A COMPARATIVE STUDY OF ADDER ARCHITECTURES FOR 3X MULTIPLE GENERATION IN RADIX-8 BOOTH ENCODING

Nirmaladevi. R and Seshasayanan. R
College of Engineering, Anna University, Chennai-600025, India
nirmala.sashi@gmail.com, +91-9941455004

Abstract: *Several energy efficient adder designs have been proposed over the years to speed up the multiplication in digital signal processors. The selection of the specific adder improves the performance of the multiplier design. In this paper, we propose a comparative study of different adders for the generation of 3X term (Hard multiple) in radix-8 booth encoding. The focus of this paper is to explore the design aspects of various adder architectures for the implementation 3X term. We consider Ripple carry adder (RCA), Carry look ahead adder(CLA), Ling adder, Parallel prefix Ling adder, Conditional carry adder(CCA), Kogge-Stone(KS) adder and Sklansky(SK) adder for our comparison. These adders are simulated using ISE simulator with the VHDL structural coding. Cadence RTL complier with TSMC library 180nm is used to synthesize and analyze the cell area, power and delay. The experimental results imply that the Sklansky adder shows the improved performance over other architectures and saves energy ~50% for 64 bits implementation of 3X term.*

Key words: *Radix-8 Booth encoding, Hard multiple, Carry propagate adders, Ling adder, Parallel prefix adder.*

1. Introduction

Faster arithmetic modules are essential for modern DSP systems, as the maximum operating speed of these processors depend largely on these modules. Addition is the fundamental arithmetic operation and the other operations like subtraction, multiplication and division can be derived from addition. Hence adders are seen as the most integral part of the arithmetic unit. This necessitates the focus on improving the power delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. However the carry propagation in adders needs to be addressed for enhancing the adder performance.

Adders play a key role in all three phases of digital multiplication namely: partial product generation, partial product accumulation and final addition. Modified Booth Algorithm is preferred to reduce the number of partial product rows as it involves a different encoding scheme (MBE-Modified Booth encoding) [1]. Although several different MBE schemes have been proposed, most of them do not take the hardware cost, timing performance, and power consumption into consideration at the same time. The complexity of modified booth encoding lies in the generation of odd multiples ($\pm$3X,$\pm$5X,$\pm$7X,….) (X-multiplicand), due to the higher radix encoding [2]. Hence Radix-8 booth multiplier architecture is selected over the higher radix architecture, since it needs to generate the odd multiple 3X only and reduces the partial products to (n+1)/3 (n- input bit width). However $\pm$3X multiple requires an addition operation together with shifting, it is referred as hard multiple. This drawback is overcome by special kind of carry propagate adders.

This proposed work examines various adder architectures for 3X multiple generation and compares their performance based on energy dissipation. The structure of the paper is organized as follows .Section 1 presents the introduction to the need of high speed binary addition. Section 2 describes the basics of 3X multiple generation. Ling's adder and Parallel prefix adders for 3X implementation are explained in section 3 & 4 respectively. Finally the results are discussed in section 5.

2. Preliminaries

Radix-8 Booth encoding involves the calculation of $\pm$2X, $\pm$3X and $\pm$4X multiples for the generation of partial product rows. $\pm$2X and $\pm$4X multiples are calculated by shifting the multiplicand X, on the other hand $\pm$3X requires an addition operation together with shifting, i.e 3X = 2X+X. In 2X+X addition, adjacent full adders share the input (multiplicand X). Therefore the sum and carry equations are defined as,

$$S_i = x_i \oplus x_{i-1} \oplus C_{i-1} \quad (1)$$

$$C_i = x_i . x_{i-1} + (x_i + x_{i-1}) C_{i-1} \quad (2)$$

In the conventional Ripple carry adder(RCA), n bit input produces n+2 bit output for 3X (= 2X+X) multiple as shown in Fig.1

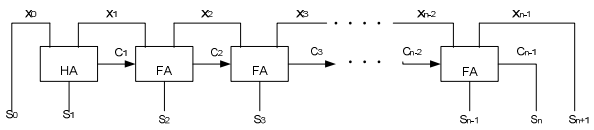

Fig 1. Conventional n bit RCA architecture for 3X generation

RCA generates the sum signal after the carry signal has rippled through the adder from least significant bit (LSB) to most significant bit (MSB). As a result, the final sum is available only after the carry propagation through the entire bit width. This indicates that RCA has O(n) delay, which will increase the delay of the booth encoder and hence the partial product generation in modified booth algorithm.

