International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

Parallel Prefix Han-Carlson Adder

Priyanka Polneti,P.G.STUDENT,Kakinada Institute of Engineering and Technology for women, Korangi.

TanujaSabbeAsst.Prof, Kakinada Institute of Engineering and Technology for women, Korangi.

Abstract: Binary addition is one of the most important arithmetic functions in modern digital VLSI systems. Adders are extensively used as DSP lattice filter where the ripple carry adders are replaced by the parallel prefix adder to decrease the delay. The requirement of the adder is that it is fast and secondly efficient in terms of power consumption and chip area. Parallel prefix adder is a technique for improving the speed of the addition. Parallel prefix adders provide a good theoretical basis to make a wide range of design tradeoffs in terms of area, delay and power. This technique is more suited for adders with wider word lengths. In this paper, a modified Parallel Prefix Han-Carlson Adder is introduced which uses different stages of Brent-Kung and Kogge-Stone adders which reduces the complexity of the adder design.

Keywords - Parallel Prefix Adders, Han-Carlson Adder, area, prefix computation, Power Consumption, delay

I Introduction:

VLSI binary adders are critically important elements in processor chips, they are used in floating-point arithmetic units, ALUs, memory addresses program counter update and magnitude comparator [1, 2]. Adders are extensively used as a part of the filter such as DSP lattice filter [3]. Ripple carry adder is the first and most fundamental adder that is capable of performing binary number addition. Since its latency is proportional to the length of its input operands, it is not very useful. To speed up the addition, carry look ahead adder is introduced. Parallel prefix adders provide good results as compared to the conventional adders.

The adders with the large complex gates will be too slow for VLSI, so the design is modularized by breaking it into trees of smaller and faster adders which are more readily implemented. For large adders the delay of passing the carry through the look-ahead stages becomes dominated and therefore tree adders or parallel prefix adders are used. High speed adders depends on the previous carry to generate the present sum. In integer addition any decrease in delay will directly relate to an increase in throughput. In nanometer range, it is very important to develop addition algorithm that provide high performance while reducing power.

Parallel prefix adders are suitable for VLSI implementation since they rely on the use of simple cells and maintain regular connection between them. We can define each prefix structures in terms of logic levels, fanout and wiring tracks. Zero or more inverters are added to each prefix cell output to minimize the delay based on this model, buffers are individually sized to minimize the delay,

buffers are used to minimize the fanout and loading on gates since high fanout causes poor performance. A modified Han-Carlson adder uses fewer number of prefix operations by adjusting the number of stages amongst Kogge-Stone and Brent-kung adder and thus reduces the area required by the adder circuitry [4].

There are three stages in performing prefix computation as shown in "Fig.1" below. First is the pre-processing stage to calculate generate and propagate bit, second stage is the carry computation stage to compute the carry bit and the third stage is the post-processing stage to compute the sum bit.

