ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design & Implementation of Efficient Adders

Ch Sravan Kumar
Pg Scholar(Vlsi&Es)
Dept Of Ece
Vidya Bharathi Institute Of Technology
Janagaon, Warangal, Telangana

B Pragathi(M Tech,(Ph.D))
Associate Professor
Dept Of Ece
Vidya Bharathi Institute Of Technology
Janagaon, Warangal, Telangana

B Kranthi Kumar
Associate Professor
Dept Of Ece
Vidya Bharathi Institute Of Technology
Janagaon, Warangal, Telangana

Abstract

In Very Large Scale Integration (VLSI) designs, Parallel prefix adders (PPA) have the better delay performance. This paper investigates four types of PPA's (Kogge Stone Adder (KSA), Spanning Tree Adder (STA), Brent Kung Adder (BKA) and Sparse Kogge Stone Adder (SKA)). Additionally Ripple Carry Adder (RCA), Carry Lookahead Adder (CLA) and Carry Skip Adder (CSA) are also investigated. These adders are implemented in verilog Hardware Description Language (HDL) using Xilinx Integrated Software Environment (ISE) 14.2 Design Suite. These designs are implemented in Xilinx Spartan 3 Field Programmable Gate Arrays (FPGA) and delays are measured,all these adder's delay, power and area are investigated and compared finally.

Key words —parallel prefix adders; carry tree adders; FPGA; logic analyzer; delay; power.

I. INTRODUCTION

The binary addition is the basic arithmetic operation in digital circuits and it became essential in most of the digital systems including Arithmetic and Logic Unit (ALU), microprocessors and Digital Signal Processing (DSP). At present, the research continues on increasing the adder's delay performance. In many practical applications like mobile and telecommunications, the Speed and power performance improved in FPGAs is better than microprocessor and DSP's based solutions. Additionally, power is also an important aspect in growing trend of mobile electronics, which makes large-scale use of DSP functions. Because of the Programmability, structure of configurable logic blocks (CLB) and programming interconnects in FPGAs, Parallel prefix adders have better performance.

The delays of the adders are discussed [1]. In this paper, above mentioned PPA's and RCA and CSA are implemented and characterized on a Xilinx vertex 5 FPGA. Finally, delay, power and area for the designed adders are presented and compared.

II. DRAWBACKS OF RIPPLE CARRY AND CARRY LOOKAHEAD ADDER

In figure1, the first sum bit should wait until input carry is given, the second sum bit should wait until previous carry is propagated and so on. Finally the output sum should wait until all previous carries are generated. So it results in delay.

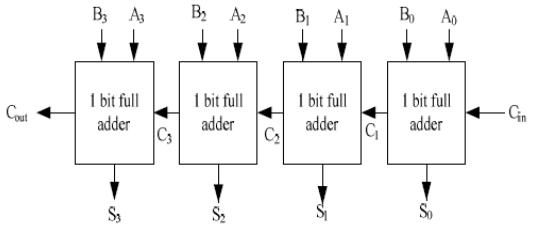


Fig. 1-4 bit ripple carry adder

In order to reduce the delay in RCA (or) to propagate the carry in advance, we go for carry look ahead adder .Basically this adder works on two operations called propagate and generate The propagate and generate equations are given by.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

$$P_i = A_i \oplus B_i \quad (1)$$
$$G_i = A_i.B_i \quad (2)$$

For 4 bit CLA, the propagated carry equations are given as

$$C_1 = G_0 + P_0 C_0 \quad (3)$$
$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0 \quad (4)$$
$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \quad (5)$$
$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0 \quad (6)$$

Equations (3),(4),(5) and (6) are observed that, the carry complexity increases by increasing the adder bit width. So designing higher bit CLA becomes complexity. In this way, for the higher bit of CLA's, the carry complexity increases by increasing the width of the adder. So results in bounded fan-in rather than unbounded fan-in, when designing wide width adders. In order to compute the carries in advance without delay and complexity, there is a concept called Parallel prefix approach.

