ISSN (Online): 2230-7893
www.IJCEM.org

Area-Power Efficient Generic Modulo Adder

I.B.K.Raju[1], P. Rajesh Kumar[2], S. Ramani Yadav[3]

**[1,3]B.V. Raju Institute of Technology, CVD, ECE Department,
Narsapur, Medak, TELANGANA**

**[2]P.V.P Siddhartha Institute of Technology,
Vijayawada, AP**

Abstract

Modular adder is a crucial component which is typically employed in Forward, Reverse and channels in a Residue Number System. In This Paper we proposed a Novel generic modulo adder architecture which is based on Look ahead Carry logic by which hardware sharing is exploited and eliminated the re-computation of carry in the final stage. VLSI Implementation results using 180 nm standard cell Technology shows that the hardware requirement and power dissipation in proposed architecture is superior to other reported architectures.

Keywords: *Carry lookahead adder, computer arithmetic, Residue number system, Modulo adder, VLSI.*

1. Introduction

Residue number system (RNS) is an ancient numerical representation system. RNS is a non-weighted numerical representation system and has carry-free property in multiplication and addition operations between residue digits. The inbuilt carry-free operations, parallelism, and high reliability of the residue number system (RNS) have made it an important for high-performance and fault-tolerant DSP systems [1], [2]. The RNS is similar to a fixed-radix number system (e.g. binary, decimal, etc.), in that it is completely described by a base.In recent years, it has been received intensive study in the very large scale integration circuits (VLSI) design for digital signal processing (DSP) systems with high speed and low power consumption. Modular adder is one of the key components for any RNS-based systems. While RNS can done addition and multiplication performance better than the other arithmetic operations like division, magnitude comparison, and sign detection are more difficult, when compared with their counterparts in the conventional binary number system.

A Residue Number System (RNS) is characterized by a base which is an N-tuple of relatively co-prime integers (m_{N-1}, m_{N-2} … m_0) and each m_i (i = 0, 1, 2 … N-1) is called a modulus [1]. For a given base, an integer X is represented by an N-tuple word, $\{x_0, x_1, x_2 \ldots x_{N-1}\}$, where $x_i = |X|_{mi}$, i.e., it is the non-negative remainder when dividing X by the modulus m_i.

The RNS dynamic range is represented by the product of all k relatively prime moduli ($M = m_0 \times m_1 \ldots \times m_{k-1}$). It denotes the interval over which every integer can be represented by the system without having two numbers with the same representation, viz. [0, M-1]. Or any other interval of M consecutive integers.

Design of modulo adders is classified in to two categories: Generic modulo adder and special modulo adder. Hardware implementations for Generic modulo adders are based on look-up tables, pure combinational logic, or a combination of both [2], [3]. Special modulo adder can be designed for specific module set like $2^n \pm 1$.Architectural simplicity and the best performance have achieved for this class of modulo adder when compared with generic modulo adder. [4], [5].

Taylor [6] introduced various methods based on conventional binary adders. Bayoumi and Jullien [3] explained three different categories of modular adders. These were based on binary adders, lookup tables (LUTs), and an amalgamation of the two. Dugdale [7] presented the implementation of sequential modular adders using two cycles of binary addition with feedback to perform the second addition. Piestrak [8] described a modular adder design as a final stage in a residue-to-binary converter. A carry-save adder (CSA) and two binary adders are used in this method. Hiasat [9] proposed an adder, which was related to the Piestrak [8] approach and it uses CSA and carry-propagate-addition (CPA) techniques. More recently, Patel [10] proposed a new contribution to the generic modular adder design space that is based on the ELM addition algorithm, which has shown to produce adders with a reduced power-delay-area.

In this paper, a new algorithm and corresponding Hardware realization for generic modular addition is described. Sharing of hardware between components in the architecture speeds up the modular addition operation. Due to this, there is a reduction in area there by power dissipation.

The organization of the remainder of the paper is as follows. Section II Describes the existing modular adders. Next, the proposed modular adder algorithm and architecture is described. Next section shows the comparative analysis of the modular adders.

2. Existing Modulo Adders

Modular adders are built using similar principles as the traditional binary adders. All optimization techniques found in binary addition can be utilized to construct modular adders.

