October 2014

IMPLEMENTATION OF 32-BIT HIGH-VALENCY LING ADDER IN MODIFIED FIR FILTER USING APC-OMS APPROACH

K. V. SURESH KUMAR

VLSI Design, Department of ECE, SRM University, Chennai., connect2suri@gmail.com

S. SHABBIR ALI

VLSI Design, Department of ECE, SRM University, Chennai., shabbir.ali46@gmail.com

E. CHITRA

Department of ECE, SRM University, Chennai., chitranithil@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijeee>

Part of the [Power and Energy Commons](#)

Recommended Citation

KUMAR, K. V. SURESH; ALI, S. SHABBIR; and CHITRA, E. (2014) "IMPLEMENTATION OF 32-BIT HIGH-VALENCY LING ADDER IN MODIFIED FIR FILTER USING APC-OMS APPROACH," *International Journal of Electronics and Electrical Engineering*: Vol. 3 : Iss. 2 , Article 10.

DOI: 10.47893/IJEEE.2014.1140

Available at: <https://www.interscience.in/ijeee/vol3/iss2/10>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Electronics and Electrical Engineering by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

IMPLEMENTATION OF 32-BIT HIGH-VALENCY LING ADDER IN MODIFIED FIR FILTER USING APC-OMS APPROACH

K. V. SURESH KUMAR¹, S. SHABBIR ALI,² E. CHITRA³

^{1,2}M.Tech, VLSI Design, Department of ECE, SRM University, Chennai.

³Associate Professor(Sr.G), Department of ECE, SRM University, Chennai.

E-Mail: connect2suri@gmail.com, shabbir.ali46@gmail.com, chitranihil@gmail.com

Abstract - Ling Adder is an advanced architecture of Parallel prefix adders. Parallel Prefix adders are used for efficient VLSI implementation of binary number additions. Ling architecture offers a faster carry computation stage compared to the conventional parallel prefix adders. Ling adders help to reduce the complexity as well as the delay of the adder further. In many computers and other kinds of processors, adders are used not only in the Arithmetic Logic Unit (ALUs), but also in other parts of the processor. Thus the delay in adders has to be decreased as maximum as possible to make the system faster. In particular, valency or the number of inputs to a single node is explored as a design parameter. High-valency Ling adders have superior area x delay characteristics over previously reported Ling-based or non-Ling based adders for the same input size. In DSP, even the multipliers require a large number of logic gates that consumes more area, power and delay. Hence, the lookup table can be used for performing computation which requires less area. Therefore, APC and OMS are the two techniques implemented in Lookup table so as to reduce the size to one-fourth of its conventional multiplier. This proposed combination of both Lookup table Multiplier and Ling adder has a better area and delay measurement.

Keywords: Parallel prefix, Valency, ALU, factorization APC-OMS, Lookup table.

1. INTRODUCTION

In electronics, an adder is a combinational or sequential logic element which computes the n-bit sum of two numbers. The family of **Ling adders** is particularly a fast adder and is designed using H. Ling's equations and generally implemented in Bi CMOS.

Binary addition is one of the fundamental operations in electronic circuits. Many modern circuits contain several adder units for applications such as arithmetic logic unit, memory addressing and program counter update. Thus, there is a considerable interest to design higher speed and less complex adder architectures. For the last few decades, several adder architectures have been proposed to optimize the adder delay, examples include ripple carry adder, carry-look ahead adder, and parallel prefix adder.

The parallel prefix adder is one of the most popular architectures and offers good compromise among area, speed and power. This type of adder implements a logic function to determine whether each bit position generates the carry, propagates it or kills it. Then these generate and propagate/not kill functions are hierarchically combined to compute the carry into each bit position forming a carry tree. The final stage computes sum at every bit position using exclusive or (XOR) gates.

2. HIGH PERFORMANCE ADDERS

1.1 Kogge Stone Adder

The Kogge-Stone adder is classified as a parallel prefix adder since the generate and the

propagate signals are pre computed. In a tree-based adder, carries are generated in tree and fast computation is obtained at the expense of increased area and power. The main advantage of this design is that the carry tree reduces the logic depth of the adder by essentially generating the carries in parallel. The parallel-prefix adder becomes more favorable in terms of speed due to the $O(\log 2n)$ delay through the carry path compared to $O(n)$ for the RCA. The Kogge-Stone adder is widely used in high-performance 32-bit, 64-bit, and 128-bit adders as it reduces the critical path to a great extent compared to the ripple carry adder.