The delay of conventional RCA is reduced by Carry look ahead adder (CLA), as the carry bits are computed in advance to speed up the addition process. In this structure, two internal signals propagate (pi) and generate (g_i) are defined as,

$$\left.\begin{aligned} p_i &= x_i \oplus x_{i-1} \\ g_i &= x_i . x_{i-1} \end{aligned}\right\} \quad (3)$$

The output carry and sum signals for 3X term are given by,

$$\left.\begin{aligned} C_i &= g_i + p_i . C_{i-1} \\ S_i &= p_i \oplus C_i \end{aligned}\right\} \quad (4)$$

In CLA, all the carry signals are generated simultaneously, hence the delay is reduced by $O(\log_2 n)$. However the carry equation become quite complex when the input bit width is increased. This leads to large fan-out problem, which can be minimized by 4 bit modules [3].

3. Ling's adder for 3x multiple generation

Ling [4] proposed a new carry recurrence to reduce the logical depth for carry computation in carry look ahead structures. This new carry is referred as pseudo or Ling carry signals. As suggested by Doran [5], the recurrence involved in carry equation can be exploited to form a number of combinations for carry generation in binary addition. He also listed the properties and advantages of Ling carry signals. Ling's algorithm provides a fast alternative to CLA.

Ling's equations for 3X term are as follows:

$$\left.\begin{aligned} t_i &= x_i + x_{i+1} \\ g_i &= x_i . x_{i+1} \\ p_i &= x_i \oplus x_{i+1} \end{aligned}\right\} \quad (5)$$

Here p_i represents half sum signal and the propagate signal (t_i) is OR of consecutive bits whereas in CLA it is EXOR of consecutive bits. From the conventional CLA, $C_{i+1} = g_i + p_i C_i$, where $p_i = x_i \oplus x_{i+1}$. Ling proposed a new term pseudo carry H_i [6],[7] instead of original carry C_i, which is propagated to the sum, defined as

$$H_i = C_i + C_{i+1} \quad (6)$$

Similar to (4), the term p_i is replaced by t_i in pseudo carry as , $H_{i+1} = g_i + t_{i-1} . H_i$ (7)

The relationship between the two carry signals is derived from the identity property as given in (8), which simplifies the logic in the carry generation stage. As a result, one AND gate is removed from the critical path.

$$g_i . t_i = (x_i . x_{i+1}) . (x_i + x_{i+1}) = x_i . x_{i+1} = g_i \quad (8)$$

Hence, $g_i = g_i . H_{i+1}$ and $C_{i+1} = t_i . H_{i+1}$ (9)

From (7), H_i can be expanded as,

$$H_{i+1} = g_i + g_{i-1} + t_{i-1} . g_{i-2} + t_{i-1} . t_{i-2} . g_{i-3} + \cdots + t_{i-1} . t_{i-2} \ldots t_0 . C_{in} \quad (10)$$

The output sum is defined as (11), which is complex compared to the conventional CLA.

$$S_i = \begin{cases} x_0 & i = 0 \\ \overline{H}_{i-1} . p_i + H_{i-1} . (p_i \oplus t_{i-1}) & i = 1 \; to \; n \\ H_{n-1} & i = n+1 \end{cases} \quad (11)$$

From (9 and 10), the new carry H_i is faster than the original carry C_i, for example,

$$H_4 = g_3 + g_2 + t_2 g_1 + t_2 t_1 g_0 + t_2 t_1 t_0 C_{in} \text{ and} (12)$$

$$C_4 = g_3 + t_3 g_2 + t_3 t_2 g_1 + t_3 t_2 t_1 g_0 + t_3 t_2 t_1 t_0 C_{in} \quad (13)$$

An examination of (12 and 13), reveals that the product term in H signal is computationally simpler than C in conventional CLA. Therefore H propagates faster than C signal. This logical reorganisation of the the carry signal reduces the critical path delay in Ling adder, but this makes the computation of output sum complex as shown in Fig.2. As the output sum are off the critical path, the complexity will not affect much of the adder speed.

Fig. 2 represents the prefix computation graph of Ling's adder for 3X multiple generation for 8 bit multiplicand. Preprocessing stage computes propagate(t_i), generate(g_i) and half sum (p_i)signals, whereas level 1,2 and 3 are prefix stages for computing Ling (pseudo) carry signals(H_i). The final stage computes the sum, (i.e., 3X for the given input) with the help of Ling carry signals. The

higher level of propagate and generate signals are represented by the capital letters with the subscript representing the level and the bit position.

