International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 10, October- 2014

Fast Performance of Parallel Adders using VLSI

Md Javeed Ahammed
Dept. of Electronics and Communication Engineering
Nimra College of Engineering and Technology, Student
Vijayawada, India

M. Muzammil Parvez
Dept. of Electronics and Communication Engineering
Nimra College of Engineering and Technology, Assistant Professor
Vijayawada, India

***Abstract*— In our daily life we use no of electronic devices such as calculators, computers etc. Every device needs arithmetic operations. This may create the complexity in the components. Now a days we have no of arithmetic operations which may reduce the complexity like Kogge-Stone, spars Kogge-Stone, and spanning tree adder and compares them to the simple Ripple Carry Adder (RCA) and Carry Skip Adder (CSA). This paper presents an attractive structure of parallel adders which gives the better delay performance and area utilization. This structure and fast performance makes them particularly attractive for VLSI implementations.**

Keywords—adder, kogge-stone adder, spars kogge- stone adder, delay performance.

I INTRODUCTION

Arithmetic operations in every electronic device are major operation for the performance. Arithmetic operations such as addition, subtraction, multiplication, division, etc. addition operation are the major issue for every performance. So, no of adders have been implemented by which the complexity of the systems reduces such as area utilization, time delay. These are the main problems in each and every electronic device.

Area and Time consumed by the circuit are the basic and important requirements. Numbers can be represented in digital circuits in various ways. Hence, developing efficient adder architecture is crucial to improving the efficiency of the design. Generally ripple carry adder uses for binary addition. After the design of ripple carry adder several techniques are used for the computation of parallel adders. Carry look ahead adders are based on parallel prefix computation gives the better performance than ripple carry adder. After many years research continuous to be focused on improving the delay performance of the adder. As such, extensive research continues to be focused on improving the delay performance of the adder. Next, Brent and Kung (BK) design parallel prefix-computation graph In an area-optimal way and the kogge-stone (KS) architecture is optimized for timing. This architecture is proposed based on KS and BK structures.

In this paper propose a new structure for parallel adders. Our proposed adder shows marginally faster performance than the regular kogge-stone adder with area saving.

II ADDERS

1. Kogge-Stone Adder

The 8-bit Kogge-Stone adder will be explained in detail in this subsection. An 8-bit Kogge-Stone adder is built from eight generate and propagate (GP) blocks, eight black cells (BC) blocks, eight gray cell (GC) blocks, and nine sum blocks as shown in the Figure. The details of the various blocks used in the structure of Kogge-Stone adder are discussed below.

GP block

The generate and propagate block takes a pair of operand bits (a, b) as inputs and computes a pair of generate and propagate signals (g, p) as output, as depicted in Figure. The output from this a block is shown in figure.

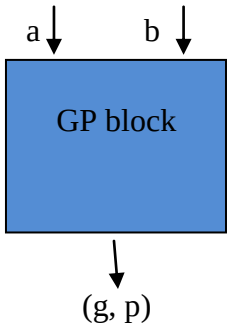


Fig. Generate and propagate block

BC block

The black cell takes two pairs of generate and propagate signals (gi, pi) and (gj ,pj) as input and computes a pair of generate and propagate signals (g,p) as output. It is shown in Figure.

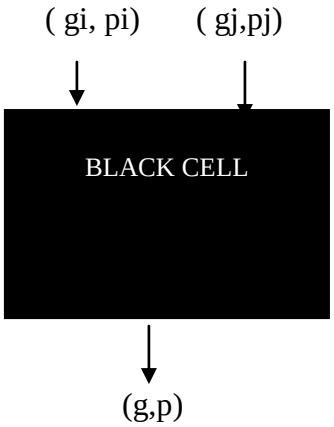


Fig. black cell

GC block

The gray cell takes two pairs of generate and propagate signals (gi,pi) and (gj,pj) as input and computes a pair of generate signal only. The output from this block is shown in figure.

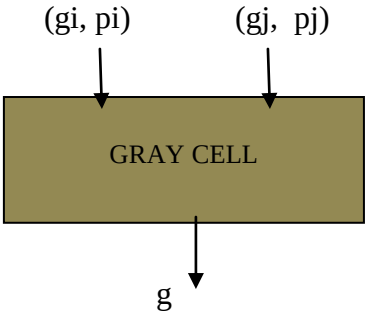


Fig. gray cell

Kogge-Stone adder design procedure

Step 1: First to generate propagation and generation signals for each bit.

