Fast modulo $2^n - 1$ and $2^n + 1$ adder using carry-chain on FPGA

Laurent-Stéphane Didier
Université de Toulon
Laboratoire IMATH
La Garde, France
Email: Laurent-Stephane.Didier@univ-tln.fr

Luc Jaulmes
Barcelona Supercomputing Center - CNS
Universitat Politecnica de Catalunya
Barcelona, Spain

***Abstract*—Modular addition is a widely used operation in Residue Number System applications. Specific sets of moduli allow fast RNS operations such as binary conversions and multiplications. Most of them use modulo $2^n - 1$ and $2^n + 1$ additions. This paper presents four fast and small architectures for these specific moduli targeting modern FPGAs with fast carry chains. The use of this arithmetic dedicated feature allows fast and small modular adders. Our modulo $2^n - 1$ adders have a single zero representation. Our modulo 2^n+1 adders are designed for binary and diminished-one representation with and without zero value management.**

***Index Terms*—Modular adder, carry-chain, FPGA, RNS, 2^n-1 and $2^n + 1$ moduli**

I. Introduction

Modular operations are widely used in several fields such as residue number systems (RNS) and cryptographic applications. It is possible to enhance RNS computations by using the specific moduli $2^n - 1$ and $2^n + 1$. The RNS arithmetic operations and RNS to binary converters on specific modular bases make a large use of arithmetic operators for specific moduli [1], [2].

Several efficient architectures for modular multiplications have been proposed for FPGA [3], [4], [5]. Less work has been made in order to improve modular additions [6] whereas several VLSI architectures for specific moduli have been published [7], [8], [9], [10].

Some recent work detailed efficient architectures on Field Programmable Gate Arrays (FPGAs) exploiting their dedicated logic [11]. Fast carry chains are a distinctive feature of modern FPGAs. They bypass the general routing network and allow fast ripple carry addition. Such ressources have recently been exploited in order to improve compressor trees [12], [13] and large adders [14]. As a general fact, using this feature allows fast and compact design by reducing routing pressure [15].

In this article we are interested in modular additions modulo $2^n - 1$ and $2^n + 1$ and propose efficient implementations on FPGAs that makes use of the fast carry logic. In the remainder of this section, we introduce the carry-chain mecanism. A new modulo $2^n - 1$ adder for FPGAs is introduced in section III and a new modulo $2^n + 1$ adder is described in section V. Comparisons with other existing adders are detailled in sections IV and VI.

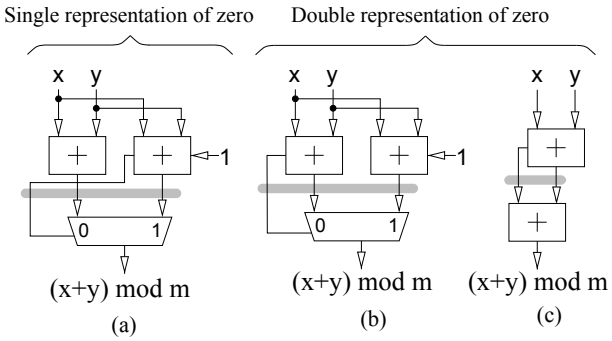

Fig. 1. Beuchat's modulo $2^n - 1$ adders

II. Generic modular addition

The addition of two positive numbers X and Y modulo m can be written as follows:

$$X + Y \bmod m = \begin{cases} X + Y - m, & \text{if } X + Y \geq m \\ X + Y, & \text{if } X + Y < m \end{cases} \quad (1)$$

However in this way, this operation requires a comparison that remains expensive in size and delay. For some specific moduli m, the cost of the comparison can be reduced.

Let x_k denote the bit of weight 2^k in X, $[X]^p$ denote the bits of X with weight larger than 2^{p-1}, and $[X]_{p-1}$ those with weight smaller than 2^p, in the way that,

$$X = \sum_{k=0}^{n-1} 2^k x_k = [X]^p 2^p + [X]_{p-1}$$

We use also the notation $|X|_m$ for $X \bmod m$. It can be noted that $|X|_{2^n} = [X]_n$

III. Carry-chain modulo $2^n - 1$ adders

A. Modulo $2^n - 1$ addition with double zero representation

Zimmerman gave an equation for computing modulo $2^n - 1$ addition [16]. It is, with our notations:

$$|X + Y|_{2^n-1} = \begin{cases} [X + Y + 1]_n, & \text{if } X + Y \geq 2^n - 1 \\ [X + Y]_n, & \text{if } X + Y < 2^n - 1 \end{cases} \quad (2)$$