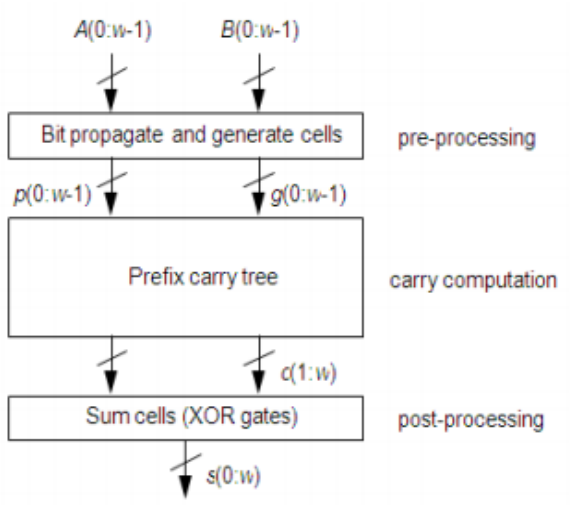


FIG 1: Parallel prefix adder structure

The graph representation of Hybrid Han-Carlson Adder is shown in Fig.2 below

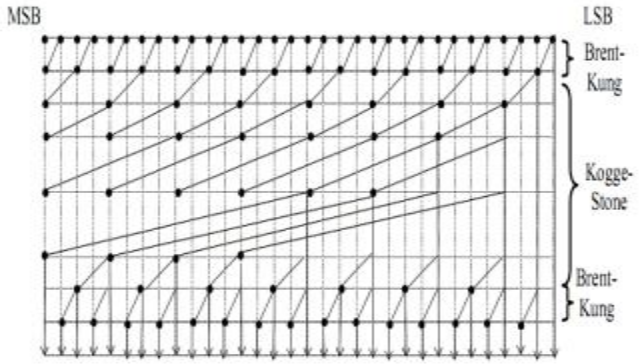


FIG 2: Graph representation of 32-bit han Carlson adder

II Previous Work:

In prefix addition, we use three stages to compute the sum: pre-processing, prefix-processing and post-processing. In the pre-processing stage the generate and propagate signal are computed as:

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$
$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$$

$$g_i = a_i b_i$$
$$p_i = a_i \oplus b_i$$

The concept of generate and propagate can be extended to a block of contiguous bits, from bit to bit (with) as follows:

$$g_{[i:k]} = \begin{cases} g_i & \text{if } i = k \\ g_{[i:j]} + p_{[i:j]} g_{[l:k]} & \text{otherwise} \end{cases}$$
$$p_{[i:k]} = \begin{cases} p_i & \text{if } i = k \\ p_{[i:j]} p_{[l:k]} & \text{otherwise} \end{cases}$$

The condition means that a carry is generated in the block, while the condition means that a carry is propagated through the block.

The different types of parallel prefix adders available are Kogge-Stone adder, Brent-kung adder, Sklansky adder, Han-Carlson adder, Knowles adder and Ladner-Fischer adder. These adders offer a tradeoff among the number of stages of logic, the number of logic gates, fanout and amount of wiring between stages. Kogge-Stone adder, Brent-kung adder and Sklansky adder are the fundamental adders. Brent-Kung uses minimal number of computation nodes which yields in reduced area but structure has maximum depth which yields slight increase in latency. Slansky reduces the delay at the expense of increased fanout. Kogge-Stone achieves high speed and low fanout but produces complex circuitry with more numbers of wiring tracks [5]. The Knowles trees are family of network between betweenKogge-Stone and Sklansky with increased fanout. Ladner Fischer introduced a network between Sklansky and Brent-Kung which provides a tradeoff between logic levels and fanout. T. Han and D.A. Carlson presented a hybrid construction of a parallel prefix adder using two designs the Kogge-Stone construction having the best feature of higher speed and the Brent-kung construction with best feature of low area requirement. A modified Han-Carlson adder uses fewer number of prefix operations by adjusting the number of stages amongst Kogge-Stone and Brent-kung adder and thus reduces the area required by the adder circuitry.

Fig 2. below shows a 3-dimentional taxonomy of tree adders [6]. There are three axes representing the fanout, wiring tracks and logic levels and each tree is indicated by three integers (l, f, t) in the range [0, L-1]. The tree adders lie on the plane l + f + t = L-1, where L= log2N and indicates the number of bits. Brent-Kung, Kogge-Stone and Sklansky represent the vertices of the cube (3, 0, 0), (0, 0, 3) and (0, 3, 0) respectively. Han-Carlson, Ladner-Fischer and Knowles lie along the diagonals. Where N indicates the number of bits the variables l, f, and t are integers in the range [0, L−1] indicating:

Logic Levels: L+ l

Fanout: 2f +1

Wiring Tracks: 2t

III Speculative Prefix-Processing:

The speculative prefix-processing stage is one of the main differences compared with the standard prefix adders recalled in previous section. Instead of computing all the and required in (8) to obtain the exact carry values, only a subset of block generate and propagate signals is calculated; in the postprocessing stage approximate carry values are obtained from this subset. The output of the speculative prefix-processing stage will also be used in the error detection and in the error correction stages discussed in the following.