If we have two integers A and B of modulo m, then their addition is expressed as sum of (A+B) mod m. The addition operation can be expressed as below

$$|A+B|_m = \begin{cases} A+B & if\ A+B<m \\ A+B-m & else \end{cases} \quad (1)$$

The existing combinational modualr adders are having varying tradeoffs in design complexity. At first glance, Bayoumi *et al.*'s purely combinatorial adder [3] gives the worst tradeoffs. Because the adder design is both slow and large. The adder proposed by Piestrak [8] provides high speed but the design complexity is high. The adder introduced by Hiasat [9] is having tradeoff between speed and area in comparison with Piestrak's adder. The sum and carry bits are computed after the MUX module in this architecture.The drawback of this adder is, it computes duplicate carries. It can be eliminated in the adder which was proposed by Patel [10]. In the most recent contribution from the literature, Patel [10] has proposed an adder which trades speed for area in comparison with Hiasat's adder. The main drawback of Patel's adder is the computation of sum and carry bits before the multiplexer module,which will leads to area cost.This problem is addressed in proposed adder by shifting the final sumation after the multiplexer opereation.

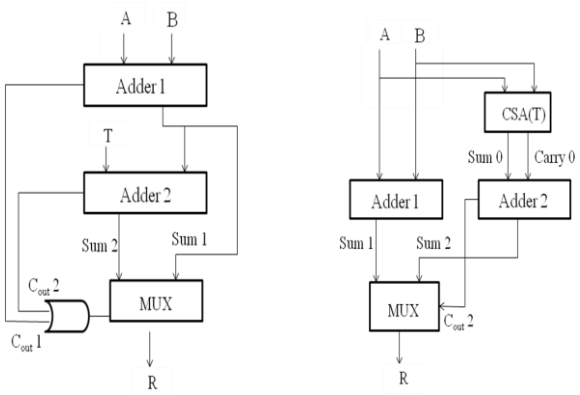
The architecture of the adder described by Bayoumi *et al*. is shown in Fig. 1.(a). It utilizes two CPA binary adders which are connected in series, an OR gate, and a multiplexer. *Adder 1* computes the sum A + B while *Adder 2* computes $(A + B)_{n-1:0}$ +T, where $(S)_{n-1:0}$ represeents the *n* LSBs of S. If the carry from *Adder 2* is 1, the multiplexer chooses *sum 2*, otherwise *sum 1* is chosen.

Piestrak's modular adder architecture is shown in Fig. 1.(b) It is based on using a CSA, two CPA adders and an MUX. The CSA evaluates A + B + T as a sum S and a carry C. Note that CSA (T) is a simplified CSA architecture, where the extent of the simplification is dependent on the binary form of T. The two CPAs operate in parallel. *Adder1* evaluates A + B, while *Adder 2* evaluates S + C. The multiplexer then selects one of the CPA adder outputs. Since it uses a CSA and two CPA adders, the Piestrak adder [8] requires more hardware than the adders proposed in [6] and [7]. The lengthy interconnections of the CSA to the two CPA adders also increase the wire delay.

The adder proposed by Hiasat is shown in Fig.2.(a). Hiasat first reduces the three-operand addition *A* + *B* + *T* to a *sum* (x) and *carry* (y) using a simplified CSA architecture called the SAC (*T*) unit (similar to Piestrak's CSA (*T*)). The *A* + *B* addition is also transformed to a *sum* (*X)* and *carry* (*Y*) with *T* set to 0 in the SAC (*T*) unit. Thereafter, a carry propoagate and generate (CPG) unit is used to compute the propagate and generate terms for the *A* + *B* addition (*PS, G*) and the *A* + *B* + *T* addition (*ps, g*). Note that propagate terms are used to be computed using XOR gates in this adder.The output carry is then calculated for the *A* + *B* + *T* addition.

For *n* – bit modular addition, a maximum of 2(*n* -1)2 × 1 multiplexers are then used to select the appropriate propagate and generate terms based on the computed carry, after which the CLAS unit uses CPA techniques to compute the correct modular sum using the selected propogate/generate pairs.