Several architectures have been proposed in [17] for this operator (Fig. 1). Most of them use the output carry C of

https://doi.org/10.1109/ACSSC.2013.6810475
Published in the 2013 Asilomar Conference on Signals, Systems and Computers, doi: 10.1109/ACSSC.2013.6810475

$X+Y$ to decide to output $[X+Y]_n$ or $[X+Y+1]_n$. However, when $[X+Y]_n = 2^n-1$, they output 2^n-1 instead of 0. This implies a double representation of 0 because these numbers, $11\cdots1$ and $00\cdots0$ in binary, are congruent modulo 2^n-1.

B. Simplification for FPGA

Our proposition is to compute successively C and $X+Y+C$ using the same carry chain (Fig. 2). Furthermore, since only the output carry is needed, the design can be compacted and we can use up to 4 bits per LUT to compute C on modern FPGAs.

This allows us to have a very low delay (avoiding muxes and the main routing framework) and an area of $1.5\times$ the area of a n-bit adder.

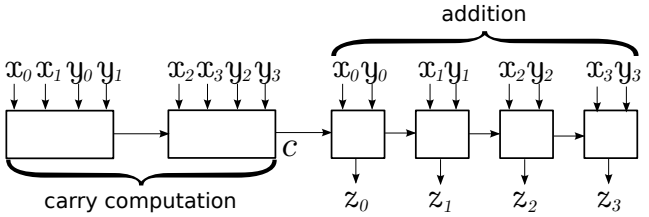

Fig. 2. Example of 4-bit chained modular adder

Let x_{2i}, x_{2i+1}, y_{2i}, and y_{2i+1} be the inputs of the i-th LUT, C_i its carry input and C_{i+1} its carry output. Their sum can be expressed as follows:

$$s_i + 4C_{i+1} = x_{2i} + 2x_{2i+1} + y_{2i} + 2y_{2i+1} + C_i$$

where we ignore $s_i \in [0,3]$. Obviously, this sum is the sum of two radix-4 digits and an input carry. Similarly to radix-2 addition, a carry is generated, propagated or killed depending on the added digits.

It is possible to substitute the carry generation for radix-4 digits by the calculation of the carry generated by the sum of two bits [18]. The idea is to build two tables f and g which input the two radix-4 digits and produce two bits that we add. These tables are constructed so that the sum of these bits has the same generate, propagate and kill cases than for the sum of two radix-4 digits.

More formally, we compute:

$$\begin{aligned} s'_i + 2C_{i+1} =& f(x_{2i}, x_{2i+1}, y_{2i}, y_{2i+1}) + \\ & g(x_{2i}, x_{2i+1}, y_{2i}, y_{2i+1}) + C_i \end{aligned} \quad (3)$$

where:

$$\begin{aligned} f(x_{2i}, x_{2i+1}, y_{2i}, y_{2i+1}) =& (x_{2i+1} \odot y_{2i+1}) \oplus \\ & ((x_{2i+1} \oplus y_{2i+1}) \odot (x_{2i} \oplus y_{2i})) \\ g(x_{2i}, x_{2i+1}, y_{2i}, y_{2i+1}) =& (x_{2i+1} \odot y_{2i+1}) \oplus \\ & ((x_{2i+1} \oplus y_{2i+1}) \odot (x_{2i} \odot y_{2i})) \end{aligned}$$

C. HDL implementation

The HDL implementation of this architecture is greatly simplified by equation (3). Indeed, the computation of the output carry C of $X+Y$ is the most significant bit of the sum of two bit vectors and an input carry. The other bits of the result are useless. This sum is efficiently inferred by the design tools by using the fast carry mechanism of the targeted FPGAs.

