Journal of Circuits, Systems, and Computers

ON THE DESIGN OF EFFICIENT MODULAR ADDERS

HARIDIMOS T. VERGOS
Computer Engineering and Informatics Department,
University of Patras, 26 500, Greece

and

CONSTANTINOS EFSTATHIOU
Informatics Department,
TEI of Athens, 12 210, Greece

Received (received date)
Revised (revised date)
Accepted (accepted date)

Modular adders are met in various applications of computer systems. In this paper we investigate a new architecture for their design that utilizes a carry save adder stage and two binary adders that operate in parallel. Realizations in static CMOS reveal that the introduced architecture leads to modular adder implementations that offer significant savings in delay and power consumption over implementations based on previously proposed architectures. In parallel, the proposed architecture offers significantly smaller implementation area for small operand widths.

1. Introduction

Modular adders play an important role in pseudorandom number generation [1] and cryptographic implementations [2,3,4]. Their main application field however, is in residue number system (RNS) based computations. The RNS has been proposed as an alternative to the binary number system because it does not suffer from the carry propagation problem since the operations are carried out on the residue digits concurrently in independent units. Furthermore, the RNS exhibits attractive fault tolerant capabilities [5]. The use of RNS has been adopted in several digital signal processing applications [6,7,8,9,10,11]. Finally, modular adders are essential building blocks for modular multipliers [12,13], residue to binary converters [14,15] and other modular operations [16,17].

Although several architectures have appeared which attack the problem of designing adders for specific moduli, for example modulo $2^n - 1$ adders [18] or $2^n + 1$ adders [19], few are available in the literature on the design of generalized modular adders. The latter architectures can be divided in two categories :

- those that require the use of ROM and
- those that are based on the use of binary adders.

Since the proposals of the latter category lead to area-time efficient implementations even for large moduli, they have received significantly more attention than those of the first category and are therefore the focus of this manuscript.

For the modulo m addition of X and Y, hereafter denoted by $|X+Y|_m$, where $X = x_{n-1}x_{n-2}\cdots x_1x_0$ and $Y = y_{n-1}y_{n-2}\cdots y_1y_0$ are two n-bit binary numbers in the range $[0, m)$ and $n = \lceil \log_2 m \rceil$, we have that

$$|X+Y|_m = \begin{cases} X+Y-m, & \text{if } X+Y \geq m \\ X+Y, & \text{otherwise.} \end{cases} \quad (1)$$

Based on Eq. 1 Bayoumi and Jullien [20], presented an architecture for modular addition that utilizes two adders and a multiplexer. A n-bit adder is used to compute $X+Y$, while a $-m$ correction is added to its output by the second $(n+1)$-bit adder. The multiplexer is then used to select between the two adders' outputs depending on the value of the carry output of the second adder.

Dugdale [21] has reduced the width of the second adder to n bits and has shown that the multiplexers can be controlled by the logical OR of the two adder's carry outputs. She has also presented an area efficient architecture that performs the modular addition using just one adder in two addition cycles.

The most efficient purely combinational architecture for single cycle modular addition has been recently proposed by Hiasat [22]. This architecture is based on generating propagate (p) and generate (g) signals for both the $X+Y$ and the $X+Y-m$ cases. The output of a carry look-ahead (CLA) unit is used to select the required set of p and g signals which are then propagated to a second CLA unit that produces the carries of the modulo addition. As shown experimentally [22], this architecture outperforms the earlier solutions [20,21] in both area and time. However, one must note that both the area and the delay of the adders proposed by Hiasat [22] depend heavily on the form of m. In the special case that $m = 2^n + 1$, multiplexing of the p and g signals is not required, and therefore very efficient adders can be derived.