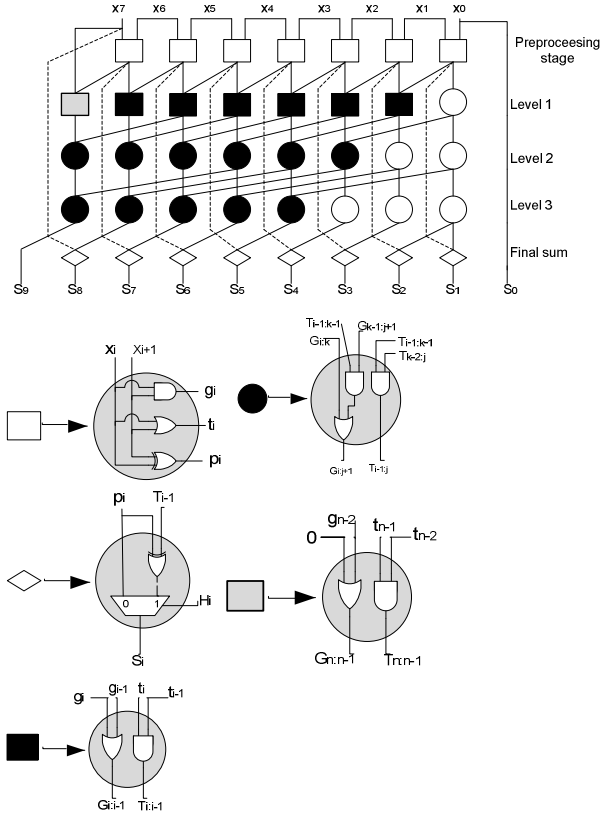

Fig. 2 Ling's Adder for 3X generation (8 bit)

A. Parallel prefix ling's adder for 3x multiple generation

An improvement of Ling's adder to achieve reduced delay as well as logic level is presented as Area efficient parallel prefix Ling adder [8]. Here the authors, compute the final carry signals (C_i), based on the Ling carry signals (H_i) produced by the lower bit positions. Since the final carry signals are computed early by one logic level compared to Ling's adder, the final sum is computed with the EXOR gate unlike Ling's adder[4], which leads to significant reduction in logic gates and lower delay. The equations defined in [8] are modified to generate 3X multiple are explained below,

$$\left.\begin{aligned} t_i &= x_i + x_{i-1} \\ g_i &= x_i . x_{i-1} \\ p_i &= x_i \oplus x_{i-1} \end{aligned}\right\} \tag{14}$$

The ling carry signals are obtained in two groups, i.e lower group for bit positions 0 to 3 (H_0 to H_3) and higher group for 4 to 7 (H_4 to H_7) in case of 8bit multiplicand. The computational nodes in Fig. 2 are modified to compute the final carry signals in parallel prefix ling adder which is depicted in Fig. 3

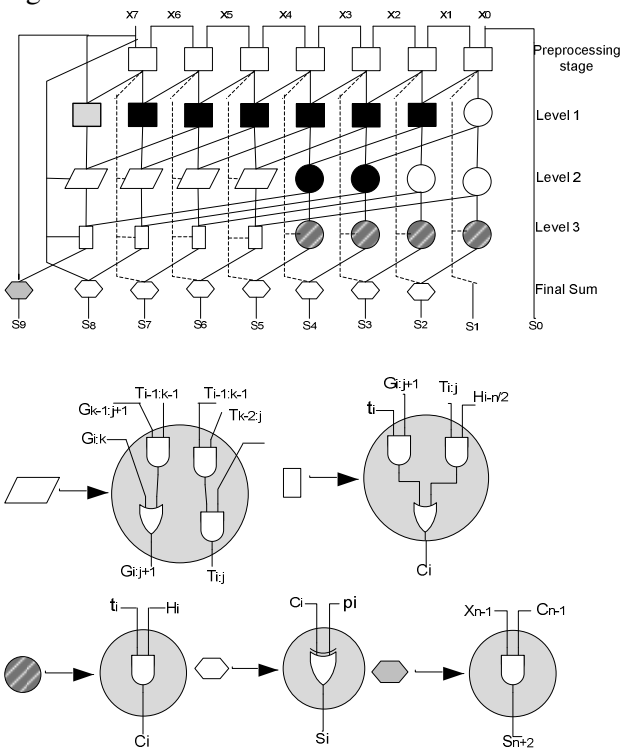

Fig.. 3 Parallel prefix Ling Adder [8] for 3X multiple generation (8 bits)

