IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 2, Ver. I (Mar-Apr. 2014), PP 61-67
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Implementation of Adder for Modulo 2^n+1 Addition

D.sowjanya[(1)],B.Anilkumar[(2)]
M.tech[(1)],Asst'prof[(2)]
Dept of ECE, Dhruva institute of engineering and technology[(1)(2)]

Abstract: *Two architectures for modulo 2^n+1 adders are introduced in this paper. The first one is built around a sparse carry computation unit that computes only some of the carries of the modulo 2^n +1 addition. This sparse approach is enabled by the introduction of the inverted circular idem potency property of the parallel-prefix carry operator and its regularity and area efficiency are further enhanced by the introduction of a new prefix operator. The resulting diminished-1 adders can be implemented in smaller area and consume less power compared to all earlier proposals, while maintaining a high operation speed. The second architecture unifies the design of modulo 2^n±1 adders. It is shown that modulo 2^n+1 adders can be easily derived by straightforward modifications of modulo 2^n - 1 adders with minor hardware overhead.*

I. Introduction:

Arithmetic modulo 2^{n+1} has found applicability in a variety of fields ranging from pseudorandom number generation and cryptography up to convolution computations without round-off errors. Also modulo 2^{n+1} operator are commonly included in residue number system (RNS) applications. The RNS is an arithmetic system which decomposes a number into parts (residues) and performs arithmetic operations in parallel for each residue without the need of carry propagation among them, leading to significant speedup over the corresponding binary operations. RNS is well suited to applications that are rich of addition/subtraction and multiplication operations and has been adopted in the design of digital signal processors FIR filters and communication components offering in several cases apart from enhanced operation speed, low-power characteristics We shall also briery discuss one other unconventional number system that has found some practical use in computer arithmetic; this is the redundant 1 signed-digit number system. We have two objectives in these preliminary discussions. The first is to facilitate a contrast between RNS and commonly used number systems. The second is to to recall a few basic properties of the conventional number systems, as ultimately, it is these that form the basis of implementations of residue arithmetic. The sub sequent introduction to RNS consists of some basic definitions, a discussion of certain desirable features of a residue number system, and a discussion of the basic arithmetic operations.

For reducing the area complexity of the parallel-prefix solutions, select-prefix and circular carry select IEAC adders have been proposed. Unfortunately, both these proposals achieve a smaller operating speed than the parallel-prefix ones. Recently, very fast IEAC adders that use the Ling carry formulation of parallel-prefix addition have appeared in that also suffer from the requirement of a double parallel-prefix computation tree.

II. Existing System

We propose improved area-efficient weighted modulo 2n + 1 adder design using diminished-1 adders with simple correction schemes. This is achieved by subtracting the sum of two (n + 1)-bit input numbers by the constant 2n + 1 and producing carry and sum vectors. The modulo 2n + 1 addition can then be performed using parallel-prefix structure diminished-1 adders by taking in the sum and carry vectors plus the inverted end-around carry with simple correction schemes. The area cost for our proposed adders is lower. In addition, our proposed adders do not require the hardware for zero detection that is needed in diminished-1 modulo 2n + 1 addition.

Exploiting Residue Number System for Power-Efficient Digital Signal Processing in Embedded Processors:

Residue Number System (RNS) has long been touted as a solution for application Digital Signal Processing (DSP) hardware. In this application the inherent parallelism of RNS has been well-known for simultaneously yielding both high-performance and modest power consumption. However, the limitations of its expressiveness in terms of arithmetic operations, together with overheads related to interaction with 2's complement arithmetic, makes programmable processor design that takes advantage of these benefits quite challenging. In this paper we meet this challenge by multi-tier synergistic co-design of compilation techniques, architecture, micro-architecture, as well as hardware components. The net result is an RNS extension to a RISC Processor that others remarkable performance and power improvements simultaneously. With efficient

automatic code generation, we have achieved an average performance improvement of 21% and 51% reduction in functional unit power consumption. For the time, we have exposed application programmers to the substantial benefits of RNS-supported embedded DSP.