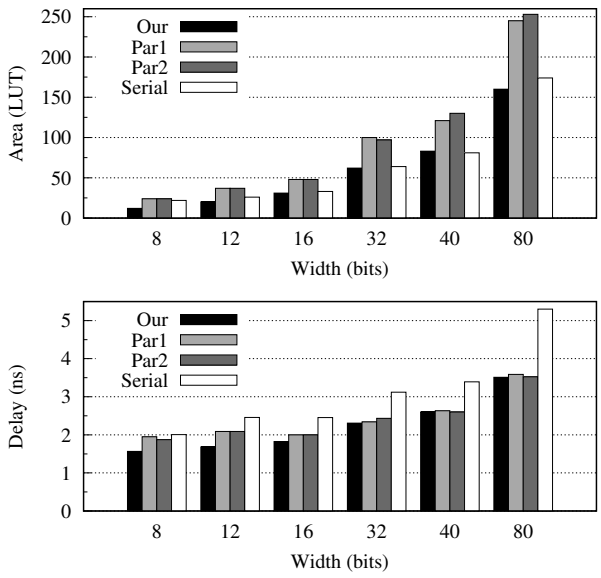

Fig. 3. Size and delay comparison between modulo 2^n-1 adders

D. Single zero representation

Managing several representations for zero may not be practical in a more complex design using modular operators. Our design can be adapted in order to get a single representation of zero.

The second stage remains the same, while the first stage simply computes the output carry of $X+Y+1$ instead of $X+Y$. The behaviour of this carry is summarized as follows:

- If $X+Y \geq 2^n$, the output carry of $X+Y+1$ is still $C_{out}=1$.
- If $X+Y < 2^n-1$, then $X+Y+1 < 2^n$, and C_{out} remains 0.
- If $X+Y=2^n-1$, this case used to generate the second representation of 0. We now have $X+Y+1=2^n$, and $C_{out}=1$ instead of 0. Thus, according to equation (2), the second stage outputs zero.

IV. Comparison with other FPGA modulo 2^n-1 adders

We compared our adder to existing modulo 2^n-1 adders for FPGA. Jean-Luc Beuchat proposed several of them [17] which rely on the principle that the output carry of $X+Y$ should be injected back in the sum as an input carry. This can be done on FPGA serially with a second adder (Fig. 1.c) or by computing in parallel two sums with the two possible carry values and a multiplexer (Fig. 1.a and Fig. 1.b).

The designs have been described in VHDL and synthetized, placed and routed with Xilinx ISE 13.4. The targeted FPGA is the Virtex 6 XC6VLX75T. The comparative results are summarized in figure 3, where the upper graph shows the

https://doi.org/10.1109/ACSSC.2013.6810475
Published in the 2013 Asilomar Conference on Signals, Systems and Computers, doi: 10.1109/ACSSC.2013.6810475

number of LUTs and the lower one shows the delay. The displayed measures are respectively related to our adder, the adders displayed in figure 1.a, 1.b and 1.c.

Our adder is slightly faster (4 to 10 percent depending on widths) than its fastest competitors, and is much smaller. Indeed, only three fourths of the area of the parallel architectures is needed. Furthermore, our adder is about the same size as the slower serial architecture, but is significally (up to 33 percent) faster.

V. Carry-chain modulo $2^n + 1$ adders

A. Modulo $2^n + 1$ additions

If the modulus m is 2^n+1, the equation (1) can be rewritten as follows:

$$|X+Y|_{2^n+1} = \begin{cases} |X+Y|_{2^n}, & \text{if } X+Y < 2^n \\ 2^n, & \text{if } X+Y = 2^n \\ |X+Y-1|_{2^n}, & \text{if } X+Y \geq 2^n+1 \end{cases} \quad (4)$$

The comparisons to 2^n+1 involved in (4) can be substituted by comparisons to 2^n which are simpler to do. This substitution can be made if we compute $X+Y+1$ rather than $X+Y$ [17], [19]. However, if we use the classic binary representation to compute this addition, we introduce a bias at each addition. This can be avoided by using diminished-one representation. In this arithmetic, a number X is represented by $X' = X-1$ [20]. Thus, there is no bias introduced in $X'+Y'+1$ because the result is still in diminished-one. The drawback is that zero can not be represented and extra logic is needed to manage this case.

Let us note I_X the bit that indicates if a number X is zero. If $X = 0$ then $I_X = 0$ and $X' = 0$.

1) Existing adders: Several modulo 2^n+1 adders for FPGA have been collected by Beuchat [17]. They are targetting binary (Fig. 4.i to 4.k) or diminished-one input (Fig. 4.g and 4.h). Similarly to 2^n-1 adders, they are built using serial or parallel structures.