III. DIFFERENCE BETWEEN PARALLEL-PRE FIX ADDERS AND OTHERS

The PPA's pre-computes generate and propagate signals are presented in [2]. Using the fundamental carry operator (fco), these computed signals are combined in [3].The fundamental carry operator is denoted by the symbol "o ",

$$(g_L, p_L) o (g_R, p_R) = (g_L + p_L.g_R, p_L, p_R)$$

For example, 4 bit CLA carry equation is given by

$$C_4 = (g_4, p_4) o [(g_3, p_3) o [(g_4, p_4) o (g_3, p_3)]] \quad (8)$$

For example, 4 bit PPA carry equation is given by

$$C_4 = [(g_4, p_4) o (g_3, p_3)] o [(g_4, p_4) o (g_3, p_3)] \quad (9)$$

Equations (8) and (9) are observed that, the carry look ahead adder takes 3 steps to generate the carry, but the bit PPA takes 2 steps to generate the carry.

IV. PARALLEL-PREFIX ADDER STRUCTURE

Parallel-prefix structures are found to be common in high performance adders because of the delay is logarithmically proportional to the adder width [2]. PPA's basically consists of 3 stages

- Pre computation
- Prefix stage
- Final computation

The Parallel-Prefix Structure is shown in figure 2.

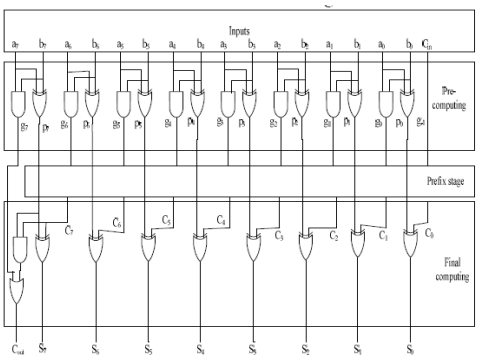


Fig. 2. Parallel-Prefix Structure with carry save notation

A. Pre computation

In pre computation stage, propagates and generates are computed for the given inputs using the given equations (1) and (2).

B. Prefix stage

In the prefix stage, group generate/propagate signals are computed at each bit using the given equations. The black cell (BC) generates the ordered pair in equation (7), the gray cell (GC) generates only left signal, following [2].

(BC) generates the ordered pair in equation (7), the gray cell (GC) generates only left signal, following [2].

$$G_{i:k} = G_{i:j} + P_{i:j}.G_{j-1:k} \quad (10)$$
$$P_{i:k} = P_{i:j}.P_{j-1:k} \quad (11)$$

More practically, the equations (10) and (11) can be expressed using a symbol "o "denoted by Brent and

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Kung. Its function is exactly the same as that of a black cell i.e.

$$G_{i:k} : P_{i:k} = (G_{i:j} : P_{i:j}) o (G_{j-1:k} : P_{j-1:k}) \qquad (12)$$

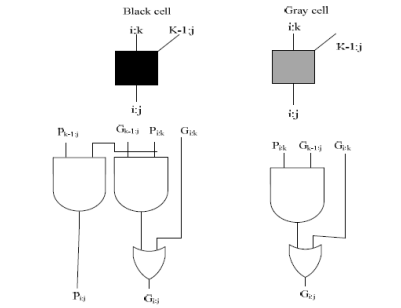


Fig. 3. Black and Gray Cell logic Definitions

The "o" operation will help make the rules of building prefix structures.

C. Final computation

In the final computation, the sum and carryout are the final output.

$$S_i = P_i \cdot G_{i-1:-1} \qquad (13)$$

$$C_{out} = G_{n:\ 1} \qquad (14)$$

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. Figure 3 shows the definitions of cells that are used in prefix structures, including BC and GC. For analysis of various parallel prefix structures, see [2], [3] &[4]. The 16 bit SKA uses black cells and gray cells as well as full adder blocks too. This adder computes the carries using the BC's and GC's and terminates with 4 bit RCA's. Totally it uses 16 full adders. The 16 bit SKA is shown in figure 4. In this adder, first the input bits (a, b) are converted as propagate and generate (p, g). Then propagate and generate terms are given to BC's and GC's. The carries are propagated in advance using these cells. Later these are given to full adder blocks. Another PPA is known as STA is also tested [6]. Like the SKA, this adder also terminates with a RCA. It also uses the BC's and GC's and full adder blocks like SKA's but the difference is the interconnection between them [7].The 16 bit STA is shown in the below figure 5.