Unified Approach to the Design of Modulo-(2n+1) Adders Based on Signed-LSB Representation of Residues: Module of the form 2n ± 1, which greatly simplify certain arithmetic operations in residue number systems (RNS), have been of longstanding interest. A steady stream of published designs for modulo-(2n ± 1) adders has gradually reduced the latency of such adders to a point where they are now quite competitive with ordinary (mod-2n) adders. The next logical step, therefore, is to approach the problem in a unified and systematic manner that does not require each design to be taken up from scratch and to undergo the error-prone and labor-intensive optimization for high speed and low power dissipation. We present a design method that constitutes a move in this direction. More specifically, we devise a new redundant representation of mod-(2n ± 1) residues that allows ordinary fast adders and a small amount of peripheral logic to be employed for modulo-(2n ± 1) addition. Advantages of the building block approach include shorter design time, asier exploration of the design space in terms of area/speed/power tradeoffs, and greater confidence in the correctness of the resulting circuits. Advantages of the unified design include the possibility of fault-tolerant and gracefully degrading RNS processor realizations with fairly low hardware redundancy.

Efficient VLSI Implementation of Modulo (2n+1)Addition and Multiplication, **Zimmerman**,**1999.**

Description:

New VLSI circuit architectures for addition and multiplication modulo (2n+1)are proposed that allow the implementation of highly efficient combinational and pipelined circuits for modular arithmetic. It is shown that the parallel prefix adder architecture is well suited to realize fast end around carry adders used for modulo addition. Existing modulo multiplier architectures are improved for higher speed and regularity. These allow the use of common multiplier speedup techniques like Wall ace tree addition and Booth recoding, resulting in the fastest known modulo multipliers. Finally, a high performance modulo multiplier adder for the IDEA block cipher is presented. The resulting circuits are compared qualitatively and quantitatively in a standard cell technology, with existing solutions and ordinary integer adders and multipliers.

III. ON MODULO $2^n + 1$ ADDER DESIGN

2.1 General

The complexity of a modulo 2n+1 arithmetic unit is determined by the representation chosen for the input operands. Three representations have been considered namely, the normal weighted one, the diminished and the signed-LSB representations. We only consider the first two representations in the following, since the adoption of the signed-LSB representation does not lead to more efficient circuits in delay or area terms. In every case, when performing arithmetic operations modulo 2n+1 the input operands and the results are limited between 0 and 2n.In the normal weighted representation, each operand requires n+1 bits for its representation but only utilizes 2n+1 representations out of the 2n+1 that these can provide. A denser encoding of the input operands and simplified arithmetic operations modulo 2n+1 are offered by thediminished-1 representation. In the diminished-1 representation, A is represented as asA where az is a single bit, often called the zero indication bit, and A is an n-bit vector, often called the number part.

If > 0, then AZ ¼ 0 and A ¼ A+1, whereas for A ¼ 0; AZ ¼ 1, and A ¼ 0. For example, the diminished-1 representation of A ¼ 5 modulo 17 is 001002. Although these architectures are faster than the carry look ahead ones proposed for sufficiently wide operands, they are slower than the corresponding parallel prefix integer adders because of the need for the extra prefix level. it has been shown that the recirculation of the inverted end around carry can be performed within the existing prefix levels, that is, in parallel with the carries computation. In this way, the need of the extra prefix level is canceled and parallel-prefix IEAC adders are derived that can operate as fast as their integer counterparts, that is, they offer a logic depth of log2 n prefix levels.

IV. Methodologies

The key parameters of high-speed digital circuits are the propagation delay and power consumption. The maximum operating frequency of a digital circuit is calculated.