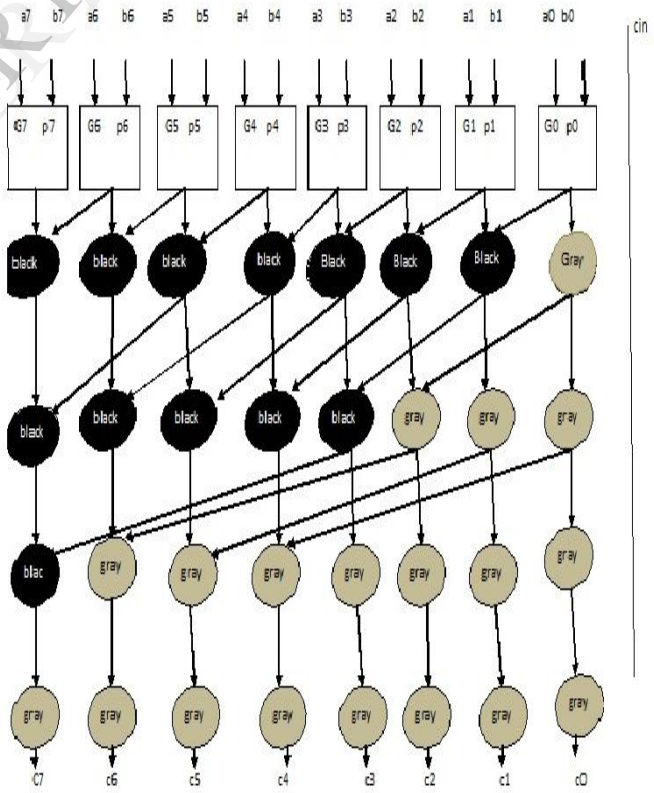
Step 2: To generate black cell and gray cell equations.

Step 3: In each step gray cells are generated.

Step 4: By using gray cell equation to generate the carry bits directly.

Step 5: To combine propagation and carry bits for each step to generate sum.

The Kogge-Stone adder is a parallel prefix form Carry-look ahead adder. It generates the carry signals in (log n) time, and is widely considered the fastest adder design possible. It is the common design for high-performance adders. An example of a 8-bit, 16-bit Kogge-Stone adder structures are shown in the figures. Each vertical stage produces a "propagate" and a "generate" bits as shown. In each step radix-2 gray cells are generated. By using the gray cells directly generate the carry values. The culminating generate bits (the carries) are produced in the last stage (vertically), and these bits are XOR with the initial propagate after the input to produce the sum bits. E.g., the first (least-significant) sum bit is calculated by XOR ing the propagate in the farthest-right black box (a "1") with the carry-in (a "0"), producing a "1". The second bit is calculated by XOR ing the propagate in second box from the right (a "0") with C0 (a "0"), producing a "0". It takes more area to implement than the Brent-Kung adder, but has a lower fan-out at each stage, which increases performance. Wiring congestion is often a problem for Kogge-Stone adders as well.



2. Sparse Kogge-Stone Adder

Enhancements to the original implementation include increasing the radix and

sparsity of the adder. The radix of the adder refers to how many results from the previous level of computation are used to generate the next one. The original implementation uses radix-2, although it's possible to create radix-4 and higher. Doing so increases the power and delay of each stage, but reduces the number of required stages. While a complete adder would produce the output of all bits, this just outputs a series of carry bits at fixed intervals. These can be used as the carry-in bits for a series of smaller adders. This is useful in particular for FPGAs, where small ripple carry adders can be much faster than general-purpose logic thanks to fast connections between neighboring slices. This allows a large adder to be composed of many smaller adders by generating the intermediate carries quickly. The sparsity of the adder refers to how many carry bits are generated by the carry-tree. Generating every carry bit is called sparsity-1, where as generating every other is sparsity-2 and every fourth is sparsity-4. The resulting carries are then used as the carry-in inputs for much shorter ripple carry adders or some other adder design, which generates the final sum bits. Increasing sparsity reduces the total needed computation and can reduce the amount of routing congestion. Above is an example of a Kogge-Stone adder with sparsity-4 are shown in figure. It uses the combination of kogge-stone and ripple carry adder structures'.

