Performance of Constant Addition Using Enhanced Flagged Binary Adder

Sangeetha A
UG Student, Department of Electronics and Communication Engineering
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India

Vinu G C
UG Student, Department of Electronics and Communication Engineering
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India

Karthick S
Assistant Professor (Senior Grade), Department of Electronics and Communication Engineering
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India

Abstract- The main aim of this paper is to analyze the performance characteristics of various adder circuits. It includes the implementation of the adder circuit by reducing the cost to propagate the carry between successive bit positions in an efficient manner. The problem of carry propagation is eliminated by expressing addition as a prefix computation. A new technique enabling the utilization of integer arithmetic to enhance the functionality of the circuits is analyzed. The new set of intermediate outputs are generated as the part of this technique, are called as flag bits. This flag bits are used to achieve three operand additions to improve the overall performance. This design is called as Enhanced Flagged Binary Adder.

Keywords- Constant addition, flag bits, carry save adder, flagged inversion cells, computation, carry skip adder and carry select adder.

I. INTRODUCTION

In Microprocessor, Digital Signal Processor and data processing in Application Specific Integrated Circuit uses the addition of two binary numbers as the fundamental arithmetic operations. The main performance of a VLSI chip is the processing of data and controlling of internal or external system components. This is typically done by algorithms, which is based on logic and arithmetic operations. In many complex arithmetic algorithms, the multi operand addition forms the basic fundamental operation in multiplication and certain DSP algorithms. One of the popular multi-operand adder is carry save adder to compute the sum of three or more bit numbers at a time. In this paper the basic three operand additions in carry skip adder and carry select adder. It also compares the performance of these adders by this technique can be used for multi bit operands [1].

The following sections will discuss the various adder circuits, which are to be analyzed. The Section II deals with adder circuits like Carry Skip adder, Carry select adder and Carry Save adder. The section III deals with the flag bit computation and the Section IV gives the conclusion of the analysis.

II.ADDER CIRCUITS

A. Carry Skip Adder –

A Carry skip adder is an adder implementation, which uses a regular full adder for every bit position. It also generate bit propagate signals. The adder structure is divided into blocks of consecutive stages with full adder scheme modified to output, the bit propagate signals. In carry skip adder, the carry will propagate to position *k.* The advantage of carry skip adder is to improve speed-up operation. Since the propagation of carry is skipped to position *i* without waiting for rippling [2]. The carry skip logic is added to each block to detect when carry-in the block can be passed directly to the next block. The carry transfer can be given as

$$T_i=a_i + b_i \qquad (1)$$

In carry skipping both propagate and generate carries are taken into account. The worst case operation time takes place when,

- Carry is generated in the first block
- Carry skips intermediate stages
- Carry is killed in the last block

The worst case delay can be improved by using several carry skip adders. In practice, the non-uniform block size gives the best performance. In general outer blocks should be smaller than the middle blocks.

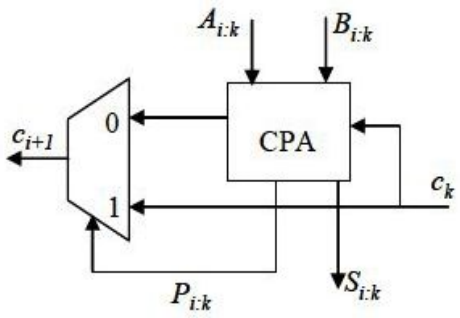

Figure.1 Carry skip block

In this figure the propagate signal is generated by the carry ripple chain, which connecting to *n* input AND gate. This propagate signal act as a select signal for 2:1 multiplexer.

B. Carry Select Adder–

A carry select adder is the composition of the concatenation and the selection scheme. Each bit position includes the generation of two sums and carry bits and also the selection multiplexers for the correct sum bit. The sum bit is selected at the end of the block [3]. The underlying strategy of the carry select adder is to generate two results in parallel. One result assumes the input carry to 0 and another assumes the input carry to be 1. The carry select adder is divided into blocks of *m* bit vectors. Once the input carry for the particular stage has been computed and assigned, the final result is selected from the two pre-computed sets [4].