F max = 1/(tp LH + tp HL) …… 1

Where tp LH and tp HL are the propagation delays of the low-to-high and high-to-low transitions of the gates, respectively. The total power consumption of the CMOS digital circuits is determined by the switching and short circuit power. The switching power is linearly proportional to the operating frequency and is given by the sum of switching power at each output node as in

Switching=$\sum_{i=1}^{n}$ fclkCLiVdd2 …… 2

Where n is the number of switching nodes, Fclk is the clock frequency, Clip is the load capacitance at the output node of the ith stage, and Add is the supply voltage. Normally, the short-circuit power occurs in dynamic circuits when there exists direct paths from the supply to ground which is given by

P sc = **I** sc * **V** dd …… 3

Where Isc is the short-circuit current. The analysis shows that the short-circuit power is much higher in E-TSPC logic circuits than n TSPC logic circuits. However, TSPC logic circuits exhibit higher switching power compared to that of E-TSPC logic circuits due to high load capacitance. For the E-TSPC logic circuit, the short-circuit power is the major problem. The E-TSPC circuit has the merit of higher operating frequency than that of the TSPC circuit due to the reduction in load capacitance, but it consumes significantly more power than the TSPC circuit does for a given transistor size. The following analysis s based on the latest design using the popular and low-cost 0.18-microm CMOS process.

V. Performance OnDelay

The time taken to charge and discharge the load capacitance C_L determines the switching speed of the CMOS gate. Rise time is defined during charging time from 10 % to 90% of its steady-state value; this is the same as the fall time. Delay time is defined by the time difference between 50% of charging time and 50% of discharging time. From the equations, in order to improve the individual gate delays, the load impedance C_L is reduced or the current gain of the transistors is increased. Increasing the current gain means higher β, approximately equal to the W/L of the transistor. Therefore, by increasing β, the transistor size will increase, thus affecting the size of the chip.

Modules Name:

- Preprocessing unit
- Carry computation unit
- CS block
- Carry prefix operator
- Carry-select block
- Diminished-1 modulo operation
- New sparse modulo $2^n + 1$ adder

VI. Modules Explanation And Diagram:

Preprocessing unit:

The preprocessing stage computes the carry-generate bits Gi, the carry-propagate bits Pi, and the half-sum bits Hi, for every i , $0 \leq i \leq n-1$. Where . , +, and ⊕ denote logical AND, OR, and exclusive-OR, respectively.

Carry generation bits =>Gi = Ai . Bi

Carry propagate bits => Pi = Ai + Bi

Half sum bits => Hi = Ai ⊕ Bi

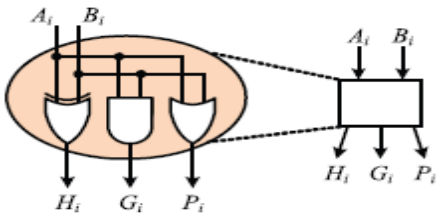


Fig.1. Preprocessing Unit

Carry computation unit:

The second stage of the adder, hereafter called the carry computation unit, computes the carry signals Ci, for $0 \leq i \leq n - 1$ using the carry generate and carry propagate bits Gi and Pi. The third stage computes the sum bits according to

Si = Hi ⊕Ci -1

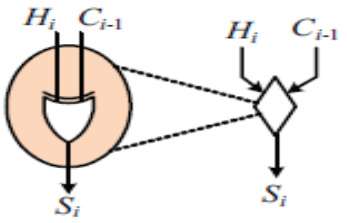


Fig.2. Carry computation unit

Carry computation is transformed into a parallel prefix problem using the "o" operator, which associates pairs of generate and propagate signals and was defined as

(G,P)o(G',P')=(G+P.G',P.P')

In a series of associations of consecutive generate and propagate pairs (G; P) can be represented as

$(G_{k:j}: P_{k:j}) = (Gk ,Pk)o(Gk-1,Pk-1)o….o(Gj , Pj)$

Where k>j

Since every carry $C_i = G_{i:0}$, a number of algorithms have been introduced for computing all the carries using only "o" operators.

VII. Carry-Select Block:

The design of sparse adders relies on the use of a sparse parallel-prefix carry computation unit and carry-select (CS) blocks. Only the carries at the boundaries of the carry-select blocks are computed, saving considerable amount of area in the carry-computation unit. The carry select block computes two sets of sum bits corresponding to the two possible values of the incoming carry. When the actual carry is computed, it selects the correct sum without any delay overhead.

