ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 2, February 2016

Performance Enhancement of Han-Carlson Adder

Subha Jeyamala K[2], Aswathy B.S[1]

Abstract:- To make addition operations more efficient parallel prefix addition is a better method. In this paper 16-bit parallel prefix addition has been implemented with the help of cells like black cell and white cell operations for carry generation and propagation. This process gives high speed computations with high fan-out and makes carry operations easier. This paper presents different types of parallel prefix adders and compares them with the Simple Adder. The adders are designed using Verilog HDL code and simulated and synthesized using Xilinx design suite 14.5 software tool and Modelsim altera 10.0c. In order to make it suitable for FPGA implementation, Han Carlson adder is modified using fast carry logic technique. The modified adder provides better performance over the Simple adder for the higher order bit widths.

Index terms - FPGA, Binary addition, Carry tree adders, Prefix computation, Prefix addition.

I INTRODUTION

The decimal numbers are easy to comprehend and implement for performing arithmetic. However, in digital systems, such as a microprocessor, DSP (Digital Signal Processor) or ASIC (Application-Specific Integrated Circuit), binary numbers are more pragmatic for a given computation. This occurs because binary values are optimally efficient at representing many values[2]. Binary adders are one of the most essential logic elements within a digital system.

Aswathy B S[1] *currently working as Assistant Professor in ECE Department at Dr.Sivanthi Aditanar College of Engineering, Tiruchendur She received her ME Degree in VLSI Design from Sethu Institute of Technology, Kariapatti under Anna University Chennai and her BE Degree in ECE from National Engineering College, Kovilpatti under Anna University Chennai.*

Subha Jeyamala K[2] *received her Bachelor of Engineering degree in Electronics and Communication Engineering from Sethu Institute of Technology under Anna University chennai, Kariapatti, Viruthunagar. Currently pursuing Master of Engineering in Very Large Scale Integration (VLSI) from Dr. Sivanthi Aditanar College of Engineering under Anna University,Chennai*

In addition, binary adders are also helpful in units other than Arithmetic Logic Units (ALU), such as multipliers, dividers and memory addressing[3]. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, hence, help improve the performance of the entire system. The major problem for binary addition is the carry chain[4]. As the width of the input operand increases, the length of the carry chain increases. In order to improve the performance of carry-propagate adders, it is possible to accelerate the carry chain, but not eliminate it. Consequently, most digital designers often resort to building faster adders when optimizing computer architecture, because they tend to set the critical path for most computations.

II PRELIMINARIES

Parallel-prefix structures are found to be common in high performance adders because of the delay is logarithmically proportional to the adder width. Such structures can usually be divided into three stages as follows: precomputation, prefix tree and post computation.

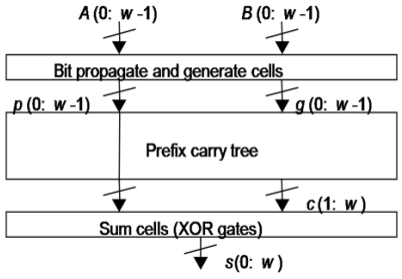

Fig1Parallel prefix adder structure

An example of a parallel-prefix structure is shown in Fig 1. In the prefix tree, generate/propagate are the only signals used. The group generate/propagate equations are based on single bit generate/propagate, which are computed in the pre-computation stage.

$$g_i = a_i \,.\, b_i \qquad (1)$$

$$p_i = a_i \oplus b_i \qquad (2)$$

Where $0 \leq i \leq n$. $g_{-1} = c_{in}$ and $p_{-1} = 0$. Sometimes, p_i can be computed with OR logic instead of an XOR gate. The OR logic is mandatory especially when Ling's scheme [5] is applied. Here, the XOR logic is utilized to save a gate for temporary sum t_i. In the prefix tree, group generate/propagate signals are computed at each bit.

$$G_{i:k} = G_{i:j} + P_{i:j} \cdot G_{j-1:k} \quad (3)$$

$$P_{i:k} = P_{i:j} \cdot P_{j-1:k} \quad (4)$$

$$(G_{i:k}, P_{i:k}) = (G_{i:j}, P_{i:j}) \circ (G_{j-1:k}, P_{j-1:k}) \quad (5)$$

More practically, Equation (5) can be expressed using a symbol " $\circ$ " denoted by Brent and Kung. Its function is exactly the same as that of a black cell.