The proposal by Patel is shown in Fig.2(b). This adder consists of three basic units. These units are: the Simplified Carry Save Adder (SCSA(T)) unit, the Double Addition with Shared Hardware using Carry Propagate Addition (DASH-CPA) unit, and the multiplexer unit. The SCSA (T) unit has the operands *A* and *B* as input and perform carry-save addition of *A* + *B* and *A* + *B* + *T*. The main module of this adder is DASH-CPA unit. It is used to compute both sums and carrys of *A* + *B* and *A* + *B* + *T* in parallel. The carry out from the result *A* + *B* + *T*, c^1_{out}, is used to select the correct modulo sum.



(a) (b)

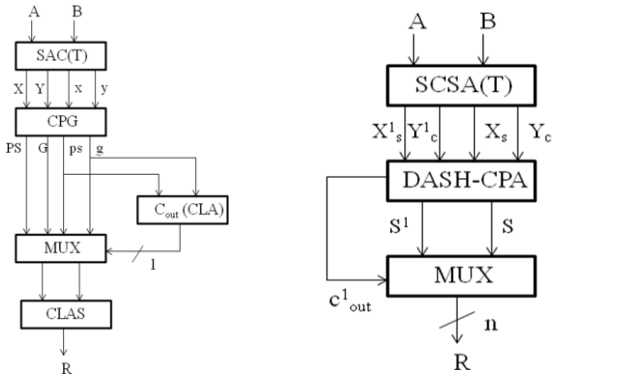
Fig.1.Modulo adders Proposed by (a) Bayoumi and Jullien [3], (b) Piestrak [8]

The DASH-CPA unit is implemented using ELM [11], Ladner-Fischer (LF) [12] and Kogge-Stone (KS) [13] parallel-prefix adders. Among these designs, the ELM based adder is efficiencient in delay, area, and power[14]. The KS and LF adders were considered over other existing prefix adders because they offer delay and area efficiency, respectively. The MUX unit is used to select the sum bits generated in the DASH-CPA unit.

3. Proposed Modulo-m adder Architecture

In this section, the architecture for the proposed modular adder is described. The proposed modular adder is shown in Fig.3. This adder consists of four basic units. These

units are: the Partial-Sum-Carry (PSC) unit, Propagate and Carry (PC) unit, Multiplexing (MUX) Unit, Sum Calculation Unit (SCU).



(a) (b)

Fig.2. Modulo adders proposed by (a) Hiasat [9], (b) Patel [10]

3.1 Partial Sum and Carry (PSC) Unit

The PSC module has the operands A and B as input and generates the carry-save representation of A+B and A+B+T. The structure is a simplified version of the conventional CSA. Here T is the two's complement of m (i.e. –m). This adder is designed for m=29. Hence T=3= $(00011)_2$. The value of T is inherent in the architecture. If $T_i = 0$, then we have to use HA cell, otherwise EHA cell is used.

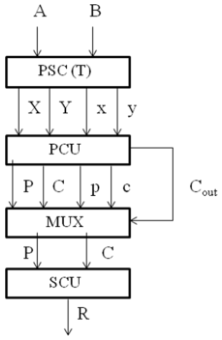


Fig.3. Block diagram of proposed modulo adder

Let *S* and *C* bits denotes the sum and carry outputs of a full adder(FA), then

$$S_i = a_i \oplus b_i \oplus c_i \quad (2)$$

$$C_{i+1} = a_i b_i + a_i c_i + b_i c_i \quad (3)$$

where a_i, b_i, and c_i are the inputs of fulladder, while $\oplus$ represents an Exclusive-OR operation. If the input bit c_i assumes the value 0 then, writing S_i as X_i and C_{i+1} as Y_{i+1}, Eqns (2), (3) can be reformulate as

$$X_i = a_i \oplus b_i \quad (4)$$

$$Y_{i+1} = a_i b_i \quad (5)$$

The last two equations describe the structure of a Half-Adder (HA) cell, shown in Fig.4(a).Similarly, if c_i is 1, then Eqns (4) and (5) can be rewritten as

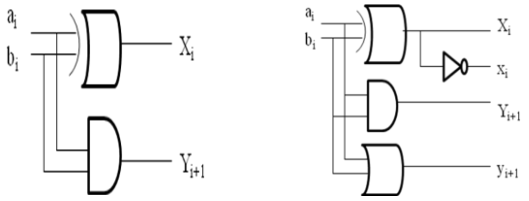
$$x_i = \overline{a_i \oplus b_i} \quad (6)$$

$$y_{i+1} = a_i + b_i \quad (7)$$

Where $\overline{a_i \oplus b_i}$ denotes the Exclusive-NOR operation.