From (15), the original carry is computed by

$$C_i = t_i . H_i \tag{16}$$

With the use of this original carry, the final sum is expressed as,

$$S_i = \begin{cases} x_0 & i = 0 \\ p_i \oplus C_{i-1} & i = 1\ to\ n-1 \\ x_{n-1} \oplus C_{n-1} & i = n \\ x_{n-1} . C_{n-1} & i = n+1 \end{cases} \tag{17}$$

From (17), it is seen that the sum calculation is EXOR of Ci and pi signals which is easy compared to the Ling adder without prefix computation. From the prefix computation graph of parallel prefix Ling adder in Fig. 3, the final carry is computed early

$$H_i = \begin{cases} g_i + g_{i-1} & i = 0,1 \\ (g_i + g_{i-1}) + ((t_{i-1} . t_{i-2}) . (g_{i-2} . g_{i-3}) & i = 2,3 \\ (g_i + g_{i-1}) + ((t_{i-1} . t_{i-2}) . (g_{i-2} . g_{i-3}) + \left(\prod_{j=i-\frac{n}{2}}^{i} t_i\right) . H_{(i-\frac{n}{2})} & i = 4\ to\ n-1 \end{cases} \tag{15}$$

by one logic level compared to Fig. 2. Hence the final sum is calculated simply by EXOR of half sum (pi) and final carry (ci) which simplifies the post computation stage unlike Ling adder.

B. Conditional carry adder

Propagation of carry signals decides the operating speed of the adder, hence various adder designs are proposed for fast carry generation. Carry select adder computes two versions of the addition with carry as 0 and 1 using multiplexers. Thus the speed can be increased by increasing the number of multiplexers, which results more area and irregular structures [9]. Similarly the conditional sum adder [10] also shows the peak performance for high speed applications. Modifications to the conditional sum addition rule result in a new architecture, which reduces the number of internal nodes and multiplexers.

This type adder is referred as Conditional carry adder (CCA). CCA has a smaller capacitive load due to reduced area and less power dissipation [11].

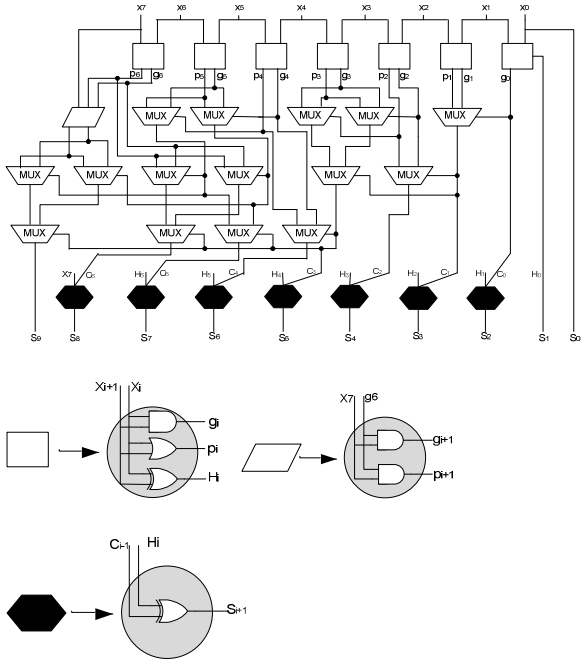

Fig.4 Conditional carry adder for 3X multiple generation (8 bits)

The structure of CCA (Fig. 4) includes three parts: a carry generation module, a conditional carry reduction unit and EXOR gates for the final sum outputs. The carry generation module generates all possible carry output signals. The conditional carry unit is used for the carry output selections of every bit using multiplexers.

The carry equation $C_i = g_i + p_i.C_{i-1}$ is also expressed as $C_i = x_i.x_{i-1} + (x_i + x_{i-1}).C_{i-1}$ for 3X multiple term. Here pi signal is defined as $p_i = x_i + x_{i-1}$. When Ci-1=0, then $C_i = x_i.x_{i-1}$ and when Ci-1=1, then $C_i = x_i + x_{i-1}$. Thus the carry generation module of every bit only contains a 2-input AND gate and a 2-input OR gate. No sum signals are generated in the unit. The 2-input EXOR gates are used to execute the final sum outputs as defined in (4).