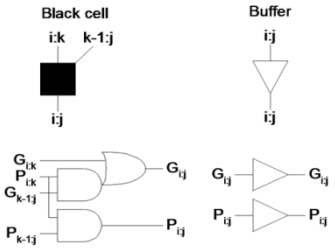

Fig 2 Cell structures

In Fig1.2, various cell structures are represented. The " $\circ$ " operation will help make the rules of building prefix structures. In the post-computation, the sum and carry-out are the final output.

$$s_i = p_i \cdot g_{i-1:-1} \quad (6)$$

$$c_{out} = G_{n:-1} \quad (7)$$

where "-1" is the position of carry-input. The generate/propagate signals can be grouped in different fashion to get the same correct carries. Based on different ways of grouping the generate/propagate signals, different prefix architectures can be created.

III KOGGE STONE ADDER FOR 16 BIT

Kogge Stone prefix tree is among the type of prefix trees that use the fewest logic levels.

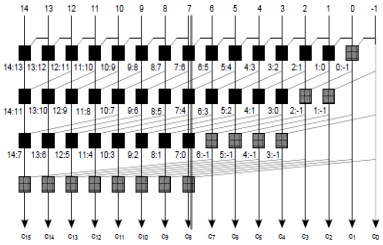

Fig 3 16-bit Kogge-Stone prefix tree

A 16-bit example is shown in Fig 3. In fact, Kogge-Stone is a member of Knowles prefix tree. The 16-bit prefix tree can be viewed as Knowels [1,1,1,1]. The numbers in the brackets represent the maximum branch fan-out at each logic level. The maximum fan-out is 2 in all logic levels for all width Kogge-Stone prefix trees.

Gray cells are inserted similar to black cells except that the gray cells final output carry outs instead of intermediate G/P group. The reason of starting with Kogge-Stone prefix tree is that it is the easiest to build in terms of using a program concept.

For the Kogge-Stone prefix tree, at the logic level 1, the inputs span is 1 bit (e.g. group (4:3) take the inputs at bit 4 and bit 3). Group (4:3) will be taken as inputs and combined with group (6:5) to generate group (6:3) at logic level 2. Group (6:3) will be taken as inputs and combined with group (10:7) to generate group (10:3) at logic level 3, and so on so forth. With this inspection, the structure can be described with the Algorithm 1.1 listed below.

Algorithm 1.1 Building Kogge-Stone Prefix Tree

$L = log_2(n)$;
for *llevel = 1; llevel* $\leq L$; *llevel ++* ***do***
$u = 2^{llevel}$; *{output bit span}*
$v = 2^{llevel-1}$; *{input bit span}*
for $i = v - 1; i < n - 1; i$ ++ ***do***
$GP_{i:i-u+1} = (GP_{i:i-v+1}) \circ (GP_{i-v:i-u+1})$
end for
end for

IV HAN CARLSON ADDER FOR 16 BIT

The idea of Han-Carlson prefix tree is similar to Kogge-Stone's structure since it has a maximum fan-out of 2 or f = 0. The difference is that Han-Carlson prefix tree uses much less cells and wire tracks than Kogge-Stone. The cost is one extra logic level. Han-Carlson prefix tree can be viewed as a sparse version of Kogge-Stone prefix tree. In fact, the fan-out at all logic levels is the same (i.e. 2). The pseudo-code for Kogge-Stone's structure can be easily modified to build a Han-Carlson prefix tree. The major difference is that in each logic level, Han-Carlson prefix tree places cells every other bit and the last logic level accounts for the missing carries.

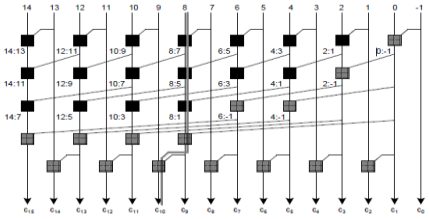

Fig 4 16-bit Han-Carlson prefix Tree

Figure 4 shows a 16-bit Han-Carlson prefix tree, ignoring the buffers. The critical path is shown

with thick solid line. A good trade-off between fanout, number of logic levels and number of black cells is given by Han-Carlson. The outer rows of the Han-Carlson topology are Brent-Kung graphs, while the inner rows are Kogge-Stone graphs.