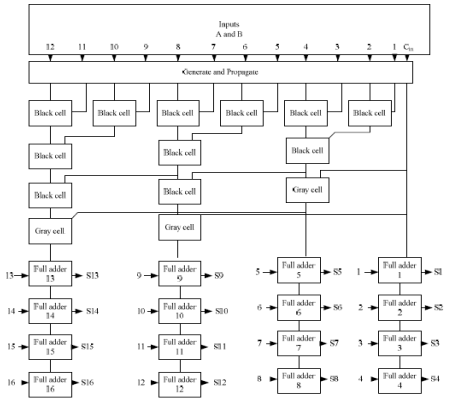


Fig. 4.16- bit sparse kogge-Stone adder

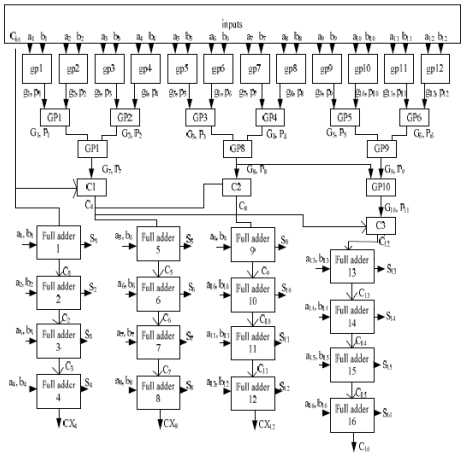


Fig. 5.16-bit spanning tree adder

KSA is another of prefix trees that use the fewest logic levels. A 16-bit KSA is shown in Figure 6. The 16 bit kogge stone adder uses BC's and GC's and it won't use full adders. The 16 bit KSA uses 36 BC's and 15 GC's. And this adder totally operates on generate and propagate blocks. So the delay is less when compared to the previous SKA and STA. The 16 bit KSA is shown in figure 6.In this KSA, there are no full adder blocks like SKA and STA [5] & [6]. Another carry tree known as BKA which also uses BC's and GC's but less than the KSA. So it takes less area to implement

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

than KSA. The 16 bit BKA uses 14 BC's and 11 GC's but kogge stone uses 36 BC's and 15 GC's. So BKA has less architecture and occupies less area than KSA. The 16 bit BKA is shown in the below figure 7.