4. Parallel prefix adders for 3x multiple generation

The parallel prefix adders are the most general form of network which is used to pre-calculate the carry signals. The maximum levels required to calculate the final carry signal is referred as the depth of the prefix graph and it is proportional to the input bit width. Many authors proposed different parallel prefix adder structures [12-15] based on the carry equation to speed up the carry propagation.

Kogge and Stone [16] developed a parallel algorithm in which the recurrence equations are split into functionally equivalent sub equations each can be executed in parallel to improve the speed of the process. Successive splitting of this sub functions further reduces the delay. This property is used in parallel prefix adder for carry generation in binary addition. Prefix adder generates the carry signals simultaneously and has O(2 log2n) delay [17]. PPA also uses the propagate (pi), generate (gi) signals defined in (3) and the prefix operator is defined as,

$$(g_i, p_i) \circ (g_{i+1}, p_{i+1}) = (g_i + p_i.g_{i+1}, p_i.p_{i+1}) \quad (18)$$

The basic difference between PPA and CLA lies in the organization of the carry equations. Since the carry path of the adder constitutes the critical path, PPA improves the speed of the addition over the other architectures.

Kogge-Stone adder employs the idempotency property to limit the number of lateral logical fan-out at each node to unity. It has the advantage of log2n logic levels at the cost of increased the number of lateral wiring at each level. In this design, the number of logic gates is increased and consumes a large amount of power because of the massive overlap between the prefix sub-terms being pre-computed (Fig. 5).

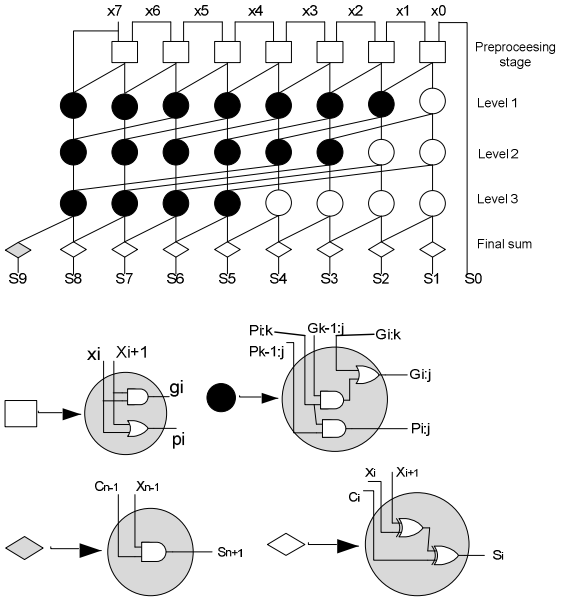

Fig. 5 Kogge-Stone (KS) adder for 3X multiple generation (8 bits)

Sklansky proposed divide-and-conquer approach to reduce the depth of the prefix tree by computing the intermediate prefixes along with the large group prefixes. This recursive doubling idea is derived from Conditional sum adder [10]. This kind of logical reorganization reduces the delay by log2 and increases the fan-outs double at each level that degrades the performance at higher logic levels. With appropriate buffering and transistor sizing, this fan-out problem could be carefully addressed to improve the performance of the adder (Fig. 6).

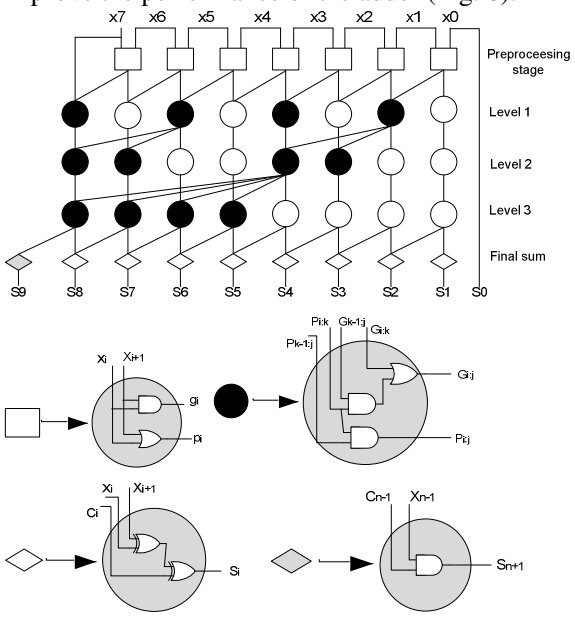

Fig. 6 Sklansky (SK) adder for 3X multiple generation (8 bits)

5. Experimental results and discussions

This section presents the experimental results of the adders taken for comparison in the context of relating the energy efficiency to the adder architecture. These adders are simulated using ISE simulator with the VHDL structural coding. Cadence RTL complier with TSMC library 180nm is used to synthesize and analyze the cell area, power and delay. All the evaluations are taken after post place and route synthesis of cadence digital flow.