In this manuscript, we investigate a modular adder architecture that consists of a carry save adder (CSA) stage and two binary adders that operate in parallel. The first binary adder computes $X+Y$, whereas the CSA and the second binary adder computes $X+Y-m$. Selection between the two results is performed by a final row of multiplexers. The use of two binary adders in parallel has been previously employed for speeding-up the last stage of a residue to binary converter [14]. This solution however has not been investigated as a candidate for a modular adder. Moreover, the scheme used in the residue to binary converter [14] can not be applied directly in the case of completely independent operands, as is the case of modular adders. Using CMOS implementations we show that when $m \neq 2^n + 1$, the adders designed following the presented architecture are on the average about 26% faster and consume about 19% less power. For small operand widths, the proposed adders

also require about 21% less implementation area than those proposed by Hiasat [22].

2. Modular Adder Architecture

Let M denote the 2's complement of m, that is $M = 2^n - m$. Then, for the modulo m addition of X and Y, we have :

$$\begin{aligned} |X+Y|_m &= \begin{cases} X+Y-m, & \text{if } X+Y \geq m \\ X+Y, & \text{otherwise.} \end{cases} \\ &= \begin{cases} X+Y+2^n-m-2^n, & \text{if } X+Y+2^n-m \geq 2^n \\ X+Y, & \text{otherwise.} \end{cases} \\ &= \begin{cases} |X+Y+M|_{2^n}, & \text{if } X+Y+M \geq 2^n \\ X+Y, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

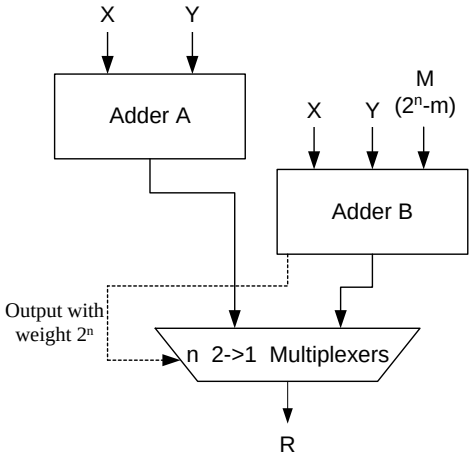

Fig. 1. Block diagram of the proposed modular adder architecture.

According to Eq. 2, a modular adder can be implemented as shown in Fig. 1. Adder A performs the addition $X + Y$, while Adder B the addition $X + Y + M$. The output of Adder B with weight 2^n indicates whether $X + Y + M \geq 2^n$, and as Eq. 2 indicates it can be used for controlling the multiplexers that select the correct output between the two addition results.

In the following we detail the implementation of Adder B. This three operand adder can be implemented as a CSA stage and an Adder D. The purpose of the CSA stage is to reduce the three operands of adder B into a sum, S, and a carry, C, vector, both of length n, which are subsequently added by Adder D. At each bit position, the CSA stage requires either a half-adder (HA) or a half-adder-like (HA*) cell [22], when the value of the corresponding bit of M, is 0 or 1 respectively.

Example 1: Consider the modulo 19 addition of $X = x_4x_3x_2x_1x_0$ and $Y = y_4y_3y_2y_1y_0$. In this case $n = 5$, $M = 13 = 01101_2$. Therefore the CSA is composed of 5 cells according to Fig. 2.

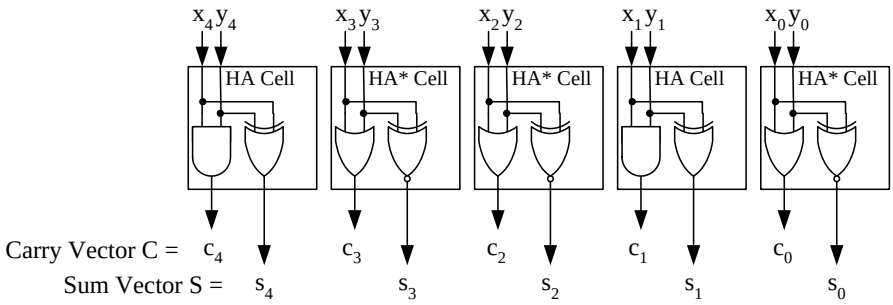

Fig. 2. CSA for the modulo 19 adder.