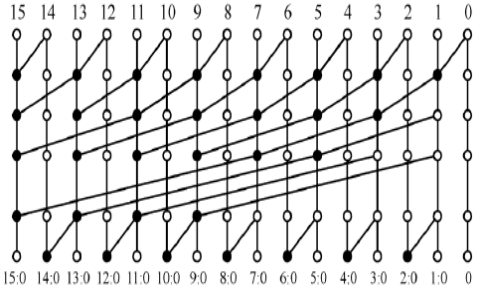

Fig 5 16-bit Han- Carlson topology

The Han-Carlson adder in Fig 5 uses a single Brent-Kung level at the beginning and at the end of the graph, and the number of levels is $1 + log_2(n)$. Here black dots represent the prefix operator, while white dots are simple placeholders. Han-Carlson adder constitutes a good trade-off between fanout, number of logic levels and number of black cells. Because of this, Han-Carlson adder can achieve equal speed performance respect to Kogge-Stone adder, at lower power consumption and area. Therefore it is interesting to implement a speculative Han Carlson adder. Moved by these reasons, we have generated a Han-Carlson speculative prefix-processing stage by deleting the last rows of the Kogge-Stone part of the adder.

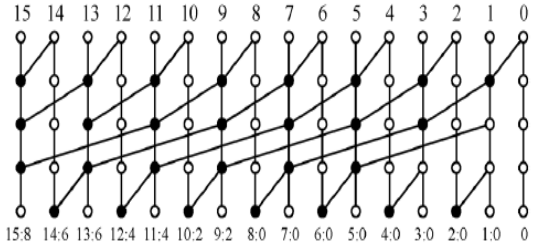

Fig 6 Han-Carlson speculative prefix-processing stage

As an example, the Fig 6 shows the Han-Carlson adder in which the two Brent-Kung rows at the beginning and at the end of the graph are unchanged, while the last Kogge-Stone row is pruned. In general, one has $K=n/2^P$ where P is the number of pruned levels; the number of levels of the speculative Han-Carlson stage reduces from $1 + log_2(n)$ to $1 + log_2(K)$ (assuming K that is a power of two).

In general, the computed propagate and generate signals for the speculative Han-Carlson architecture are:

$(g,p)_{i:0}$ for: i ≤ K
$(g,p)_{i:i-K+1}$ for: i > K, i odd
$(g,p)_{i:i-K}$ for: i > K, i even

V HAN-CARLSON ADDER WITH ERROR DETECTION AND ERROR CORRECTION

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. The error correction stage computes the exact carry signals, 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.

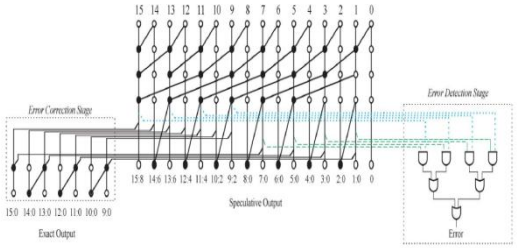

Fig 7 Error correction and detection stages for the proposed speculative Han-Carlson adder

The Fig 7 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.

VI SIMULATION AND RESULT

The Ideal N-bit tree adder would have:

- L= log N logic levels
- Fan-out of 2
- No more than one wiring track between levels

Kogge-Stone, Han-Carlson and Knowles adders require a large number of parallel wiring for wide bit adders. Thus packing the wires close together will increase the coupling capacitance on each wire. Sklansky architecture becomes slow due to its high fan-out. When interconnect is considered Han-Carlson become attractive one as it requires only half the number of columns. Individually specifications are like *Kogge-Stone* has least logic levels but hard to P and G. *Brent-Kung* is the very first and bad-one. *Ladner-Fischer* has a bit more logic levels and high fan-out. *Han-Carlson* has more logic levels but less cells. *S. Knowles* possesses many cells and wires and some fan-out. *Sklansky* has least logic levels and highest fan-out. If wire capacitance is neglected *Kogge-Stone adder is the best among the others.*

Simulation and synthesis is done using Xlinx design suite 14.5, selecting device Spartan-3E FPGA (XC3S500E-4FG320C). The adder abbreviations used in the following discussions are: RCA for the ripple carry adder, KSA for the Kogge-Stone adder, and HCA for the Han-Carlson adder. Table.1 shows the synthesis results of different adders.

Table 4.1 Comparison table on RCA, KSA and HCA

PARAMETER	RIPPLE CARRY	KOGGE STONE	HAN CARLSON
Number of slices	2%	4%	2%
Number of 4 input LUTs	2%	3%	2%
Number of I/Os	40%	63%	51%
Estimation of timing	33.378ns	15.946ns	15.460ns

In Fig 8, it shows the Simulation and synthesis of Han-Carlson adder using ModelSim altera 10.0c for 16 bit binary variable.