From equation (4) it can be observed that the modular sum $|X+Y+1|_{2^n+1}$ depends from the output carry C_{out} of $X+Y$. The adders depicted in figure 4 are based on this observation which is summarized as follows:

$$|X+Y+1|_{2^n+1} = \begin{cases} \left|X+Y+\overline{C_{out}}\right|_{2^n}, & \text{if } X+Y \neq 2^n-1 \\ 2^n, & \text{if } X+Y = 2^n-1 \end{cases} \quad (5)$$

B. Simplification for FPGA

Similarly to our 2^n-1 adders, the adders we propose are built in two stages. The first computes a carry that is injected in the second stage which is a binary adder.

1) Diminished-one modular adder: The implementation of equation (5) using diminished-one numbers is straightforward. It is built from equation (6) which is adapted from eq. (5). It gives a structure similar to our modulo 2^n-1 adder. We use the same carry computation structure as a first stage and inject the inverted carry into a binary adder. This adder (Fig. 5), similarly to the adders in figure 4.g and 4.h, does not manage the zero values.

$$\left|X'+Y'+1\right|_{2^n+1} = \begin{cases} X'+Y'+1, & \text{if } X'+Y' < 2^n-1 \\ \text{exception}, & \text{if } X'+Y' = 2^n-1 \\ \left|X'+Y'\right|_{2^n}, & \text{if } X'+Y' \geq 2^n \end{cases} \quad (6)$$

Because of the structure of modern FPGA, we can not insert an inverter in the carry chain as schematized. As a consequence, we compute the inverted carry all along the chain before the inversion point. Each level outputting an inverted carry has an inverted carry as input, so we swap carry kill and generate cases, and keep the propagate cases from our original f and g functions. In practice, it suffices to use the functions $\overline{f}$ and $\overline{g}$ and the opposite of the first carry input, since $\overline{f}+\overline{g} = 10_b - (f+g)$.

2) Diminished-one modular adder with 0 management: We remind that a number X is equal to zero if $I_X = 0$ and its diminished-one representation $X' = 0$. The specific management of zero arises in two cases:

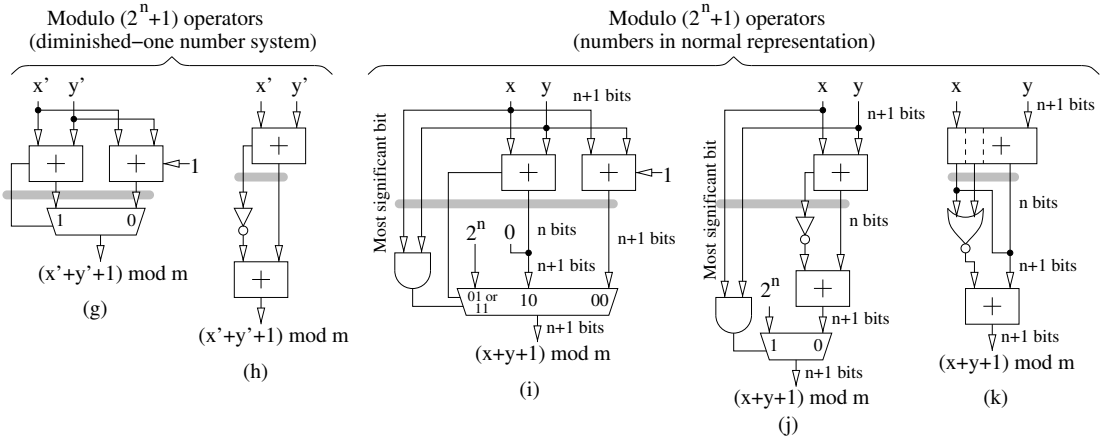

Fig. 4. Beuchat's modulo $2^n + 1$ adders

https://doi.org/10.1109/ACSSC.2013.6810475
Published in the 2013 Asilomar Conference on Signals, Systems and Computers, doi: 10.1109/ACSSC.2013.6810475

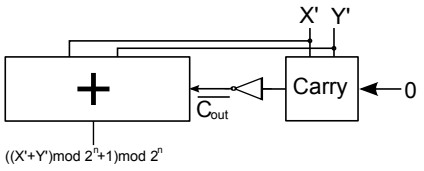

Fig. 5. Schematic of the diminished-one modulo $2^n + 1$ adder

Summing a 0: $X = 0$ or $Y = 0$, thus X' or Y' is zero. We replace the computed carry $\overline{C_{out}}$ by 0 to get $X'+Y'+0 = X'$ or $X'+Y'+0 = Y'$. This is done by a simple logic between the carry computation and the binary adder (Fig. 6). Furthermore, $I_{X+Y} = 0$ when $I_X = I_Y = 0$.