Using Eq.(4) through Eq. (7), a circuit that can produce the sum and carry for both cases (i.e., for c_i = 0,1), and called as Extended-Half-Adder (EHA) cell, is the one shown in Fig.4(b).

The first stage of the proposed adder is shown in Fig.5 for the *m* = 29.



(a) (b)

Fig.4. (a) Half-adder Cell, (b) Extended-Half-Adder Cell

3.2 Propagation and Carry (PC) Unit

This unit has two sub modules. One is Generate-Propagate (GP) unit, and other is Macro-LAC (M-LAC).

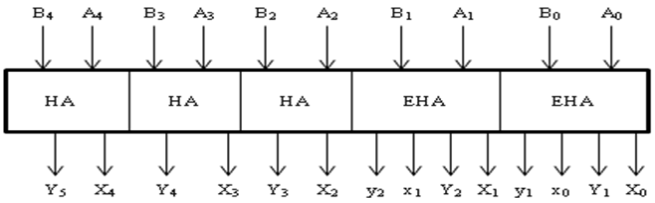


Fig.5. PSC unit of proposed adder for *m* = 29

The GP unit receives the bits of X and Y. Carry is generated locally if both input bits X_i and Y_i are ones and it is expressed by the generate equation $G_i = X_i Y_i$. A carry is propogated if either X_i or Y_i is one and the propogate equation is $P_i = X_i \oplus Y_i$. Fig.4(a) shows the Generate-Propagate (GP) cell that is used in evaluating P_i and G_i (GP cell is identical to a HA cell). Therfore, the carry propogate (P) and generate (G) output vectors of the GP unit.

Similarly, the GP unit also receives the bits of x, and y, where $g_i = x_i y_i$ and $p_i = x_i \oplus y_i$. The GP unit of proposed adder is shown Fig.6.

The M-LAC cell receives the bits of P, G, p, and g. These bits are used to calculate the carry bits only. It outputs the one bit variable C_{out}.

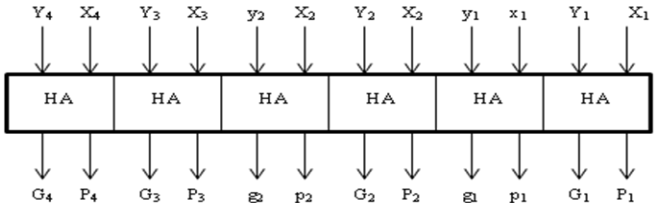
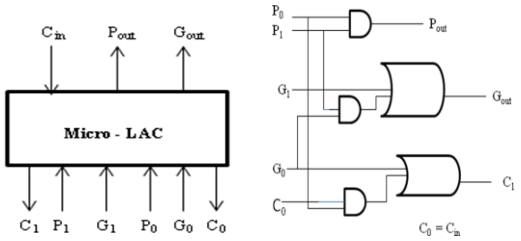


Fig.6. GP unit of Proposed adder for m = 29

This M-LAC cell designed by using several micro-LACs. The micro-LAC cell is designed based on the equations given below

$$C_0 = C_{in}$$
$$P_{out} = P_0 P_1$$
$$G_{out} = G_1 + P_1 G_0$$
$$C_1 = G_0 + P_0 C_0$$

The basic block diagram of micro-LAC is shown in Fig.7. The general block diagram of the M-LAC adder (N=8) is shown in Fig.8.