To find the carry out, the carry has to propagate in the critical path to get minimum delay. As far as carry is in the critical path, the adder works faster. But as the number of bits increases, the number of stages also increases which becomes complicated for a designer to design. Hence we go for some other advanced structures.

The logic used in the Kogge stone Adder is Carry Look Ahead Adder which is also known in brief in the next section.

Equations used in the structure

- Black cell
 - $g = g_i + p_i \cdot g_j$
 - $p = p_i \cdot P_j$
- Gray cell
 - $g = g_i + p_i \cdot g_j$
- Buffer
 - $g = g_i \ \& \ p = p_i$

Fig 2.1: 16-bit Kogge Stone Adder

1.2 Carry Look Ahead Adder

To reduce the computation time, engineers devised faster ways to add two binary numbers by using carry-look ahead adders. They work by creating two signals (P and G) for each bit position, based on if a carry is propagated through from a less significant bit position (at least one input is a '1'), a carry is generated in that bit position (both inputs are '1'), or if a carry is killed in that bit position (both inputs are '0'). In most cases, P is simply the sum output of a half-adder and G is the carry output of the same adder. After P and G are generated the carries for every bit position are created. Some advanced carry-look ahead architectures are the Manchester carry chain, Brent–Kung adder, and the Kogge–Stone adder. The CLA Adder is able to generate carries before the sum is produced using the propagate and generate logic to make addition much faster.

Fig 2.2: 16-bit Carry LookAhead Adder

- The equations used in Carry look Ahead adder to decrease the complexity

$$\begin{aligned} p_i &= a_i \oplus b_i & , \quad g_i &= a_i \cdot b_i \\ s_i &= a_i \oplus b_i \oplus c_i & \Rightarrow s_i &= p_i \oplus c_i \\ c_0 &= a_i \cdot b_i + p_i \cdot c_i & \Rightarrow c_0 &= g_i + p_i \cdot c_i \end{aligned}$$

Where 'p_i' is the carry propagation and

'g_i' is the carry generation

'a_i and b_i' are the input bits

's_i and c_o' are sum and carry of the last (output) stage

- Carry propagation in CLA

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$

$$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

$$C_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

Where, C₁ is the carry out of 1st stage, and propagated to the next stage.

G and P are group generation and group propagation respectively.

The disadvantage of CLA is that the carry logic block gets very complicated for more than 4-bits. For that reason, CLAs are usually implemented as 4-bit modules and are used in a hierarchical structure to realize adders that have multiples of 4-bits.

3. LING ADDER USING LADNER FISCHER STRUCTURE

3.1 Parallel Prefix Adder

Prefix adders compute the carry from a group of bits by hierarchically combining the carries from subgroups forming a tree structure called carry tree. The equation for computing the carry out from one group of bits (0 to i-1), by combining the carry out (group generate and not kill) from two subgroups formed over bits i-1 to n and n-1 to 0 is given as follows:

$$C_i = G_{i-1}^0 + K_{i-1}^n \cdot G_{n-1}^0 \quad (1)$$

where $1 \leq n \leq i-1$, and G and K are group generate and group kill signals, respectively.

Fig 3.1: Prefix operator with valency 2, 3 and 4

The number of stages to produce the carries is equal to $\log_2 n$ (n is the width of the adder). Base is 2 as this is for valency-2 adder. This stage is formed by implementing prefix cells which are called grey or black prefix cells where the prefix operator is implemented. The number of lines going into the dot in the prefix cell determines the valency of the adder as illustrated in Fig 3.1. Note that the prefix cell has actually four inputs (two G's and two K's).

The sum, $S = s_{n-1} s_{n-2} \dots s_0$, is calculated by

$$s_i = p_i \oplus C_i = p_i \oplus G_{i-1}^0 \quad (2)$$

where 'C_i' is the carry in the bit position i, which is the carry out of (i-1)th stage. Thus, in the sum logic stage, the sum at each bit position can be computed by implementing an XOR gate with one input as bit level propagate at that bit position and the second input as carry into that bit position.