The basic problem faced in speeding up carry propagation is the fast processing of the late carry input. Since this carry-in can have only two values, 0 and 1, the two possible addition results can be pre-computed and selected afterwards by the late carry-in using small and constant time. The n bit adder is broken down into groups of m bits. Each group of m bits are added utilizing what are called m bit conditional adders [5].

The carry select adder comes in the category of conditional sum adder in actual carry input which gives the actual calculated values of sum and carry are selected using a multiplexer. The carry-out calculated from the last stage that is the least significant bit stage is used to select the actual calculated values of output carry and sum. The selection is done by using the multiplexer. This technique of dividing adder into two stages in which it require more area but addition take place faster compare to normal addition [6].

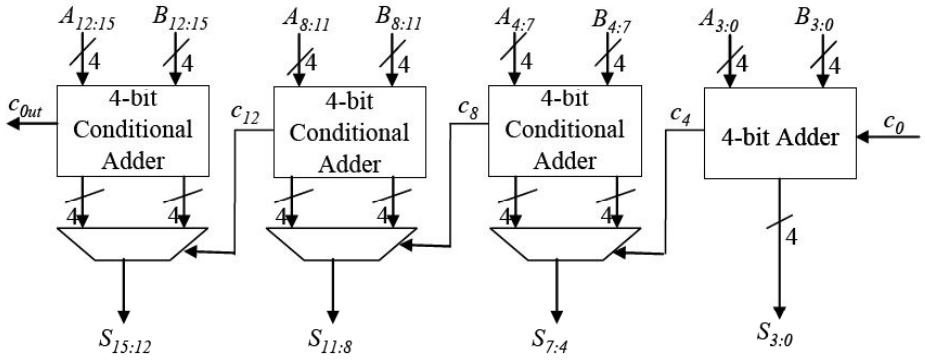

Figure.2. Carry select adder

C. Carry save adder –

The carry save adder is a type of digital adder, to compute the sum of three or more *n*-bit numbers in binary. It is clear that carry-propagation is time consuming and must be repeated several times i.e. *k* operands require k-1 propagations. There exist techniques for lowering this penalty and the one that is most commonly user is the carry save addition. Carry propagates only in the last step, while, other steps generate partial sum and sequence of carries. The basic CSA accepts three *n* bit operands and generates two *n* bit results, which are: *n* bit partial sum and *n* bit carry. While the second CSA accepts the two sequences and another input operand, that generates new partial sum and carry. Therefore, carry save adder is used to add three or more operands.