□

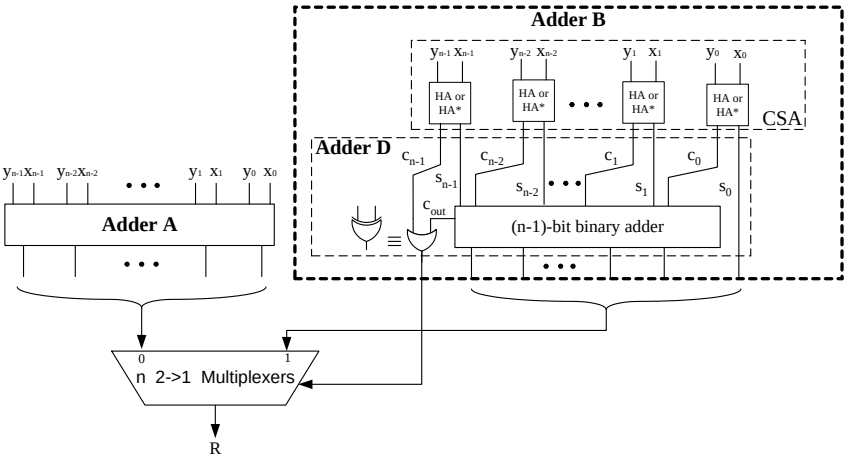

Fig. 3. Detailed implementation of the modular adder architecture.

Note that Adder D operates on the bits of the vectors $S = s_{n-1}s_{n-2}\cdots s_1s_0$ and $C = c_{n-1}c_{n-2}\cdots c_1c_0$ that have equal weight, that is, bit c_0 is added to s_1, c_1 to s_2 and so forth. We may consider this adder as being decomposed into an $(n-1)$ adder and an XOR gate as Fig. 3 suggests.

We can then notice that c_{n-1} and c_{out} signals of Fig. 3, can not be both at 1. c_{n-1} is at 1, when both x_{n-1} and y_{n-1} are at 1. The $(n-1)$-bit adder in this case

computes $(X - 2^{n-1}) + (Y - 2^{n-1}) + M - s_0 = X + Y - m - s_0$. Since $X, Y < m$, $X + Y - m - s_0 < m < 2^n$ and therefore $c_{out} = 0$. The fact that c_{n-1} and c_{out} can not be simultaneously at 1 implies that, the XOR gate of Fig. 3 never receives both inputs at 1 and can therefore be equivalently substituted by an OR gate.

3. Comparisons

In this section, we compare the architecture of Fig. 3, against the one proposed by Hiasat [22], which has been shown to be more area and time effective than earlier solutions [20,21].

Considering the delay, the proposed architecture's speed is determined by the delay of a $(n-1)$-bit binary adder plus the delay of a XOR gate and a multiplexer. Hiasat's architecture apart from the above delays also imposes the delay of a carry look ahead unit of a $(n-1)$-bit adder.

Considering the implementation area, the architecture of Hiasat requires :

(a) a CSA similar to the proposed architecture,
(b) one full binary adder,
(c) the part of a second carry computation unit that produces the carry output. One may note that this is the most area consuming part of the carry computation unit, and
(d) multiplexers to choose the correct propagate and generate signals. Depending on the value of m, the required multiplexers may reach the value of $2(n-1)$.

On the other hand the proposed architectures' area requirements are :

(a) the CSA,
(b) two full binary adders, and
(c) n multiplexers.

One however should also take into account that :

- for the bits of M that are zero, the resulting bits of S and C vectors are also the p and g bits required for adder A, and
- for the rest bits of M the resulting S bits are the complement of the p bits required for adder A.

We therefore expect that for small values of n, the area required by the proposed architecture will be significantly smaller than that of the architecture proposed by Hiasat, while for large values of n both architectures will require similar area.