The parallel prefix adder employs the 3-stage structure of the CLA adder. The improvement is in the carry generation stage which is the most intensive one.

Fig 3.2: Prefix Computation in 3 stages

Prefix: The outcome of the operation depends on the initial inputs.

Parallel: Involves the execution of an operation in parallel.

This is done by segmentation into smaller pieces that are computed in parallel.

Ling Adder is a special kind of Carry Look-Ahead Adder. An Ex-OR gate is Replaced with an OR gate which results in a much Cheaper operation. Propagates h_i instead of c_i Where $h_i = c_i + c_{i-1}$ This approach is faster and less expensive.

Design Principle

The term p_i is replaced with t_i as follows:

$$p_i = a_i \oplus b_i \text{ and } t_i = a_i + b_i$$

$$\text{where } t_i = p_i + g_i$$

$$c_i = g_i + t_i \cdot c_{i-1}$$

$$= t_i \cdot (g_i + c_{i-1}) \quad \text{where } g_i = t_i \cdot g_i$$

$$= t_i \cdot (g_{i-1} + t_{i-1} \cdot c_{i-2} + c_{i-2})$$

$$= t_i \cdot (c_i + c_{i-1})$$

$$= t_i \cdot h_i$$

$$s_i = p_i \oplus c_{i-1} = p_i \oplus (t_{i-1} \cdot h_{i-1})$$

Compared with normal CLA adder:

$$c_i = g_i + c_{i-1} t_i$$

$$= g_i + g_{i-1} \cdot t_i + g_{i-2} \cdot t_{i-1} \cdot t_i + \dots + t_i \cdot t_{i-1} \cdot \dots \cdot t_0 \cdot C_{-1}$$

$$h_i = g_i + g_{i-1} + g_{i-2} \cdot t_{i-1} + \dots + t_{i-1} \cdot t_{i-2} \cdot \dots \cdot t_0 \cdot h_{-1}$$

Each product term in h_i has one less "AND" gate correspondingly.

3.2 Ladner Fischer

The 32-bit Ling Adder using Ladner Fischer structure is depicted in the figure 3.3 by dividing it into two blocks consisting 12-bit with the valency 2X3X2 and 20-bit with the valency 2X5X2.

Fig 3.3: 32-bit Ling Adder using Ladner Fischer structure

The structure is divided into two parts 12-bit and 20-bit to compute the carry faster and delay is decreased further when compared to CLA, Kogge Stone Adder. The power consumption is also decreased when compared to previous adders as shown in table 3.1.

Table 3.1 Delay and Power comparison among different 32-bit Adders.

Adder(32-bit)	Delay(ns)	Power(W)
RCA	3.80	1.55
CLA	3.62	0.96
Ling Adder(with H.Ling Eq's)	3.272	0.625
Kogge Stone	3.192	0.444
Ling Adder with Valency	3.132	0.349

4. LOOKUP TABLE MULTIPLIER

Digital signal processing can be defined as the processing of digital information with minimum noise. The computation in digital systems increases with decreasing area size. Therefore, new approaches are to be considered to optimize the size of memory along with power consumption. Multiplication, nothing but the repeated addition plays a vital role in signal processing. Memory based computations are more regular than the multiply and accumulate structures and offer many advantages.

For optimizing lookup table in order to obtain memory utilization different schemes are proposed. They are Anti-Symmetric product coding scheme (APC), Odd-Multiple Storage scheme (OMS).

The proposed LUT design involves the combination of both the anti-symmetric product coding and the odd multiple storage schemes.

4.1 Anti-Symmetric Product Coding Scheme

For any of the circuit provided we are storing those values in a Memory array and reducing the storage values and, retrieving the left over values. Therefore, the total size reduces to half. Finally, instead of storing all the input values only the first half are stored and vice versa.

Fig4.1: Anti-symmetric product coding

4.2 Odd Multiple Storage scheme

In an OMS scheme, only the odd multiple values are stored. The addressed Apc values are re-addressed in OMS using 4to3 Address Encoder. A memory element (or) Memory array can be designed using a 3to8 decoder. Barrel Shifter is designed using 4to1 Multiplexers.

BLOCK DIAGRAM:

Fig 4.2: Odd multiple storage block diagram

All the product values can be reset using NOR Cell. Switching of the selected lines is controlled by the control circuit.