The sum yields 0: In this case, X' and Y' are both non-zero, $X' + Y' = 2^n - 1$ and we should obtain $X' + Y' + \overline{C_{out}} = 0$ and $I_{X+Y} = 0$. We already have $\overline{C_{out}} = 1$, thus $X' + Y' + \overline{C_{out}} = 2^n$. It comes without further modification that $X'+Y'+\overline{C_{out}} = 0$ on the n-bit adder, the most significant bit being ignored.

Extra logic is required to compute I_{X+Y}. We note C'_{out} the output carry of $X' + Y' + \overline{C_{out}}$. We know that if $\overline{C_{out}} = 1$ then $X' + Y' < 2^n$. When $C'_{out} = 1$, it also means that $X' + Y' + 1 \geq 2^n$. Thus, $X' + Y' = 2^n - 1$ if and only if $\overline{C_{out}} = 1$ and $C'_{out} = 1$.

The logic dealing with this case is placed at the output of the carry on the second stage (Fig. 6).

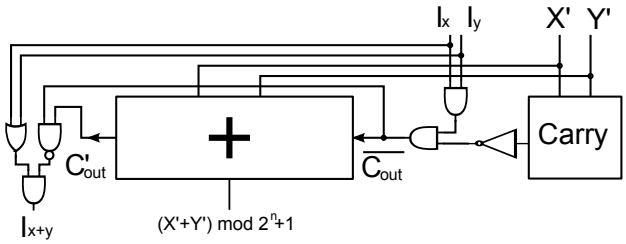

Fig. 6. Diminished-one modulo $2^n + 1$ adder with 0 management

3) Binary modular adder: This modulo $2^n + 1$ adder is based on the adder proposed by Beuchat [17] which is depicted in figure 4.k. Let us note $Z = X + Y = \sum_{i=0}^{n+1} z_i$. His proposition is based on the following equation:

$$|X + Y + 1|_{2^n+1} = z_{n+1}z_{n-1}z_{n-2}\cdots z_0 + (z_{n+1} \text{ nor } z_n) \quad (7)$$

Our modular adder illustrated in figure 7 is adapted from this equation. From equation (7), the input carry of the adder is $(z_{n+1}$ nor $z_n)$. This carry which depends on the two most significant bits of the sum (7) can be early computed as follows.

Let C_{out} be the output carry of the sum of $[X]_{n-1}+[Y]_{n-1}$. We observe that $(z_{n+1}$ nor $z_n) =$ not$(C_{out}$ or x_n or $y_n)$, since z_{n+1} and z_n can be computed by a full adder with inputs x_n, y_n and C_{out}.

According to (7) we then sum this bit, $[X]_{n-1}$, and $[Y]_{n-1}$. This outputs a carry C'_{out} and the $n-1$ least significant bits of the result. The last bit is computed separately. We can see from (4) that the n-th bit of $|X + Y + 1|_{2^n+1}$ is 1 in two cases:

- if $X = Y = 2^n$. In other words, if x_n and y_n are 1, or
- if $X + Y = 2^n - 1$. This arises when $X + Y < 2^n$ and $X + Y + 1 \geq 2^n$. In other words, it holds true when $C_{out} = x_n = y_n = 0$ and $C'_{out} = 1$.

As a consequence, the most significant bit of the result is:

$$x_n \odot y_n \oplus \overline{z_{n+1} \oplus z_n} \odot C'_{out}$$

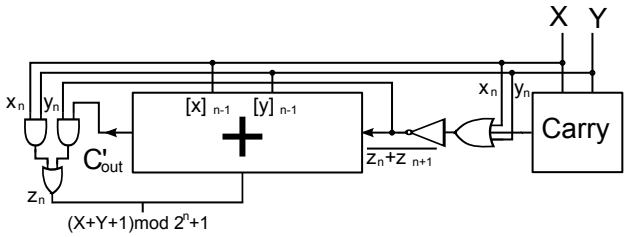

Fig. 7. Binary modulo $2^n + 1$ adder

VI. Comparison with other FPGA modulo $2^n + 1$ adders

In this section we will use the same procedures as in section IV to perform our comparisons.