(a) (b)

Fig.7. (a) Block diagram of Micro – LAC, (b) internal structure of Micro-LAC

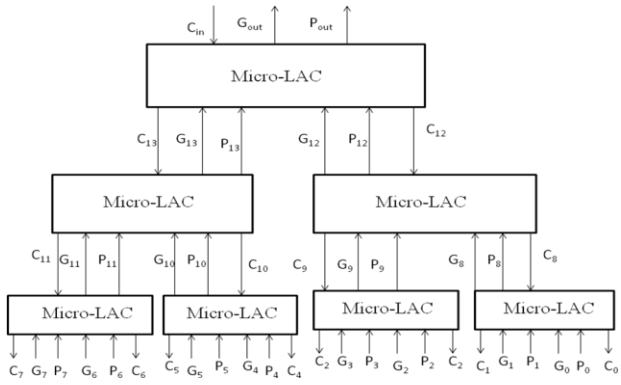


Fig.8. Block diagram of the ordinary M-LAC adder for N=8.

For N-bit M-LAC adder, there are $\log_2 N$ rows of Micro-LAC circuits, each row with half the number of Micro–LAC circuits as the row above it. The worst case (longest) delay through the tree starts from a, b inputs going down the tree branches through generate and propagate signals to the root of the tree, and then coming back up the tree branches through the carry signals. There are $\log_2 N$ in Micro - LAC's along each branch so that the worst case delay is proportional to log N. Thus, for the N-bit adder, the worst case delay goes through 2 $\log_2 N$ +1 Micro-LAC's which is much better than going through N pieces of the linear carry chain.

But in this proposed adder m = 29 (i.e. n = 5) is considered. Hence one micro-LAC cell is duplicated in this architecture. Due to this hardware sharing, area can be reduced. In this adder all carry bits are computed at same time.

The detailed block diagram of M-LAC unit for m = 29 is shown in Fig. 9.

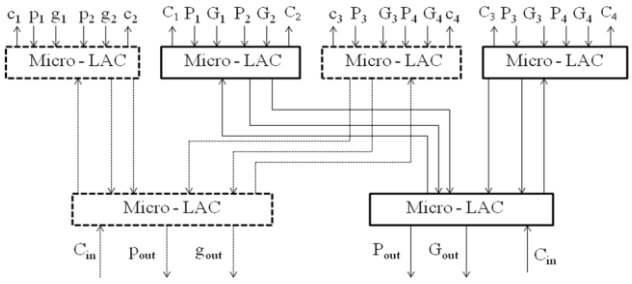


Fig.9 Block diagram of M-LAC cell for m =29

3.3 Multiplexing (MUX) Unit

The multiplexing unit selects the vectors of P and C of (A + B) if C_{out} = 0. Otherwise, it selects the vectors of p and c of (A + B + C). This module consists of n 2 × 1 multiplexers. The C_{out} signal is used as selection line for all multiplexers.

3.4 Sum Computing (SC) Unit

It performs the summation of bits which are coming from the MUX outputs to produce the output R. The result is computed using the equation

$$r_i = P_i \oplus C_i \qquad (8)$$

4. Results and Comparative Analysis

In this paper, the proposed adder is comapared with the existing modular adders. The widely used unit-gate model is used to estimate the delay and area of the architecture [15]. In this model, each two input gate (e.g., AND or OR) counts as one gate in both area and delay. An XOR/XNOR gate counts as two gates in area and delay. Table 1 shows the unit-gate delay values for modulo adder elements.

ISSN (Online): 2230-7893
www.IJCEM.org

Table 1: Unit-gate characterization of Modulo adder elements

Elements	***Delay***
AND, OR	1
XOR/XNOR	2
2x1 MUX	2
HA	2
EHA	2
Micro-LAC	2

All modulo adders are described in verilog,simulated in Xilinx ISE14.4 and implemented using 180nm standard cell technology libraries of TSMC using Cadence RTL.

Table 2: Area, power, delay results for proposed and existing modular adders for modulus m=29, n=5

Type of Adder	**Area (μm^2)**	**Power (µW)**	**Delay (pS)**	**A × D ($\times 10^{-6}\mu m^2$ Sec)**	**A × P (μm^2W)**	**D × P ($\times 10^{-12}$W Sec)**	**A × D × P ($\times 10^{-9}\ \mu m^2$W Sec)**
Bayoumi [3]	2525	2059	1500	3.78	5.2	3.08	7.80
Piestrak [8]	3453	2228	1272	4.39	7.69	2.83	9.78
Hiasat [9]	2521	2056	970	2.44	5.18	1.99	5.02
ELMMA [10]	2571	2472	900	2.31	6.35	2.22	5.72
KSPPMA [10]	2944	2373	875	2.57	6.98	2.07	6.11
LFPPMA [10]	2768	2327	885	2.44	6.44	2.05	5.70
Proposed adder	2442	1922	990	2.41	4.69	1.90	4.64

