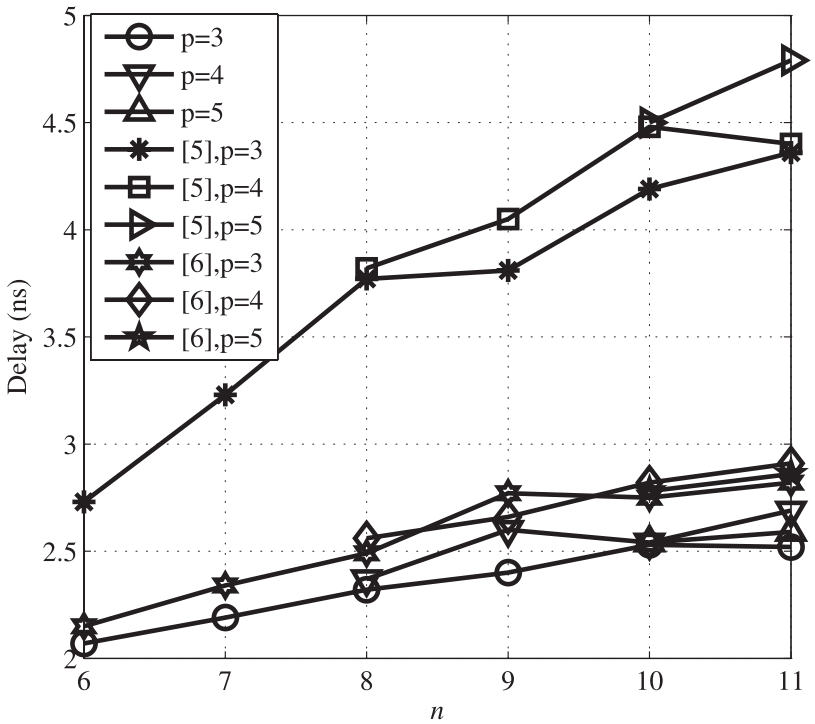

Fig. 2. The delay performance of the proposed multipliers and the reference multipliers [5, 6]

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015

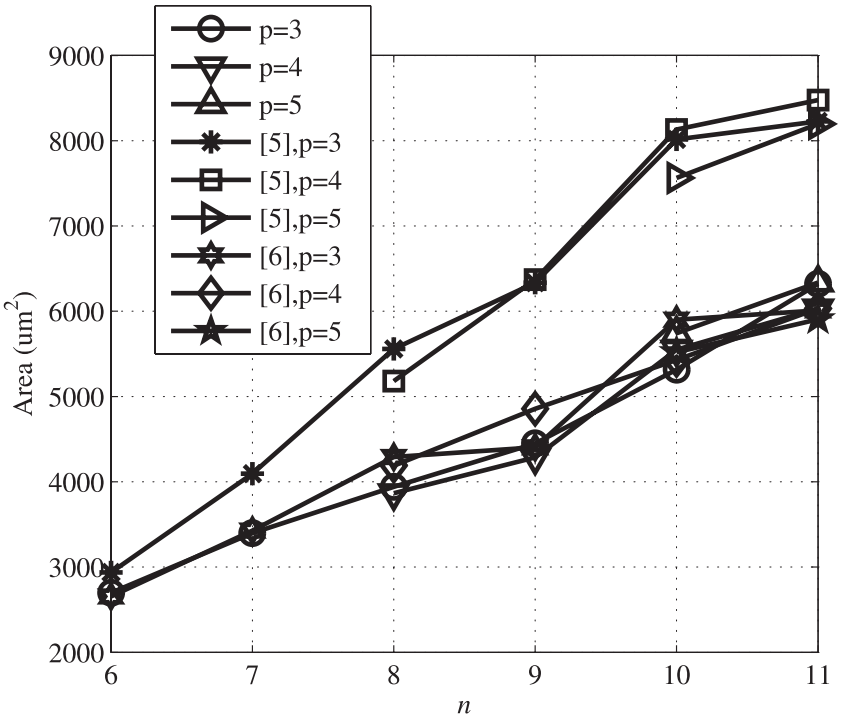

Fig. 3. The synthesized area of the proposed multipliers and the reference multipliers [5, 6]

paper can achieve an average delay savings of 6.8% with an average area savings of 4.3%. When $p = 5$, compared with that in [6], the proposed modulo $(2^n - 2^p + 1)$ multipliers in this paper can achieve an average delay savings of 9% with an average extra area overhead of 5.7%. In conclusion, compared with the state-of-art, the proposed multipliers can achieve an average delay savings of about 7.5% with an average area savings of about 1.4%.

4 Conclusion

In this express, we proposed an optimized architecture for modulo $(2^n - 2^p + 1)$ multipliers on the condition $n \geq 2p$. Compared with the state-of-art, synthesized results demonstrate that the proposed multipliers can achieve an average delay savings of about 7.5% with an average area savings of about 1.4%.

Acknowledgments

The authors would like to thank ASIC Team for providing advice and discussion.

DOI: 10.1587/elex.11.20141054
Received November 7, 2014
Accepted November 26, 2014
Publicized December 19, 2014
Copyedited January 10, 2015