We compared our adders in diminished-one representation to those Jean-Luc Beuchat proposed in [17], which respectively have parallel (Fig. 4.g) and serial (Fig. 4.h) designs. Note that only our adder without 0 management (Fig. 5) is comparable, since the legacy adders do not manage the 0.

We display in the upper graph of figure 8 the size comparisons and in the lower one, the speed comparisons. The displayed measures correspond to, from left to right, our adder with 0 management, our adder without it, and the adders displayed in figure 4.g and 4.h.

Again, our adder is about the size of its smallest competitor and has about the speed of its fastest competitor. Adding the 0 management creates a negligeable overhead in size, but routing the intermediate carry outside of the carry chain takes its toll on the critical path. This effect however is less present with longer chains (about 25% for $n = 8$, 10% for $n = 40$ and 2% for $n = 80$).

We also compared our binary adder (Fig. 7) to those proposed in [17], which have respectively designs that are parallel a with multiplexer (Fig. 4.i), serial with a multiplexer (4.j), and without multiplexer (4.k). The measures are displayed in that order in figure 9, in the same fashion as before.

We can see here that our adder is clearly faster than its competitors though the parallel architecture has approaching performance. But all three non-parallel architectures have a size of about $1.5 \times n$ whereas the parallel is over $2 \times n$.

https://doi.org/10.1109/ACSSC.2013.6810475
Published in the 2013 Asilomar Conference on Signals, Systems and Computers, doi: 10.1109/ACSSC.2013.6810475

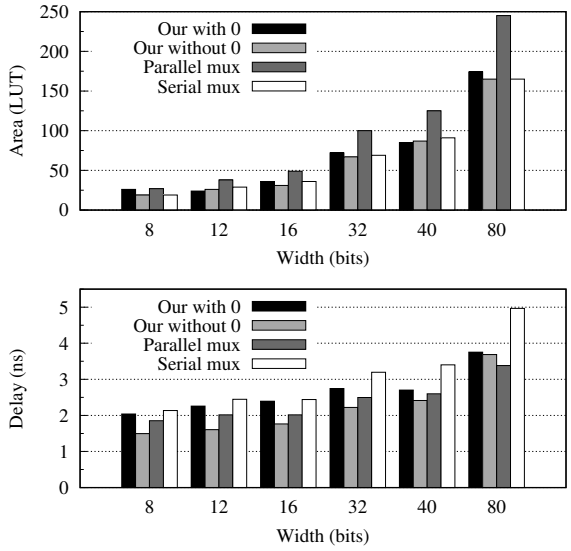

Fig. 8. Comparison of modulo 2^n+1 adders in diminished-one representation

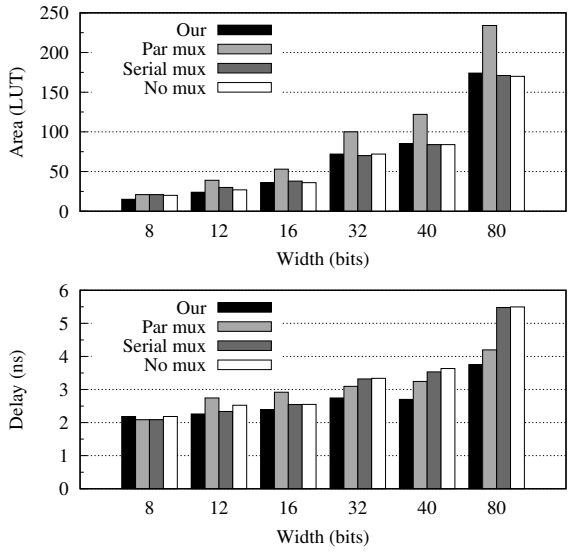

Fig. 9. Comparison of modulo 2^n+1 adders in binary representation

VII. Conclusion

In this paper we presented new fast and small FPGA architectures for modular adders for the moduli 2^n-1 and 2^n+1. Because of the use of the carry chain structure that can be found in all modern FPGAs, they are at least as fast as their fastest competitors and smaller than the smallest.

Our adders are made of two parts, a carry computation and a classical adder. This relies upon the following property of the 2^n-1 and 2^n+1 moduli : the modular addition can be rewritten as the modulo 2^n addition to which we add 0 or 1. These adders show significant improvements in size and space compared to the existing adders for these architectures.

In the case of the 2^n+1 modulus, corner cases that were not tackled by legacy adders have been managed by routing a bit outside the carry chain, thereby sacrificing delay.