The practical analysis of the proposed and existing adders is shown in Table 2 for modulo m=29 and n=5. In Table 2, the results show that, on average for the proposed adder is efficient in terms of area and power when compared with the other modulo adders. It is worth noting that KSPPMA's [10] adder is approximately 10% faster than the proposed adder on average, but at an expense of at least 20% in both area and power complexities.

In Table 3, the results show that the area and power savings for the proposed adder with respective to other modulo adders.

Table 3: Area and power savings achieved with proposed adder

Proposed Adder vs.	***Area Savings (%)***	***Power Savings (%)***
Bayoumi [3]	3	7
Piestrak [8]	29	14
Hiasat [9]	3	7
ELMMA [10]	5	23
LFPPMA [10]	12	17
KSPPMA [10]	19	19

Table 4: Average savings resulting from the proposed adder in the $A \times P$, $D \times P$, $A \times D \times P$ complexity evaluation measures

Proposed Adder vs.	***$A \times P$ Savings (%)***	***$D \times P$ Savings (%)***	***$A \times D \times P$ Savings (%)***
Bayoumi [3]	10	38	40
Piestrak [8]	39	33	53
Hiasat [9]	10	5	8
ELMMA [10]	26	14	19
LFPPMA [10]	27	7	19
KSPPMA [10]	33	8	24

Table 4 shows the savings of the proposed adder with existng modulo adders in area-power product ($A \times P$), area-delay-power product ($A \times D \times P$), and delay-power product($D \times P$) complexities. When compared with bayoumi's adder[3], Proposed adder provides savings of approximately 10%, 40% and 38% in $A \times P$, $A \times D \times P$, and $D \times P$, respectively. When compared with Hiasat [9], Proposed adder provides

savings of approximately 10%, 5% and 8% in $A \times P$, $A \times D \times P$, and $D \times P$, respectively. Where the cost functions $A \times P$, $A \times D \times P$, and $D \times P$ are considered, the proposed adder provides the greatest efficiency, with marginal improvements over Hiasat's adder.