The Sparse Kogge-Stone adder consists of several smaller ripple carry adders (RCAs) on its lower half and a carry tree on its upper half. Thus, the Sparse Kogge-Stone adder terminates with RCAs. The number of carries generated is less in a Sparse Kogge-Stone adder compared to the regular Kogge-Stone adder. The functionality of the GP block, black cell and the gray cell remains exactly the same as in the regular Kogge-Stone adder. The schematic for a 16-bit Sparse Kogge-Stone adder is shown in Figure. Sparse and regular Kogge-Stone adders have essentially the same delay when implemented on an FPGA although the former utilizes much less resources.

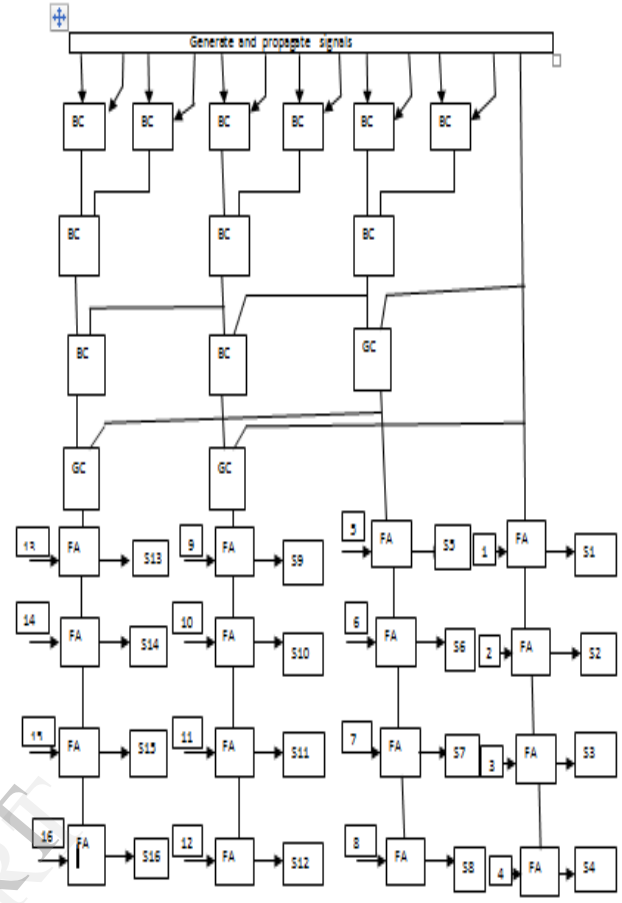
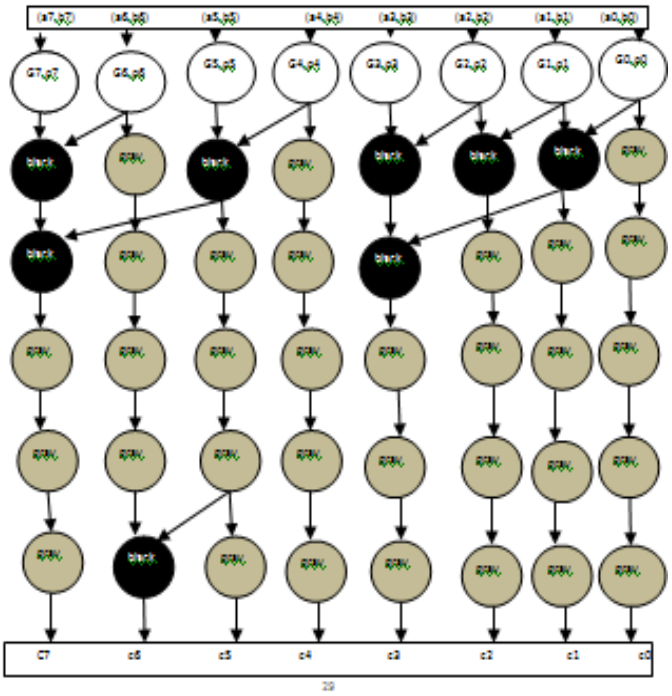


Fig. Sparse Kogge-Stone adder

Proposed Adder

The proposed adder is based on the combination of Brent-Kung adder and Kogge-Stone adder. The Kogge-Stone adder has maximum area and Brent-Kung adder has maximum delay. These two drawbacks are occurred in previously used adders. Proposed adder has to remove these drawbacks and it gives better area and delay results compared to kogge-stone adder. Flowchart for proposed adder is shown in figure.