The basic assumption behind speculative prefix-processing stage is that carry signals propagate for no more than bits, with and. This assumption is corroborated by the analyses in [13], [17] that demonstrate that having a propagate chain longer that is a very rare event.

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

3.1Kogge-Stone Topology:

The Kogge-Stone speculative prefix-processing stage has been proposed in [12], [13] and can be obtained by pruning the last levels of a traditional Kogge-Stone adder. In the example shown in Fig. 2, the last level of a bit Kogge-Stone adder is pruned. As it can be observed, for the length of propagate chains extends for 8 bits, resulting in a speculative prefix-processing stage

In general, one has, where is the number of pruned levels; the number of levels of the speculative stage is correspondingly reduced from to (assuming that is a power of two).

In general, the computed propagate and generate signals for the speculative Kogge-Stone architecture are:

3.2 Post-Processing:

In the post-processing stage we firstly compute the approximate carries, , and then use them to obtain the approximate sum bits as follows:

$$\tilde{s}_i = p_i \oplus \tilde{c}_{i-1}$$

The approximate carries are obtained as the generate signals available in the last level of the prefix-processing stage. We have:

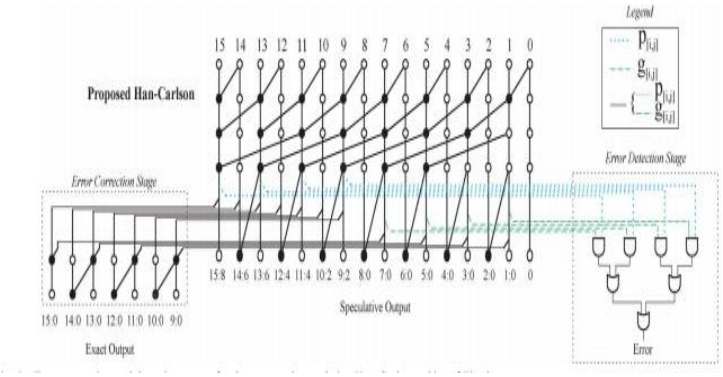
$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \leq K-1 \\ g_{[i:i-K+1]} & \text{otherwise} \end{cases} \quad (\text{Kogge} - \text{Stone})$$

and:

$$\tilde{c}_i = \begin{cases} g_{[i:0]} & \text{for: } i \leq K \\ g_{[i:i-K+1]} & \text{for: } i > K,\ i \text{ odd} \\ g_{[i:i-K]} & \text{for: } i > K,\ i \text{ even} \end{cases} \quad (\text{Han} - \text{Carlson})$$

3.3 Error Detection:

The conditions in which at least one of the approximate carries is wrong (misprediction) are signaled by the error detection stage. In case of misprediction, an error signal is asserted by error detection stage and the output of the post-processing stage is discarded. The error correction stage will give the correct sum in the next clock period.



Error correction and detection stages for the proposed speculative Han Carlson adder

3.3.1 Han-Carlson:

The error condition for carry can be obtained as:

$$e_i = \begin{cases} 0 & \text{for: } i \leq K \\ p_{[i:i-K+1]} g_{[i-K:0]} & i > K,\ i \text{ odd} \\ p_{[i:i-K]} g_{[i-K-1:0]} & i > K,\ i \text{ even} \end{cases}$$

The error signal can be written as:

$$E_{HC} = \sum_{\substack{i=K+1 \\ i \text{ odd}}}^{n-1} p_{[i:i-K+1]} g_{[i-K:0]} + \sum_{\substack{i=K+1 \\ i \text{ even}}}^{n-1} p_{[i:i-K]} g_{[i-K-1:0]}$$