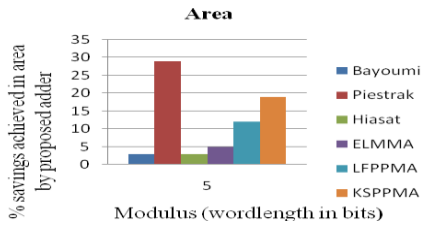


Fig. 10. % Savings achieved in the Area by proposed adder

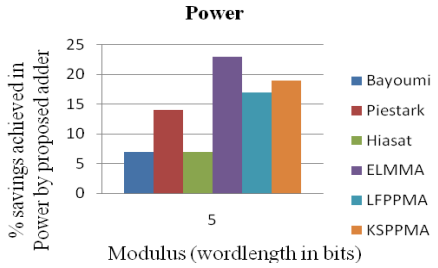


Fig. 11. % Savings achieved in Power by proposed adder

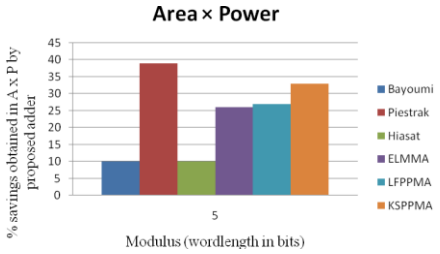


Fig. 12. % Savings achieved in the A × P product by proposed adder

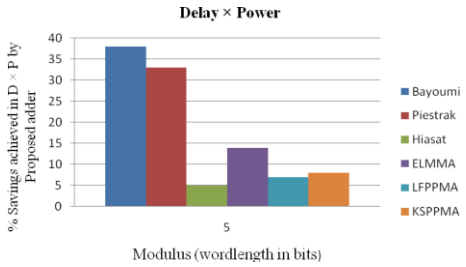


Fig. 13. % Savings achieved in the D × P product by proposed adder

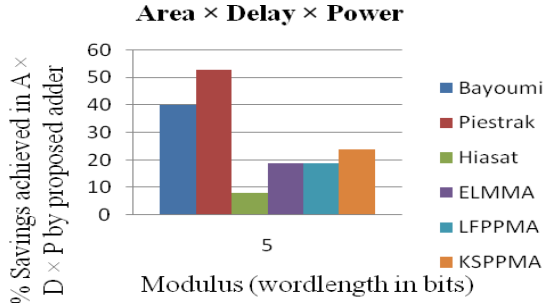


Fig. 14. % Savings achieved in the A × D × P product by proposed adder

Fig.10-14 illustrating the results of % savings achieved with the proposed adder in terms of area and power and in cases such as, $A \times P$, $D \times P$, $A \times D \times P$.

Conclusion

In this paper, a new architecture for a modular adder is presented. This new modulo adder offers higher speed and a smaller area. The proposed adder is based on Look ahead Carry logic and, by which hardware sharing is exploited and eliminated the re-computation of carry in the final stage. As a result; it has on an average 12% and 15% efficient with respective to area and power. VLSI implementations show that the proposed adder design is superior under criteria such as *area × power*, *delay × power and area × delay × power*, when compared to state of art modular adder architectures.

Acknowledgement

The authors would like to thank the management of Dr. B.V Raju Institute of Technology, Narsapur for providing Cadence EDA tools and support.

References

[1] N. Szabo and R. tanaka, *Residue Arithmetic and its Applications to Computer Technology*. New York: McGraw-Hill, 1967.

[2] *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, M. Soderstand, M.AW. Jenkins, G. Jullien, and F. Taylor, eds. New York: IEEE Press, 1986.

[3] M. Bayoumi and G. Jullien, "A VLSI Implementation of Residue Adders," *IEEE Trans. Circuits and Systems*, vol. 34, pp. 284-288, Mar. 1987.

[4] G. Dimitrakopoulos, "A family of parallel-prefix modulo 2n-1 adders," in *Proc. IEEE Int. Conf. Application-Specific Syst., Arch., Processors (ASSAP 2003)*, The Hauge, Netherlands, Jun. 2003, pp. 315-325.

[5] R.A Patel., "Power-delay-area efficient modulo $2^n + 1$ adder architecture for RNS," *Electron. Lett.*, vol. 41, no. 5, pp. 231-2, Mar. 2005.

[6] F. Taylor, "A single modulus complex alu for signal processing," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 33, no. 5, pp. 1302-15, Oct. 1985.

[7] M. Dugdale, "VLSI implementation of residue adders based on binary adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 5, pp. 325-9, May 1992.

[8] S. J Piestrak, "Design of high-speed residue-to-binary number system converter based on chinese remainder theoram," in *Proc. IEEE Int. Conf. Comput. Design (ICCD 1994)*, Cambridge, MA, Oct. 1994, pp. 508-511.

[9] A. A. Hiasat, "High-speed and reduced-area modular adder structures for RNS," *IEEE Trans. Cmput.*, vol. 51, no. 1, pp. 84-9, Jan. 2002.

[10] R. A. Patel, M. Benaissa, N. Powell and Said Bousskata, "Novel Power-Delay-Area-Efiicient Approach to Generic Modular Addition," *IEEE Trans. Circuits and Systems I*, vol. 54, no. 6, pp. 1279-1292, 2007.

[11] T. P. Kelliher, "ELM-a fast addition algorithm discovered by a program," *IEEE Trans. Comput.* vol. 41, no. 9, pp. 1181-1184, Sep. 1992.

[12] R. E. Ladner and M. J. Fischer, " Parallel prefix computation," *J.Assoc. Computiing Machinery*, vol. 27, no.4, pp. 831-8, Oct. 1980.

[13] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recuurrence equations," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 786-92, Aug. 1973.

[14] C. Nagendra , "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II. Analog Digit. Signal Process.*, vol. 43, no. 10, pp, 689-702, Oct. 1996.

[15] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1163-1170, Oct. 1993.