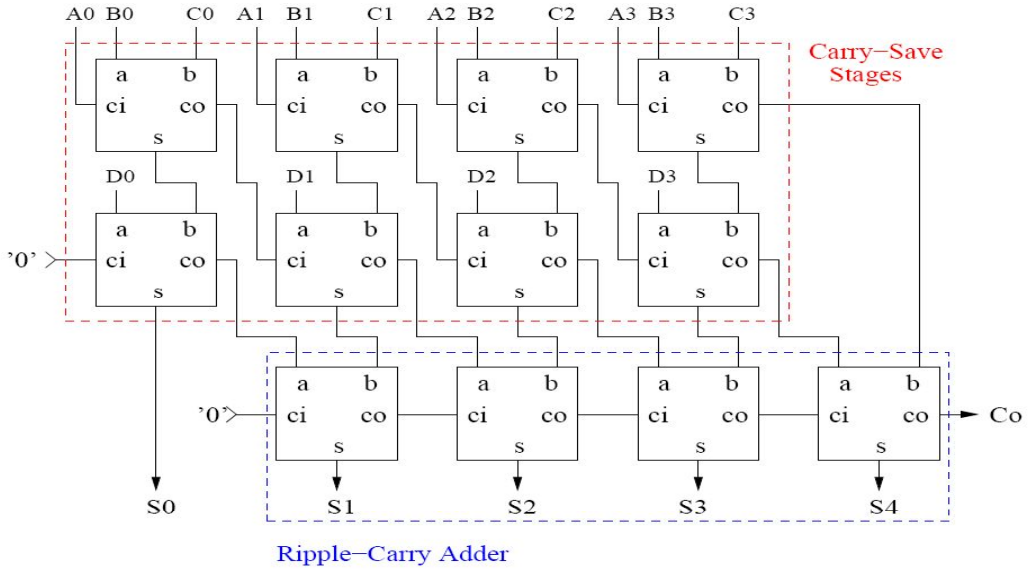

Figure.3.Carry save adder

The carry save adders-as the most commonly used redundant arithmetic adders. It plays an important role in the efficient implementation of multi-operand addition circuits. They are very fast due to the absence of any carry propagation paths, their structure is very simple, but potential for further optimization is minimal, same holds for single digital adders, which use a slightly different redundant number representation .The addition results however usually have to be converted into an irredundant integer representation in order to proceed further [3].This operation is done using a carry propagation adder.

A row of binary full adders can be viewed as a mechanism to reduce three numbers to two numbers to their sum. Figure 4 shows the relationship of the ripple carry adder for the latter reduction and a carry save adder for the former. The dotted lines represented the flow of the carry signals within a ripple carry adder.

The basic idea is to perform an addition of three binary vectors using an array of one bit adders without propagating the carries. The sum is a redundant *n* digital carry save number, consisting of two binary numbers sum bit (s) and carry bit (c). A carry save adder accepts three binary input operands or alternatively one binary and one carry save operand. It has a constant delay independent of *n* .Mathematically,

$$2c + s = a0 + a1 + a2 \qquad (2)$$

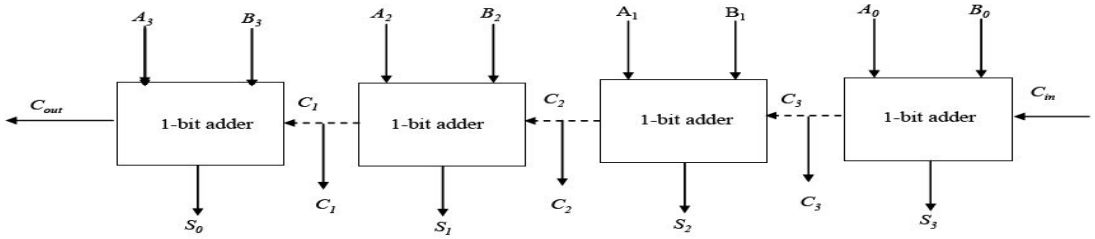

Figure.4.Carry save Block

Speeding-up the operation of the carry propagate adder (Ripple carry adder) is not efficient because each ripple carry adder has to be replaced by some faster adder structure. On the other hand, the balanced bit arrival profile of the carry save adder array allows for massive speed-up since it utilizes the carry ripple adder only in the last stage has shown in the Figure.3.

The following section derives the technique to achieve multi-operand addition utilizing regular binary adders. The first step during this process is to assume that one of the operand is a constant. This is then followed by deriving the necessary logic that allows the third binary operand to be any arbitrary number.