Table 4.1: Storage of values in OMS approach

Address word	Product word	Address word	Product word	Address word	Product word
0000	0	1000	8A	0001	A
0001	A	1001	9A	0011	3A
0010	2A	1010	10A	0101	5A
0011	3A	1011	11A	0111	7A
0100	4A	1100	12A	1001	9A
0101	5A	1101	13A	1011	11A
0110	6A	1110	14A	1101	13A
0111	7A	1111	15A	1111	15A

Table 4.2: Storage of values in APC approach

Input	Product value 'X'	Input	Product value 'X'	Address	Apc Word
00001	A	11111	31A	1111	15A
00010	2A	11110	30A	1110	14A
00011	3A	11101	29A	1101	13A
00100	4A	11100	28A	1100	12A
00101	5A	11011	27A	1011	11A
00110	6A	11010	26A	1010	10A
00111	7A	11001	25A	1001	9A
01000	8A	11000	24A	1000	8A
01001	9A	10111	23A	0111	7A
01010	10A	10110	22A	0110	6A
01011	11A	10101	21A	0101	5A
01100	12A	10100	20A	0100	4A
01101	13A	10011	19A	0011	3A
01110	14A	10010	18A	0010	2A
01111	15A	10001	17A	0001	A
10000	16A	10000	16A	0000	0

Finally, the proposed logic is implemented in Digital signal Processing FIR filter. The multiplier section of the filter circuits are implemented with the proposed LUT based multiplier and the adder section is implemented using Ling Adder.

Fig 4.3: Filter design using Lut Multiplier and Ling Adder

5. SIMULATION RESULTS

Implementation of an LUT Multiplier in a 4tap FIR filter

Fig 5.1: Simulation results of an APC-OMS Lookup table based Multiplier

Simulation results for Ling Adder **Fig 5.2: Simulation result of a 32-bit ling Adder** **Fig 5.2: Simulation result of a 32-bit Ling Adder using Ladner Fischer Structure****6. CONCLUSION**

As the no. of bits increases, the delay is increases, to decrease the delay further Ling factorization can be recursively applied to all stages in a carry computation tree of an adder. This makes some other paths more complex, but if the complexity is properly balanced the resulting adder can work faster. Using valency-5, 20-bit and 32-bit adders are better in terms of both speed and area than the conventional adders. Therefore, the lookup table based multiplier is an area efficient alternative for

other computation methods and requires comparatively less time using Ling Adder for processing, than the conventional multipliers and therefore it can offer high speed operations and is more suitable for High speed low level DSP applications.

REFERENCES

- [1] Taskin Kocak, Preeti Patil, "Design and Implementation of High-Performance High-Valency Ling Adders." 978-1-4673-1188-5/12/ ©2012 IEEE
- [2] Dayu Wang, Xiaoping Cui, Xiaojing Wang, "Optimized design of Parallel Prefix Ling Adder", International Conference on Electronics, Communications and Control (ICECC) 2011.
- [3] Dimitrakopoulos. G, Nikolos. D, "High-speed parallel-prefix VLSI Ling adders", IEEE Transactions on Computers, Feb. 2005.
- [4] Matthew Keeter, David Money Harris, Andrew Macrae, Rebecca Glick, Madeleine Ong and Justin Schauer, "Implementation of 32-bit Ling and Jackson Adders", Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2011
- [5] P.K. Meher, "LUT Optimization for Memory-Based Computation" in Circuits and SystemII, April. 2010, pp. 285–289.
- [6] H.-C. Chen, J.-I. Guo, T.-S. Chang, and C.-W. Jen, "A memory efficient realization of cyclic convolution and its application to discrete cosine transform"
- [7] P.K. Meher, "Memory-based hardware for resource constrained digital signal processing systems," in Proc. 6th Int. Conf. ICICS, Dec. 2007
- [8] Keshab K. Parhi- University of Minnesota, VLSI Digital Signal Processing Systems, Design and Implementation, copyright 1999, by John Wiley & Sons Inc., reprint 2008. ISBN: 987-81-265-1098-6.
- [9] "Adder Designs", http://www.acsel-lab.com/Projects/fast_adder/adder_designs.htm.