The block diagram of proposed adder is shown in figure. In this diagram First taken the given inputs up to (a_0, a1…….a_{16}) and (b_0, b_1……..b_{16}). By using these inputs to calculate the generation and propagation signals for (g_0,p_0), (g_1,p_1)…….(g_{16},p_{16}). After the calculation of generation and propagation signals using Kogge-Stone ((KS) and Brent-Kung (BK) structures to perform Black cell and gray cell equations. In Kogge-stone structure each step 2^n gray cells are generated. Brent-Kung structure uses half of the gray cells are generated. To perform the generation of black cell and gray cell values to generate the carry values by using Kogge-Stone and Brent-Kung structures. After the calculation of all carry values these are combine with propagation signals with XOR operation to get sum(S) bits(S_0,S_1,S_2,S_3……S_{16})

RESULTS

The synthesis results for proposed adder. Proposed adder uses no of slices, no of LUTS and no of bounded IOBs. The results for ripple carry adder, carry-look ahead adder, carry-tree adders and our proposed adder.

Synthesis report for proposed adder

Selected Device: xa3s500eftg256-4
Number of Slices: 145 out of 4656 3%
Number of 4 input LUTs: 252 out of 9312 2%
Number of IOs: 443
Number of bonded IOBs: 443 out of 190 233%
Maximum combinational path delay: 12.178ns

Adder name	Area(slices)	Delay(ns)	No of logic levels
Ripple carry adder 16- bit	15	30.586	16
Carry-look ahead adder 16-bit	32	29..613	16
Sparse Kogge-stone adder 16-bit	45	14.85	10
Spanning tree adder 16-bit	42	13.55	10
Kogge-Stone adder 16-bit	80	10.72	8
Proposed adder 16-bit	60	9.45	7
Ripple carry adder 32-bit	35	42.69	33
Carry-look ahead adder 32-bit	70	41.52	33
Sparse Kogge-stone adder 32-bit	105	14.35	11
Spanning tree adder 32-bit	100	12.65	11
Kogge-Stone adder 32-bit	190	11.45	8
Proposed adder 32-bit	142	10.85	7

These are Synthesis results for proposed adder. This report tells about selected device, Number of slices, Number of Look up tables (LUTs) and Delay. Area is represented in Number of slices and Delay is measured in nanoseconds (ns).

CONCLUSION

In this paper we have seen that the area utilization and delay performance of the field programming gate array (FPGA) kit has been reduced when compared to the proposed adders.

REFERENCES

1. N.H.E.Weste and D.Harris, CMOS VLSI Design, 4thedition, Pearson–Addison-Wesley, 2011.
2. K. Vitoroulis and A. J. Al-Khalili, "Performance of Parallel Prefix Adders Implemented with Technology," IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007.
3. P. Ndai, S. Lu, D. Somesekhar, and K. Roy, "Fine-Grained Redundancy in Adders," Int. Symp. On Quality Electronic Design, pp. 317-321, March 2007.
4. M.Bečvář and P.Štukjunger, "Fixed-Point Arithmetic in FPGA," Acta Polytechnic, vol. 45, no. 2, pp. 67-72, 2005.
5. D. Gizopoulos, M. Psarakis, A. Paschalis, and Y.Zorian, "Easily Testable Cellular Carry Look ahead Adders," Journal of Electronic Testing: Theory and Applications 19, 285-298, 2003.
6. D. Harris, "A Taxonomy of Parallel Prefix Networks," in Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–2217, 2003.
7. S. Xing and W. W. H. Yu, "FPGA Adders: Performance Evaluation and Optimal Design," IEEE Design & Test of Computers, vol. 15, no. 1, pp. 24-29,Jan. 1998.
8. T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry Look ahead Adder," IEEE Trans. on Computers,vol. 41, no. 8, pp. 931-939, Aug. 1992.
9. R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp.260- 264, 1982.
10. P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. on Computers, Vol. C-22, No 8, August 1973.