To verify the above rough comparisons, we modelled several modular adders in HDL. In all cases, we assumed a Ladner-Fischer [23] parallel-prefix implementation of the carry computation units. The descriptions were then mapped in the UMC-VST 25 implementation technology (0.25 μm, up to 5-metal layers, 1.8 / 3.3 V). We followed two different optimization approaches.

In the first one, each mapped design was recursively optimized for speed. A final step of area recovery followed. This approach leads to the fastest attained designs using each architecture. The results obtained using this approach are given in Table I. In all examined cases the adders designed according to the presented architecture outperform the ones proposed by Hiasat [22]. The delay savings achieved range from 17.93% up to 36.07% with an average value of 26.11%.

Table I. Time optimization results.

Modulus	Hiasat's Architecture		Proposed Architecture		Time Savings
m	Time(ns)	Area (μm^2)	Time(ns)	Area (μm^2)	(%)
29	1.45	6226	1.19	5695	17.9
41	1.69	8967	1.36	7310	19.5
97	1.94	9672	1.39	7462	28.4
211	1.92	10773	1.37	11994	28.6
453	2.19	11563	1.40	12894	36.1

Table II. Area optimization results.

Modulus	Target Delay	Hiasat's Architecture	Proposed Architecture	Area Savings
m	(ns)	Area (μm^2)	Area (μm^2)	(%)
29	1.74	4033	3404	15.6
41	2.03	6345	3561	43.9
97	2.33	6610	4854	26.6
211	2.30	7998	7217	9.8
453	2.63	8686	8046	7.4

In the second approach all mapped designs were optimized until they achieved a specific operating frequency. This frequency was set at 0.8 of the maximum frequency reported in Table I for Hiasat's architecture. Successive area minimization steps were then applied until the tool was unable to provide a smaller design. The results obtained using this approach are given in Table II. The implementation area savings offered by the proposed architecture compared to the one proposed in [22], range from 7.36% up to 43.87%, with an average value of 20.6%.

Table III. Power consumption results.

Modulus	Hiasat's Architecture	Proposed Architecture	Power Savings
m	Power (mW)	Power (mW)	(%)
29	1.066	0.932	12.6
41	1.466	0.887	39.5
97	1.368	1.096	19.9
211	1.699	1.502	11.6
453	1.781	1.599	10.2

Since power reduction is an important target in battery operated systems, we also performed power consumption comparisons. For them, we used the netlists of

Table II. All designs were placed and routed using keeping the design constraints, such as output load, max fanout, and floorplan initialization information constant for both compared architectures. Then the parasitics were extracted from the actual layouts of the designs. The gathered design data for each design were then passed to our power analysis tool which estimated the average power required per vector over 5000 random vectors applied. Our experimental results on power consumption are shown in Table III. The results indicate that the savings offered by the proposed architecture compared to the one proposed by Hiasat, range from 10.2% up to 39.5%, with an average value of 18.8%.

4. Conclusions

Residue-based applications, such as RNS implementations for digital signal processing, cryptography and pseudorandom number generation make the use of high-speed combinational modular adders imperative. Moreover, modular adders are essential elements of modular multipliers and residue to binary converters. To this end, a new modular adder architecture has been presented. The proposed architecture is built upon a CSA and two binary adders, operating in parallel. The experimental results indicate that the proposed adders offer significant savings in operation delay and power consumption. In parallel, for small operand widths they provide more compact designs.