Table. 1 compares the performance of various adder architectures that have been taken for the reference. Here the cell area and power refer to the amount of space occupied and the power dissipated by the logic, and the delay refers to the maximum path delay involved in producing the sum output signals. On the examination of the simulated results, RCA achieves the least power and area, however with the penalty of maximum delay. CLA, Ling adders are using the modified recurrence equations to minimize the path delay which is proved in the result table. Parallel prefix ling adder is showing reduced area compared to Ling adder.

The conditional carry adder mainly concentrates on the fast carry generation by using multiplexers in the critical path. As multiplexers are occupying more space, the area of CCA is increased in comparison to other adder architectures.

Parallel prefix architectures are characterized by three main design parameters: logic level, number of fan-outs and wire tracks. Here we consider the Kogge-Stone and Sklansky architectures for the reason that both the architectures are having the minimum logic levels (log2n) when compared to other prefix architectures. Though the KS adder has minimum logic level, it has larger number of computational nodes and hence the wire tracks. This increases the area as well as more power dissipation in wire tracks which results in maximum energy dissipation. In contrast to the KS adder, SK adder has reduced number of computational nodes and wire tracks. This design minimizes the energy dissipation ~50% in comparison to KS adder for 64 bit input (Fig. 7)

As the Sklansky network has increased fan-out problems at each level, which cause poor performance on large input bit width. Hence the fan-out problems have to be addressed carefully. The domino logic style of implementation mainly concern about the processing speed of the architecture due to the reduced fan-in capacitance

Table 1 Comparison of Adders for cell area, Power and Delay

Adder Type	8 bits			16 bits			32 bits			64 bits		
	Cell area (μm^2)	Power (nw)	Delay (ns)	Cell area (μm^2)	Power (nw)	Delay (ns)	Cell area (μm^2)	Power (nw)	Delay (ns)	Cell area (μm^2)	Power (nw)	Delay (ns)
RCA	127	3759	1.24	268	8637	2.66	538	18702	4.83	1024	35534	8.45
CLA	217	5014	0.97	780	14819	1.79	1826	32647	2.33	4764	63143	3.94
Ling	213	4943	0.93	775	13463	1.61	1724	28598	2.27	4176	60854	3.82
PP-Ling[8]	236	4528	0.87	697	12513	1.51	1484	26393	2.12	3854	59365	3.54
conditional carry adder[10]	237	4710	0.77	573	11578	1.13	1825	25143	2.02	4002	59043	3.45
PPA-Kogge-Stone	216	4275	0.84	622	11492	1.21	1723	30610	1.85	4530	64403	2.82
PPA_Sklansky	94	3527	0.86	465	8442	1.12	1135	20590	1.58	2364	44760	1.98

and faster switching thresholds, this style of implementation could be adopted for addressing this fan-out problem [18]. The transistor which drives the next stage has to be sized large hence this fan-out problem can be solved. However this variable sizing of transistors results in irregular layout [19].

In ASIC design, each module is customized for enhancing performance of the overall design. However, a close observation of the gate sizes in the fully custom designed, Sklansky adder shows that, of all the fan-out gates at each stage, only one gate needs to be large and the rest can be sized uniformly smaller. This kind of design suits for high speed applications with reduced energy dissipation.