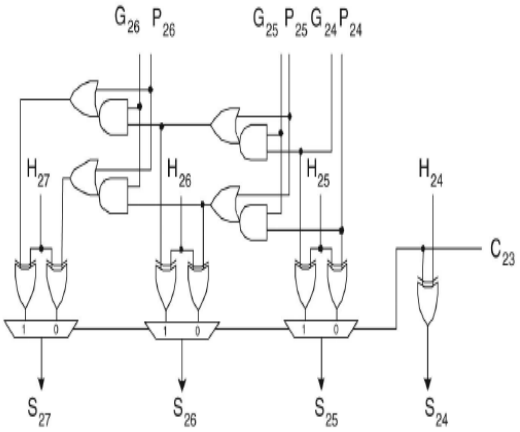


Fig.3. Logic-level implementation of a 4-bit carry-select block

VIII. Parallel Prefix Adder:

Parallel prefix adders are constructed out of fundamental carry operators denoted by ¢ as follows: (G", P") ¢ (G', P') = (G"+G'·P", P'·P"), where P" and P' indicate the propagations, G" and G' indicate the generations. The fundamental carry operator is represented as Figure 4.

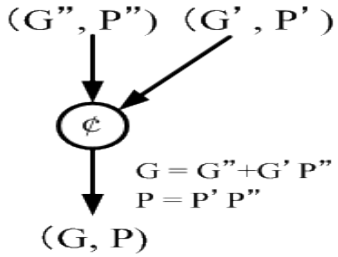


Figure 4. Carry operator

A parallel prefix adder can be represented as a parallel prefix graph consisting of carry operator nodes. Figure 5 is the parallel prefix graph of a Ladner-Fischer adder. This adder structure has minimum logic depth, but has large fan-out requirement up to n/2.

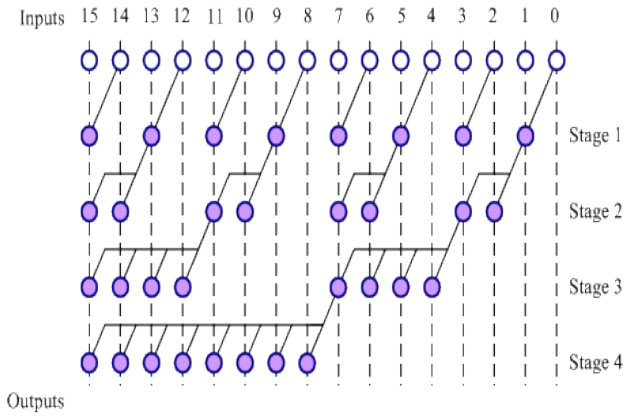


Figure 5. 16-bit Ladner-Fischer adder

IX. Diminished-1 Modulo Operation (Sparse Carry Computation):

The sparse version of the parallel-prefix adders introduced in this paper alleviates a lot the regularity and the area-overhead problem, as it can be verified from Fig 8, there is still a lot
of space for improvement. In the following, we attack this problem by introducing a new prefix operator and an even simpler parallel-prefix carry computation unit. The new technique will be
Presented via an example. Let us consider the design of a sparse-4 diminished-1 modulo 2^{16}+1 adder. In this case, we need a carry computation unit that implements the following prefix equations:

$$C_{15}^{+} \leftrightarrow \overline{(G_{15:0}, P_{15:0})},$$
$$C_{11}^{+} \leftrightarrow (G_{11:0}, P_{11:0}) \circ \overline{(G_{15:12}, P_{15:12})},$$
$$C_{7}^{+} \leftrightarrow (G_{7:0}, P_{7:0}) \circ \overline{(G_{15:8}, P_{15:8})},$$
$$C_{3}^{+} \leftrightarrow (G_{3:0}, P_{3:0}) \circ \overline{(G_{15:4}, P_{15:4})}.$$

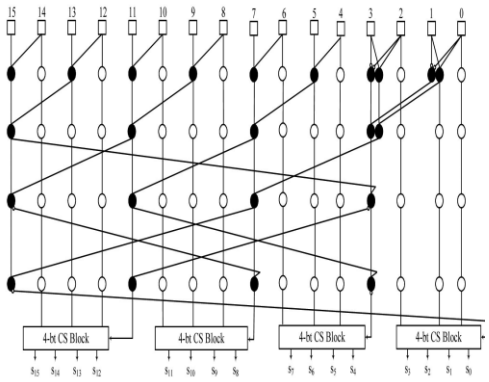


Fig 6 Modulo 2^{16}+1 diminished adders using a sparse carry computation unit.