Acknowledgements

This work was co-founded by 75% from E.C. and by 25% from the Greek Government under the framework of the Education and Initial Vocational Training Program - Archimedes II

References

1. D. H. Lehmer, *Proc. of the* 2^{nd} *Symp. on Large-Scale Digital Calculating Machinery*, Cambridge, MA, Harvard University Press, (1951), pp. 141–146.
2. R. Zimmermann, A. Curiger, H. Bonnenberg, H.Kaeslin, N. Felber, and W. Fichtner, "A 177 Mb/s VLSI implementation of the international data encryption algorithm", *IEEE J. of Solid-State Circuits*, **29**, (1994), 303–307.
3. A. Curiger, "VLSI architectures for computations in finite rings and fields", Ph.D. thesis, Swiss Federal Institute of Technology, 1993.
4. X. Lai and J. L. Massey, "A proposal for a new block encryption standard", *Proc. of EUROCRYPT '90, Lecture Notes in Computer Science*, **473**, Springer, (1991), pp. 389-404.
5. B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley, New York, (1989), pp. 119–123.
6. W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters", *IEEE Trans. on Circuits and Systems*, **CSA-24**, (1977), 191–201.
7. W. K. Jenkins, "Recent advances in residue number techniques for recursive digital filtering", *IEEE Trans. on Accoustics, Speech and Signal Processing*, **27**, (1979),

19–30.
8. W. K. Jenkins, "The design of specialized residue classes for efficient recursive digital filter realization", *IEEE Trans. on Accoustics, Speech and Signal Processing*, **30**, (1982), 370–380.
9. M. Soderstrand, M. A. W. Jenkins, G. Jullien and F. Taylor, *Residue Number System Arithmetic : Modern Applications in Digital Signal Processing*, IEEE Press, New York, 1986.
10. K. M. Elleithy and M. A. Bayoumi, "Fast and flexible architectures for RNS arithmetic decoding", *IEEE Trans. on Circuits and Systems II*, **39**, (1992), 226–235.
11. E. Di Claudio, F. Piazza and G. Orlandi, "Fast combinatorial RNS processors for DSP applications", *IEEE Trans. on Computers*, **44**, (1995), 624–633.
12. K. M. Elleithy and M. A. Bayoumi, "A systolic Architecture for Modulo Multiplication", *IEEE Trans. on Circuits and Systems II*, **42**, (1995), 725–729.
13. A. Hiasat, "New efficient structure for a modular multiplier for RNS", *IEEE Trans. on Computers*, **49**, (2000), 170–174.
14. S. J. Piestrak, "Design of high-speed residue-to-binary number system converter based on Chinese remainder theorem", *Proc. of the International Conference on Computer Design (ICCD)*, (1994), 508–511.
15. Y. Wang, "Residue to binary converters based on new Chinese remainder theorems", *IEEE Trans. on Circuits and Systems II*, **47**, (2000), 197–205.
16. D. Dimauro, S. Impedovo and G. Pirlo, "A new technique for fast number comparison in residue number system", *IEEE Trans. on Computers*, **42**, (1993), 608–612.
17. V. Paliouras and T. Stouraitis, "Multifunction architectures for RNS processors" *IEEE Trans. on Circuits and Systems II*, **44**, (1999), 1041–1054.
18. L. Kalamboukas, D. Nikolos, C. Efstathiou, H. T. Vergos and J. Kalamatianos, "High-speed parallel-prefix modulo 2^n-1 adders", *IEEE Trans. on Computers*, **49**, (2000), 673–680.
19. H. T. Vergos, C. Efstathiou and D. Nikolos, "Diminished-one modulo $2^n + 1$ adder design", *IEEE Trans. on Computers*, **51**, (2002), 1389-1399.
20. M. Bayoumi M. and G. Jullien, "A VLSI implementation of residue adders.", *IEEE Trans. on Circuits and Systems I*, **34**, (1987), 284–288.
21. M. Dugdale, "VLSI Implementation of residue adders based on binary adders", *IEEE Trans. on Circuits and Systems II*, **39**, (1992), 325–329.
22. A. Hiasat, "High-speed and reduced-area modular structures for RNS", *IEEE Trans. on Computers*, **51**, (2002) 84–89.
23. R. E. Ladner and M. J. Fischer, "Parallel prefix computation", *J. of the ACM*, **27**, (1980), 831–838.