III. FLAG BIT COMPUTATION

Assume that there are three input operands A, B and C. The full adder, the basic building block takes three input bits, a_k, b_k, c_k and produces two outputs a sum bit (s) and carry bit (c) [7]. The equations of the full adder are,

$$S_k = a_k \text{ XOR } b_k \text{ XOR } c_k \qquad (3)$$

$$C_{k+1} = a_k b_k + a_k c_k + b_k c_k \qquad (4)$$

Assuming that the output s is to be added with the third number, the full adder equation can be rewritten as,

$$R_k = s_k \text{ XOR } d_k \text{ XOR } c_k \qquad (5)$$

$$C_{k+1} = s_k \,.\, d_k + d_k \,.\, c_k + c_k \,.\, s_k \qquad (6)$$

Utilizing these equations, then new function called as flag function, can be computed such that,

$$F_k = m_k \text{ XOR } c_{k+1} \qquad (7)$$

Where F_k is the flag bit. The flag function is utilized such that it determines whether the current is flagged to change. Consequently, this structure can be formulated by developing flag equations,

$$C_{k+1} = \begin{cases} s_k c_k & \text{if } d_k = 0 \\ s_{k+} c_k & \text{if } d_k = 1 \end{cases} \qquad F_k = \begin{cases} c_k & \text{if } d_k = 0 \\ c_k' & \text{if } d_k = 1 \end{cases} \qquad (8)$$

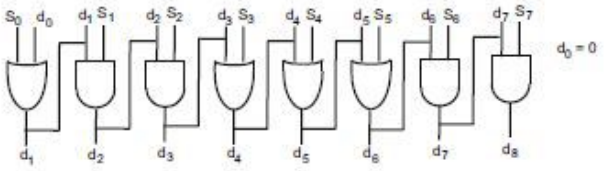

Figure.5. Carry bit computation

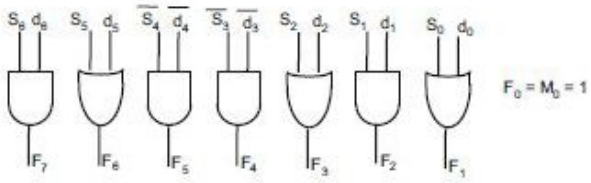

Figure.6. Flag bit computation

Similar to the conditional sum adder, the flag bit is chosen is based on either a 1 or a 0 for the present and the previous bit of the third operand. Utilizing the above equations, the new flag equations can be computed.

D_k	D_{k-1}	F_k
0	0	$S_{k-1}\, c_{k-1}$
0	1	$S_{k-1} + c_{k-1}$
1	0	$S_{k-1}' + c_{k-1}'$
1	1	$(S_{k-1}\, c_{k-1})'$

Figure.7. Flag bit logic

Based on the flag logic equations derived the hardware was designed to achieve constant addition. The resulting adder design is called as enhanced flagged binary adder. It incorporates extra hardware allowing the third operand to be any constant. The underlying technique is to generate flag bits but computation of flag bits gets more complicated due to the dependence on the bits of the third operand. Computation of flag bits for this new adder design depends on the carry outputs, readily available from the each of the selected adder designs.

Consider the following example where A=10, B=10, D=6

A		1	0	1	0
B		1	0	1	0
S	1	0	1	0	0
D		0	1	1	0
F	0	1	1	1	0
R	1	1	0	1	0

Figure.8. 4-bit addition

Consider another example with 8 bit,

A=9	0	0	0	0	1	0	0	1
B=78	0	1	0	0	1	1	1	0
S	0	1	0	1	0	1	1	1
D=57	0	0	1	1	1	0	0	1
F	1	1	0	0	0	1	1	1
R=S+57	1	0	0	1	0	0	0	0

Figure.9. 8-bit addition

The order of the basic gates in the first stage will change depending on the constant that is being added. The cells only consist of very basic gates with a multiplexer at every bit position choosing the appropriate flag bit depending on the carry out at the corresponding bit position. Thus, the new block called flagged inversion cells, is the extra hardware required in the final stage of the adder to compute the final output.