New Sparse Modulo 2^n + 1 Adder:

Fig. 9 presents the resulting architecture for a diminished-1 modulo 2^{16} + 1 adder, in which two gray operators are used. The top one which resides at prefix level 3, accepts a feedback signal and therefore has its T_V^3 V input tied to zero. This operator is used to compute (G3:0; P3:0) o($G_{15:12}$,$P_{15:12}$), which is necessary for the computation of both C_3^+ and C_{11}^+. Its vertical successor is also replaced by a gray operator that computes the final

$$C_{3}^{+} \leftrightarrow (G_{3:0}, P_{3:0}) \circ \overline{(G_{15:12}, P_{15:12}) \circ (G_{11:4}, P_{11:4})}.$$

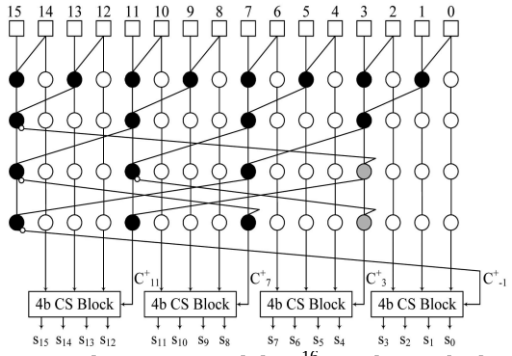


Fig. 7.Proposed sparse-4 modulo 2^{16} + 1 diminished-1 adder.

X. Parallel Prefix Adder

3.1 General

Parallel Prefix Adder (PPA) is very useful in today's world of technology because of its implementation in Very Large Scale Integration (VLSI) chips. The VLSI chips rely heavily on fast and reliable arithmetic computation. These contributions can be provided by PPA. There are many types of PPA such as Brent Kung, Kogge Stone, Ladner Fisher, Hans Carlson and Knowles . For the purpose of this research, only Brent Kung and Kogge Stone adders will be investigated. Fig. 8 shows the structured diagram of a PPA. PPA can be divided into three main parts, namely the pre-processing, carry graph and post-processing. The pre-processing part will generate the propagate (*p*) and generate (*g*) bits. The acquirement of the PPA carry bit is differentiates PPA from other type of adders. It is a parallel form of obtaining the carry bit that makes it performs addition arithmetic faster.