It can easily be seen that in (26) the terms in the second OR are implied by the terms in the first OR. Let us consider, for instance, the first two terms of the OR (assuming that is even). We have:

$$p_{[K+1:2]}g_{[1:0]} + p_{[K+2:2]}g_{[1:0]} = p_{[K+1:2]}g_{[1:0]}$$

As it can be observed, in Kogge-Stone some of the checking cells are at the last level of the graph; their output signals are available after three black cells delay. In Han-Carlson the critical checking cells are in the second last level of the graph and are also available after three black cells delay, in spite of the larger number of levels of the Han-Carlson prefix-processing stage. From the above observations, it can be concluded that error detection is sensibly simplified and potentially faster in HanCarlson, compared to Kogge-Stone. As an additional note, the need of driving the gates of the error detection stage increases the fanout of the checking cells, slowing the speculative prefix-processing stage.

3.4 Error Correction:

The error correction stage computes the exact carry signals (9), to be used in case of misprediction. The error correction stage is composed by the levels of the prefix-processing stage pruned to obtain the speculative adder. The Fig. 5 shows the error correction stage of the proposed speculative Han-Carlson adder; the error correction for Kogge-Stone topology can be obtained similarly. It can be observed that the inclusion of the error correction stage increases the fanout of some of the cells of the speculative prefix-processing stage, with adverse effect on adder speed.

TABLE1

SPATIAL AND TIMING COMPLEXITY

Speculative Adder	N_{or}	Spatial Complexity	Timing Complexity	
			Speculative Sum	*Error detection*
Han-Carlson (proposed)	$\frac{n-K}{2}$	$\{n\log_2 n+n-K-2\}+[n-K]$	$2\log_2 K+6$	$2\log_2 K+\log_2(n-K)+2$
Kogge-Stone	$n-K$	$\{2n\log_2 n-2n+2\}+[2(n-K)]$	$2\log_2 K+4$	$2\log_2 K+\log_2(n-K)+3$
Non-Speculative Adder	N_{or}	**Spatial Complexity**	**Timing Complexity**	
			Sum	
Han-Carlson	n.a.	$n\log_2 n$	$2\log_2 n+6$	
Kogge-Stone	n.a.	$2n\log_2 n-2n+2$	$2\log_2 n+4$	

TABLE 2

ERROR PROBABILITY VALUES

Speculative Adder	n	*K=8*		*K=16*	
		Precise	*Coarse*	*Precise*	*Coarse*
Han-Carlson (proposed)	32	1.74×10^{-2}	3.58×10^{-2}	4.57×10^{-5}	9.53×10^{-5}
Kogge-Stone	32	2.32×10^{-2}	4.82×10^{-2}	6.09×10^{-5}	1.29×10^{-4}
Han-Carlson (proposed)	64	4.06×10^{-2}	8.05×10^{-2}	1.37×10^{-4}	2.78×10^{-4}
Kogge-Stone	64	5.39×10^{-2}	1.07×10^{-1}	1.83×10^{-4}	3.74×10^{-4}
Han-Carlson (proposed)	128	8.45×10^{-2}	1.63×10^{-1}	3.21×10^{-4}	6.44×10^{-4}
Kogge-Stone	128	1.11×10^{-1}	2.12×10^{-1}	4.27×10^{-4}	8.61×10^{-4}

3.5 Post-Processing:

The approximate carries are already available at the output of the prefix-processing stage. The post-processing, according to (14), is equal to the one of a non-speculative adder and consists of xor gates.

Comparison between variable latency adder and the non-speculative Han-Carlson topology reveal that variable latency adders allow to reduce the minimum achievable delay. For instance, in the 64-bit case, the minimum achievable delay is about 280 ps for the non-speculative adder and reduces up to 225 ps in the variable latency architecture.