References

[1] L. Sousa and S. Antao, "Mrc-based rns reverse converters for the four-moduli sets $\{2^n+1, 2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n+1, 2^n-1, 2^{2n}, 2^{2n+1}-1\}$," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 4, pp. 244–248, Apr. 2012.

[2] P. V. A. Mohan, "Rns-to-binary converter for a new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 9, pp. 775–779, Sep. 2007.

[3] R. Beguenane, J.-L. Beuchat, J.-M. Muller, and S. Simard, *Modular Multiplication of Large Integers on FPGA*. IEEE, 2005.

[4] J.-L. Beuchat and J.-M. Muller, "Automatic Generation of Modular Multipliers for FPGA Applications," *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 1600–1613, Dec. 2008.

[5] G. Sutter, J.-P. Deschamps, and J. L. Imana, "Efficient FPGA Modular Multiplication and Exponentiation Architectures Using Digit Serial Computation," in *2010 International Conference on Field Programmable Logic and Applications*. IEEE, Aug. 2010, pp. 496–501.

[6] J.-L. Beuchat and J.-M. Muller, "Modulo M multiplication-addition: algorithms and FPGA implementation," *Electronics Letters*, vol. 40, no. 11, p. 654, 2004.

[7] S.-h. Lin and M.-h. Sheu, "VLSI Design of Diminished-One Modulo 2^n+1 Adder Using Circular Carry Selection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 9, pp. 897–901, Sep. 2008.

[8] T.-b. Juang, C.-c. Chiu, and M.-y. Tsai, "Improved Area-Efficient Weighted Modulo 2^n+1 Adder Design With Simple Correction Schemes," vol. 57, no. 3, pp. 198–202, Mar. 2010.

[9] H. T. Vergos and C. Efstathiou, "A Unifying Approach for Weighted and Diminished-1 Modulo 2^n+1 Addition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 10, pp. 1041–1045, Oct. 2008.

[10] H. T. Vergos and G. Dimitrakopoulos, "On Modulo 2^n+1 Adder Design," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 173–186, Feb. 2012.

[11] J.-L. Beuchat, E. Okamoto, and T. Yamazaki, "Compact implementations of BLAKE-32 and BLAKE-64 on FPGA," in *2010 International Conference on Field-Programmable Technology*, vol. 4. IEEE, Dec. 2010, pp. 170–177.

[12] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Exploiting fast carry-chains of FPGAs for designing compressor trees," in *2009 International Conference on Field Programmable Logic and Applications*. IEEE, Aug. 2009, pp. 242–249.

[13] H. Parandeh-Afshar, A. Neogy, P. Brisk, and P. Ienne, "Compressor tree synthesis on commercial high-performance FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 4, pp. 1–19, 2011.

[14] F. de Dinechin, H. D. Nguyen, and B. Pasca, "Pipelined FPGA Adders," in *2010 International Conference on Field Programmable Logic and Applications*. IEEE, 2010, pp. 422–427.

[15] H. Parandeh-Afshar, G. Zgheib, P. Brisk, and P. Ienne, "Reducing the pressure on routing resources of FPGAs with generic logic chains," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays - FPGA '11*. ACM Press, 2011, p. 237.

[16] R. Zimmermann, "Efficient VLSI implementation of modulo (2/sup n/±1) addition and multiplication," in *Proceedings 14th IEEE Symposium on Computer Arithmetic*. IEEE Comput. Soc, 1999, pp. 158–167.

[17] J.-L. Beuchat, "Some modular adders and multipliers for field programmable gate arrays," in *Proceedings International Parallel and Distributed Processing Symposium*. IEEE Comput. Soc, 2003, p. 8.

[18] T. Preusser, M. Zabel, and R. Spallek, "Accelerating Computations on FPGA Carry Chains by Operand Compaction," in *IEEE Symposium on Computer Arithmetic (ARITH)*. IEEE, Jul. 2011, pp. 95–102.

[19] H. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2/sup n/+1 adder design," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1389–1399, Dec. 2002.

[20] D. P. Agrawal and T. Rao, "Modulo (2n+1) arithmetic logic," *Electronic Circuits and Systems, IEE Journal on*, vol. 2, no. 6, pp. 186 –188, november 1978.

https://doi.org/10.1109/ACSSC.2013.6810475
Published in the 2013 Asilomar Conference on Signals, Systems and Computers, doi: 10.1109/ACSSC.2013.6810475