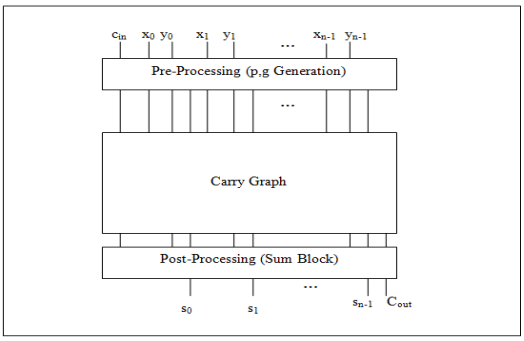


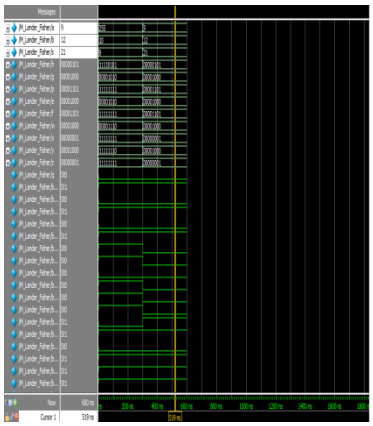
Figure 8. General Parallel Prefix Adder

XI. Parallel Prefix Networks

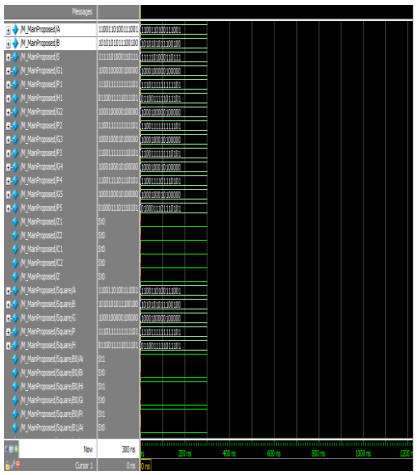
Parallel prefix network are distinguished by the arrangement of prefix cells. Fig. 2 shows six such networks for *N*=16. The upper box performs the precipitation and the lower box performs the post computation. In the middle, black cells, gray cells, and white buffers comprise the prefix network. Black cells perform the full prefix operation, as given in EQ (3). In certain cases, only part of the intermediate variable is required. For example, in many adder cells, only the *Gi*:0 signal is required, and the *Pi*:0 signal may be discarded. Such gray cells have lower input capacitance. White buffers are used to reduce the loading of later non-critical stages on the critical path. The span of bits covered by each cell output appears near the output. The critical path is indicated with a heavy line. The prefix graphs illustrate the tradeoffs in each network between number of logic levels, fanout, and horizontal wiring tracks. All three of these tradeoffs impact latency; Huang and Ercegovac showed that networks with large number of wiring tracks increase the wiring capacitance because the tracks are packed on a tight pitch to achieve reasonable area. Observe that the Brent-Kung and Han-Carlson never have more than one black or gray cell in each pair of bits on any given row. This suggests that the data path layout may use half as many columns, saving area and wire length.

XII. Results

Lander Fisher:



New Sparse Modulo $2^n + 1$ Adder:



XIII. Conclusions

Efficient modulo 2n+1 adder are appreciated in a variety of computer applications including all RNS implementations. In this paper, two contributions are offered to the modulo 2n+1 addition problem. A novel architecture has been proposed that uses a sparse totally regular parallel-prefix carry computation unit. This architecture was derived by proving the inverted circular idem potency property of the parallel-prefix carry operator in modulo 2n+1 addition and by introducing a new prefix operator that eliminates the need for a double computation tree in the earlier fastest proposals. The experimental results indicate that the proposed architecture heavily outperforms the earlier solutions in implementation area and power consumption, while offering a high execution rate. The modulo 2n+1 addition problem was also shown to be related to the modulo 2n+1 addition problem. The unifying theory presented in this paper revealed that a simple post processing stage composed of an XOR gate for each output bit needs to be added to a modulo 2n+1 adder for attaining a modulo 2n+1 added. As a result, every architecture that has been and more importantly that will be proposed for designing modulo 2n+1 adders, can be reused for the design of modulo 2n+1 adders.

References

[1] X. Lai and J.L. Massey, "A Proposal for a New Block Encryption Standard," EUROCRYPT, D.W. Davies, ed., vol. 547, pp. 389-404, Springer, 1991.
[2] R. Zimmermann et al., "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm," IEEE J. Solid-State Circuits, vol. 29, no. 3, pp. 303-307, Mar. 1994.
[3] H. Nozaki et al., "Implementation of RSA Algorithm Based on RNS Montgomery Multiplication," Proc. Third Int'l Workshop Cryptographic Hardware and Embedded Systems, pp. 364-376, 2001.
[4] Y. Marikina, H. Hamada, and K. Nagoya's, "Hardware Realization of High Speed Butterfly for the Maximal Length Fermat Number Transform," Trans. IECE, vol. J66-D, pp. 81-88,1983.
[5] M. Beanies, S.S. Dlay, and A.G.J. Holt, "CMOS VLSI Design of a High-Speed Fermat Number Transform Based Convolve/Correlate Using Three-Input Adders," Proc. IEE, vol. 138, no. 2,pp. 182-190, Apr. 1991.