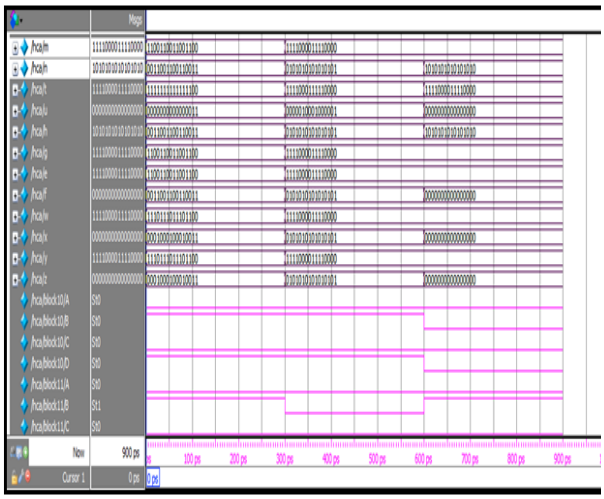

Fig 8 Simulation result of Han-Carlson adder

In Fig 9, it shows the Simulation and Synthesis of Han-Carlson adder using Xilinx design suite 14.5 for 16 bit binary variable.

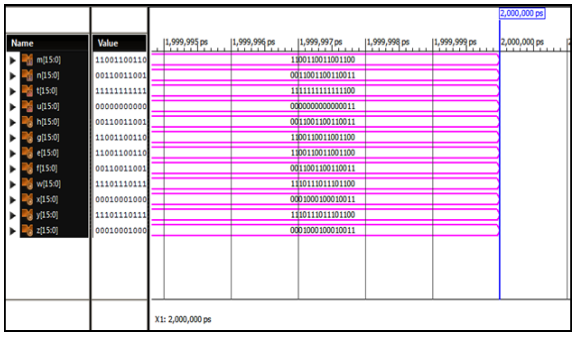

Fig 9 Simulation result of Han Carlson adder

The following are RTL schematic of the kogge stone adder and the han Carlson adder.

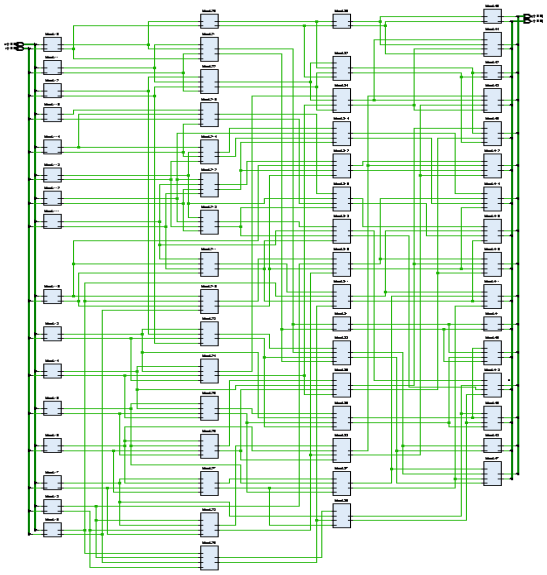

Fig 10 RTL schematic of Kogge Stone adder

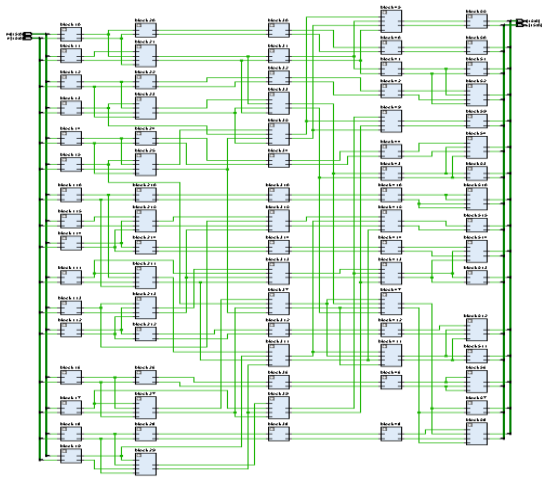

Fig 11 RTL schematic of Han Carlson adder

VII CONCLUSION

The variable latency Han-Carlson parallel prefix speculative adder for high-speed application is proposed. A new, more accurate, error detection

network is introduced, which allows reducing the error probability compared to the other approaches. Compared with traditional, non-speculative, adders, our analysis demonstrates that variable latency Han-Carlson adders show sensible improvements when the highest speed is required; otherwise the burden imposed by error detection and error correction stages overwhelms any advantage. This can be used in various applications like digital signal processing, satellite and mobile phones.