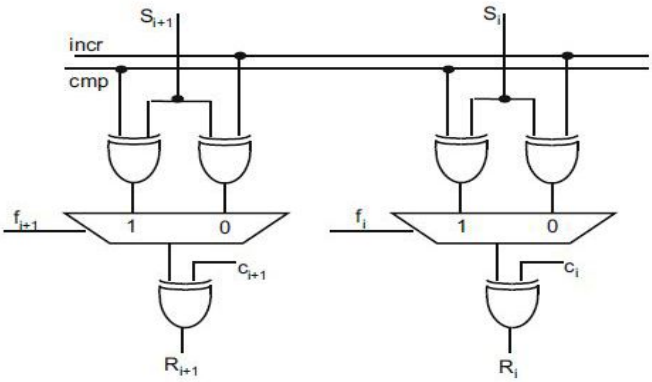

Figure.10. Flag inversion cells

The signals, incr and cmp are used to select the appropriate result from all the different possible results. The minimal amount of additional hardware therefore comprises of two levels of XOR gates and a multiplexer, thereby affecting the critical path minimally. The same can be accomplished using the carry skip adders. The carry skip adders can conveniently incorporate the computation of flag bits since, it generates the bit propagate signals for every bit position.

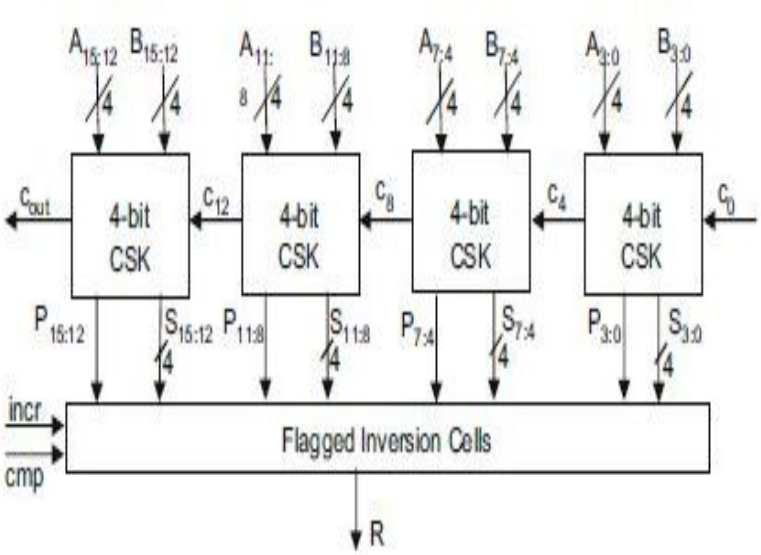

Figure.11. Flagged carry skip adder

The flagged carry skip adder uses the flag inversion cells, in this the group propagate signals are generated inside the carry skip block are used as the carry bits. The flag bits are computed by utilizing the equations. In all the adder circuits the flag inversion cells are added at the final stage to perform three-bit addition.