To design Parallel Prefix Hybrid Han-Carlson Adder. It differs from other adder in that it can be used for large word sizes. The proposed design reduces the number of prefix operation by using more number of Brent-Kung stages and lesser number of Kogge-Stone stages. This also reduces the complexity, silicon area and power consumption significantly.

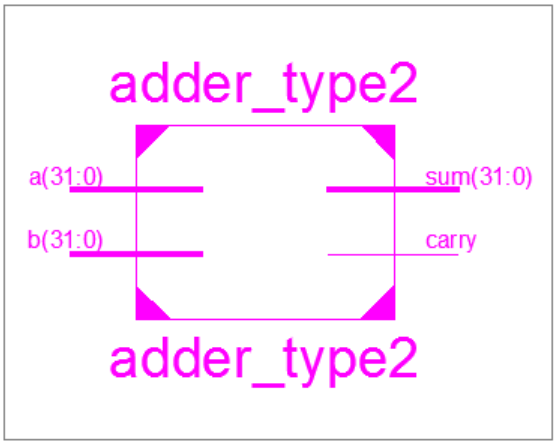
SPECULATIVEPARALLEL PREFIX-PROCESSING

The designing of proposed adder architecture is done using Xilinx ISE 13.1 Tool and the complete source code for 32 bit implementation of proposed adder is done. The design is implemented using Spartan 6 device. The basic elements of the design is modeled as components which are independently functional. These are then wired together by means of signals to construct the structure of the adder. The design is implemented using the Spartan 6 device.

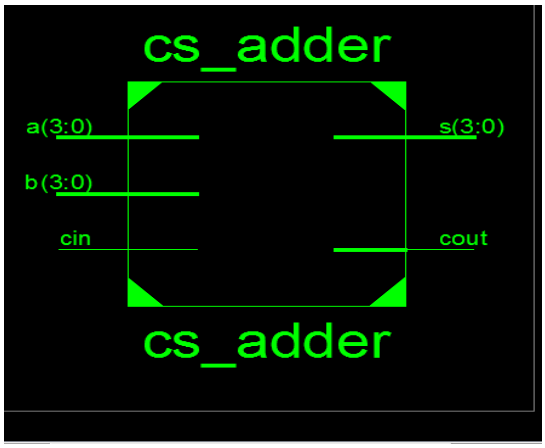
TABLE 3

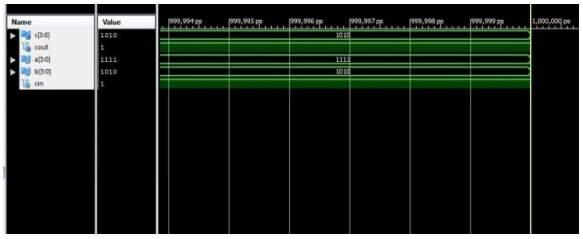
	CLA	KOGGE-STONE	HCA
Delay	22.95ns	21.82ns	20.82ns
Number of Slices	1%	1%	1%
Number of 4 i/p LUT's	1%	1%	1%
Number of I/O	23%	21%	21%

FIG. 3 below shows the simulation result for 32 bit Hybrid Han-Carlson adder.



PARALLEL RTX





Result: The simulation results of Han-Carlson adder and HybridHan-Carlson adder indicates that the Hybrid Han-Carlson Adder provides better results as compared to Han-Carlsonadder. Table below gives the comparison of these twodesigns in terms of area, delay and power.