The 16 Bit Existing and Proposed adders are implemented. In the future work, proposed architecture will be converted into 32 bit Adders and analyzes the area and speed. This adder will be done by using our verilog and the Adder design will implemented into FPGA.

VII REFERENCES

[1] O. J. Bedrij, .Carry-select adder,. *IRE Trans. Electron. Comput.*, pp. 340.346, June 1962.

[2] R. P. Brent and H. T. Kung, .A regular layout for parallel adders,. *IEEE Trans.Computers*, vol. C-31, no. 3, pp. 260.264, Mar. 1982.

[3] J. Chen and J. E. Stine, .Optimization of bipartite memory systems for multiplicative divide and square root,. *48th IEEE International Midwest Symp. Circuits and Systems*, vol. 2, pp. 1458.1461, 2005.

[4] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in *Proc. Design, Autom., Test Eur. Conf. Exhib, (DATE'09)*, Apr. 2009, pp. 664–669.

[5] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Design, Autom., Test Eur. Conf. Exhib.(DATE '12)*, Mar. 2012, pp. 1257–1262.

[6] D. Goldberg, .What every computer scientist should know about floating point arithmetic,. *ACM Computing surveys*, vol. 23, no. 1, pp. 5.48, 1991.

[7] T. Han and D. Carlson, .Fast area-efficient VLS Adders,. in *Proc. 8th Symp. Comp. Arith.*, Sept. 1987, pp. 49.56.

[8] D. Harris, .A taxonomy of parallel prefix networks, in *Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, Nov. 2003, pp. 2213.2217.

[9] S. Knowles, .A family of adders,. in *Proc. 15th IEEE Symp. Comp. Arith.*, June 2001, pp. 277.281.

[10] P. Kogge and H. Stone, .A parallel algorithm for the efficient solution of a general class of recurrence relations,. *IEEE Trans. Computers*, vol. C-22, no. 8, pp. 786.793, Aug. 1973.

[11] I. Koren, *Computer Arithmetic Algorithms*. Natick, MA, USA: A K Peters, 2002.

[12] H. Ling, .High speed binary adder,. *IBM Journal of Research and Development*, vol. 25, no. 3, pp.156.166, 1981.

[13] R. Ladner and M. Fischer, .Parallel prefix Computation,.*J. ACM*, vol. 27, no. 4, pp. 831.838, Oct. 1980.

[14] J. Sklansky, .Conditional sum addition logic,. *IRE Trans. Electron. Comput.*, pp. 226. 231, June 1960.

[15] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proc. Comput. Digit. Tech.*, vol. 143, no. 5, pp. 301–307, Sep. 1996.

[16] V. G. Oklobdzija, B. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, Energy delay estimation technique for high-performance microprocessor VLSI adders,. *Proc. 16th IEEE Symp. Computer Arithmetic (ARITH-16'03)*, p. 272, June 2003.

[17] A. K. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *Proc. Design, Autom., Test Eur. (DATE '08)*, Mar. 2008, pp. 1250–1255.

[18] S. Winograd, .On the time required to perform addition,.*J. ACM*, vol. 12, no. 2,pp. 277.285, 1965.

[19] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. thesis, Swiss Federal Institute of Technology, (ETH) Zurich, Zurich, Switzerland, 1998, Hartung-Gorre Verlag.

ABOUT THE AUTHOR

Aswathy B S[1] currently working as Assistant Professor in ECE Department at Dr.Sivanthi Aditanar College of Engineering, Tiruchendur She received her ME Degree in VLSI Design from Sethu Institute of Technology, Kariapatti under Anna University Chennai and her BE Degree in ECE from National Engineering College, Kovilpatti under Anna University Chennai.

Subha Jeyamala K[2] received her Bachelor of Engineering degree in Electronics and Communication Engineering from Sethu Institute of Technology under Anna University chennai, Kariapatti, Viruthunagar. Currently pursuing Master of Engineering in Very Large Scale Integration (VLSI) from Dr. Sivanthi Aditanar College of Engineering under Anna University,Chennai

.