IV.SIMULATION RESULTS

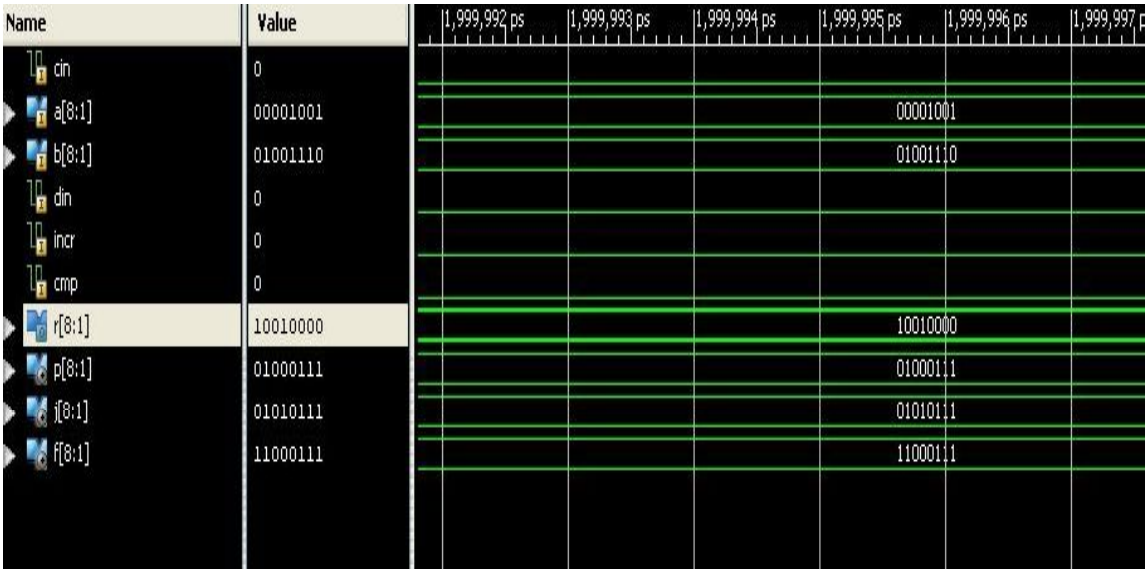

Figure.12.The Simulation Output Carry Skip Adder

The Carry skip adder circuit is implemented and the simulation result is verified for the three-bit adder circuit. Further all the adder circuits can be easily implemented by this logic and can be easily used for various applications. The Carry Select adder can also be implemented by this technique and then compared with Carry save adder.

V.RESULT

ADDERS	Two operand addition		Three operand addition		Three operand addition by flag bits (Proposed System)	
	AREA No. of LUTs	DELAY (ns)	AREA No. of LUTs	DELAY (ns)	AREA No. of LUTs	DELAY (ns)
CARRY SKIP ADDER	20	10.4	30	11.966	28	11.404
CARRY SAVE ADDER	-	-	17	10.506	14	10.014
CARRY SELECT ADDER	13	10.052	20	11.056	18	10.632

Figure.13. Comparison table

Comparing three adders, Carry save adder having less delay and high area consumption. The delay percentage 0.95 is reduced by using this Enhanced Flagged Binary Adder in Carry Skip adder. The number of LUT's used is also reduced by the use of flag bits. This Enhanced Flagged Binary adder is analyzed using VHDL Code in Xilinx ISE Design Suite 14.2.

VI.CONCLUSION

The main aim of the Carry skip adder is to reduce the worst case delay by reducing the number of full adder modules through which the carry has to be propagated. Therefore the adder consists of small groups of ripple carry adders, modified to include the skip network, which leading to a less attractive performance in speed. The Carry Select adder consists of a pair of Carry propagation adders for each group, which leading to significant area consumption. The area, delay and power estimation also obtained from this adder design. A comparison has also been made with the conventional design and the flagged architectures, to study the impact of additional hardware and the critical path delay and area consumption.

REFERENCES

[1] Vibhuti Dave, Erdal Oruklu and Jafar Sanile, "Constant addition with flagged binary adder architectures", Integration, VLSI Journal 43(2010) 258-267.

[2] Quelle: Timo Hamalainen, Computer arithmetic: Carry –skip adder.

[3] R. Zimmemen, "Binary adder architectures for cell based VLSI and their synthesis", Ph.D. dissertation, Swiss Federal Institute of Technology,Zurich,1997.

[4] V. Dave, E. Oruklu and J. Sanile," Design and Synthesis of Flagged Binary adder with constant addition" in: proceedings of the 49th IEEE International midwest Symposium on circuits and Systems,Vol.1.pp.23 to 27,2006

[5] V.Dave, "High-speed multi-operand addition utilizing flag bits" Ph.D. dissertation, Illinois Institute of Technology,Chicago,2007.

[6] "Improved carry select adder with reduced area and low power consumption" International Journal of Computer applications(0975-8887)volume .no.4,June 2010.

[7] J.Stine,c.Babb and V.Dave ,"Constant addition utilizing flagged prefix adders for constant addition",in:proceedings of the 6th IEEE International Conference on Electro/Information Technology,2006.