Table 1Comparison of Han-Carlson Adder and Hybrid Han-CarlsonAdder
TABLE 4

Design Parameters		Han-Carlson Adder(N=32)	Hybrid Han-Carlson Adder(N=32)
Prefix Operations		80	63
No of gates		940	855
Delay(ns)		0.68	0.76
Power (Microwatt)	Dynamic Power	445.98	405.23
	Leakage Power	7.45	6.79

Studied the structure of Hybrid Han-Carlson Adder and various design parameters. Also studied different Prefix cells that are being used in the design and their equations [4] are shown below(spacing)

Square Cells: for pre-processing parallel prefix stage to calculate generate and propagate.

g= a and b

p= a xor b

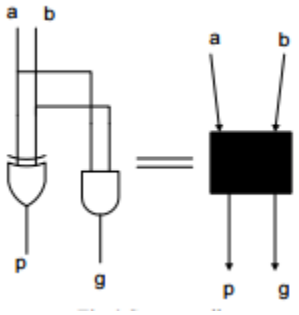
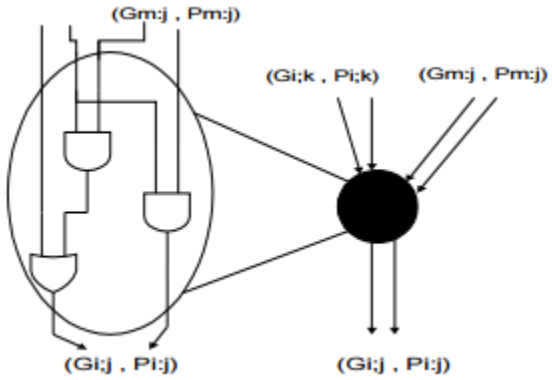


FIG 3: Square cell structure

Circular cells: for computation of prefix operation [7,8]

(gi . p i) o (gj . pj) = (gi+ pi . gj , pi . pj)



The simulation results of Han-Carlson adder and Hybrid Han-Carlson adder indicates that the Hybrid Han-Carlson Adder provides better results as compared to Han-Carlson adder.

IV Conclusion

From the above work, it is seen that the Han-Carlson adder presented a reduction in the complexity and hence provides a tradeoff for the construction of large adders. These wide adders are useful in applications like cryptography for security purpose, global unique identifiers used as a identifier in computer software and this wide adder also provides good speed.

References:

[1] S. Veeramachaneni, M. K. Krishna, L. Avinash, P. Sreekanth Reddy, and M. B. Srinivas, "Efficient design of 32-bit comparator using carry look-ahead logic," in Proceedings of the IEEE North-East Workshop on Circuits and Systems (NEWCAS '07), pp. 867–870, August 2007.

[2] M. Nesenbergs and V. O. Mowery, "Logic synthesis of high speed digital comparators," Bell System Technical Journal, vol.38, pp. 19–44, 1959.

[3] DeepaYagain, Vijaya Krishna A and AkanshaBaliga "Design of High-Speed Adders for Efficient Digital Design Blocks", 2012.

[4] SreenivaasMuthyala Sudhakar, Kumar P. Chidambaram and Earl E. Swartzlander Jr. "Hybrid Han-Carlson Adder "The University of Texas at Austin, 2012.

[5] D. Harris, "A Taxonomy of Parallel Prefix Networks,"inProc. 37thAsilomar Conf. Signals Systems andComputers, pp. 2213–7, 2003.

[6] Neil H.E. Weste, David Harris, Ayan Banerjee, "CMOS VLSI Design ," Third Edition.

[7] GiorgosDimitrakopoulos and Dimities Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders", IEEE Trans. On Computer, Vol. 54, No. 2, February 2005.

[8] DeepaYagain, Vijaya Krishna A, " High Speed Digital Filter Design using register Minimization Timing & Parallel Prefix Adders.",2011

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

TanujaSabbe received Bachelordegree in Electronics and CommunicationEngineering fromKakinada Institute of Engineering and technology for women(JNTUK) and awarded M.Techdegree in Computers and Communications from Jawaharlal Nehru Technological university,kakinada.

Priyanka polneti pursuingM.Tech VLSI&Embedded systems in Kakinada Institute of Engineering Technology for womenkorangi. She received Bachelor degree inDepartment of Electronics and Communication Engineering from Kakinada institute of engineering and technology for women.