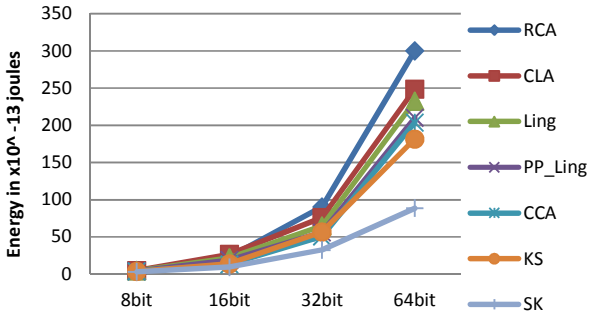

Fig. 7 Comparison of Energy dissipation in 10^{-13}joules

6. Conclusion

This paper has presented a comparative study of adder performances for Energy efficient design. Radix-8 booth encoding necessitates the generation of hard multiple (3X) to enhance the multiplication speed. Hence the selection of adder architecture is the key factor for any multiplication. Parallel prefix structures exhibits flexibility in the design constraints which is more suited for the efficient implementation of custom binary adders. In this paper, we analyze the areas and performances of various families of adder architectures and we found that the Slansky adder has the minimum area and energy dissipation suited for the custom design of 3X adder .Methodologies for minimizing the fan-out problem in Sklansky adder also discussed in this paper.

References

1 O.L. MacSorley, " *High Speed Arithmetic in binary computers*", Proc. IRE, vol49, pp.67-91, 1961..

2 P E Madrid , B Millar , and E Swartzlander , Jr, "*Modified Booth Algorithm for High radix Multiplication*", IEEE Computer Design Conf., pp.118-121, 1992.

3 B. Parhami," *Computer Arithmetic Algorithms and Hardware Designs*", Oxford Univ. Press, New York, 2000.

4 H.Ling,"*High speed Binary Adder*," IBM J. R& D, vol.25, pp.156-166,1981.

5 R.W.Doran, "*Variants of an Improved carry Look Ahead Adder,*" IEEE Trans. Computers, vol.37.pp-1110-1113,1988.

6 N. Burgess, "*Implementation of Recursive Ling Adders in CMOS VLSI*," Proc. 43rd Asilomar Conf. Signals,Systems, and Computers, pp. 1777-1781, 2009.

7 Lakshmanan,Ali Meaamar, Masuri Othman ,"*High-Speed Hybrid Parallel-Prefix Carry-Select Adder using Ling's Algorithm,*" ICSE,Malaysia, 2006,

8 Tso-Bing Juang, Pramod Kumar Meher, Chung-Chun Kuan,"*Area-efficient parallel-prefix Ling adders,*" APCCAS ,pp. 736-739,2010

9 Y. Wang, C. Pai, and X. Song, "*The design of hybrid carry-lookahead/carry-select adders*," IEEE Transactions on Circuit and Systems II: Analog and Digital Signal Processing, Vol. 49, 2002, pp. 16-24

10 J.Sklansky,"*Conditional-sum addition logic*",IRE Transactions on Electronic Computers, Vol. EC-9,pp.226-231,1960

11 Kuo-Hsing Cheng, Shun-wen Cheng, "*Improved 32 bit Conditional Sum Adder for Low-Power High Speed Applications*, "Journal of Information Science and Engineering, vol.22,pp. 975-989,2006

12 R.E. Ladner and M.J. Fisher, "*Parallel Prefix Computation*,"J. ACM,vol. 27,no. 4, pp. 831-838, Oct. 1980.

13 R.P. Brent and H.T. Kung, "*A Regular Layout for Parallel Adders*,"IEEE Trans. Computers,vol. 31, no. 3, pp. 260-264, Mar. 1982.

14 T. Han and D. Carlson, "*Fast Area-Efficient VLSI Adders,*"Proc. Symp.Computer Arithmetic,pp. 49-56, May 1987.

15 S. Knowles, "*A Family of Adders*,"Proc. 14th Symp. Computer Arithmetic, pp. 30-34, Reprinted in ARITH-15, pp. 277-281, Apr. 1999

16 P. M. Kogge and H. S. Stone, "*A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations*", IEEE Transactions on Computers, Vol. 22 (a), pp. 783-791, Aug 1973

17 G. A. Ruiz and M. Granda, "*An area-efficient static CMOS carry-select adder based on a compact carry look-ahead unit,*" Microelectronics. J.,vol. 35, no. 12, pp. 939–944, Dec. 2004

18 B.R.Zeydel, D.Baran, V.G.Oklobdzija, "*Energy-Efficient Design Methodologies: HighPerformance VLSI Adders*", IEEE Journal of Solid-State Circuits, vol.45, no.6, pp.1220-1233, 2010

19 Dinesh Patil, Omid Azizi , Mark Horowitz, Ron Ho and Rajesh Ananthraman,"*Robust EnergyEfficient Adder Topologies*" Proc. of 18th IEEE Symposium on Computer Arithmetic(ARITH'07), pp.16–28. IEEE Computer Society Press, 2007