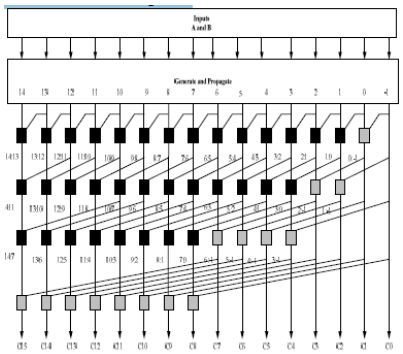


Fig. 6. 16-bit kogge stone adder

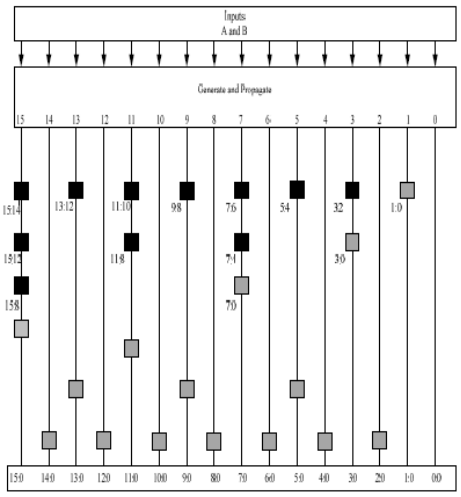


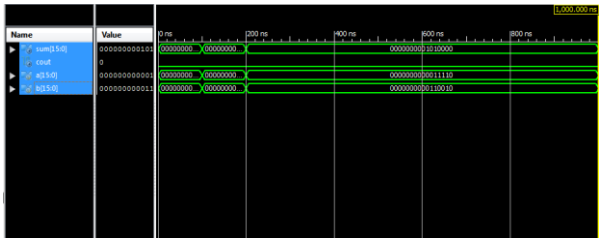
Fig. 7. 16 bit brent kung adder

BKA occupies less area than the other 3 adders called SKA, KSA, STA. This adder uses limited number of propagate and generate cells than the other 3 adders. It takes less area to implement than the KSA and has less wiring congestion. The

operation of the 16 bit brent kung adder is given below [3]. This adder uses less BC's and GC's than kogge stone adder and has the better delay performance which is observed in agilent 1692A logic analyzer. These adders are implemented in verilog HDL in Xilinx 13.2 ISE design suite and then verified using Xilinx virtex 5 FPGA through chip scope analyzer [7], [8] and [9]. And these were tested using Agilent 1692A logic analyzer. This allows to measure the adder delays directly. The Agilent 1692A logic analyzer is integrated to PC(Personal Computer) through Xilinx virtex 5 FPGA [10]. The test setup is depicted in the figure 10.

V. DISCUSSION OF RESULTS

The delays observed for adder designs from synthesis reports in Xilinx ISE 14.2 synthesis reports are shown in Figure11.



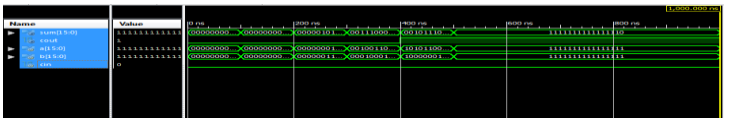
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	25	9112	0%
Number of fully used LUT-FF pairs	0	25	0%
Number of bonded IOBs	49	232	21%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 12.557ns
```

Fig.8. Sparse Tree Adder wave form ,area & delay

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	26	9112	0%
Number of fully used LUT-FF pairs	0	26	0%
Number of bonded IOBs	50	232	21%

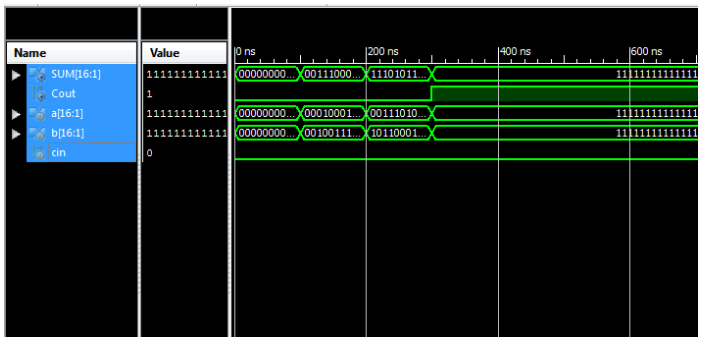


```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 12.317ns
```

Fig.9. Brent Kung Adder wave form,area & delay.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

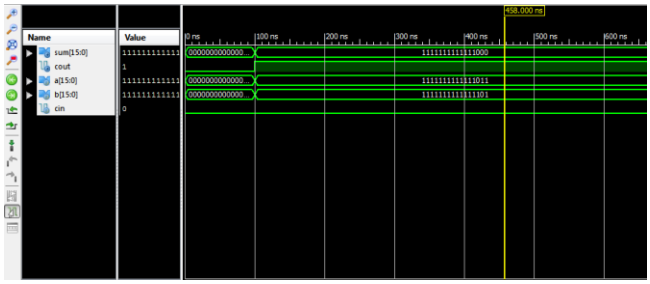


Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	28	9112	0%
Number of fully used LUT-FF pairs	0	28	0%
Number of bonded IOBs	50	232	21%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No pa
   Maximum output required time after clock: No p
   Maximum combinational path delay: 11.349ns
```

Fig.10.Spanning Tree Adder wave form, area & delay.

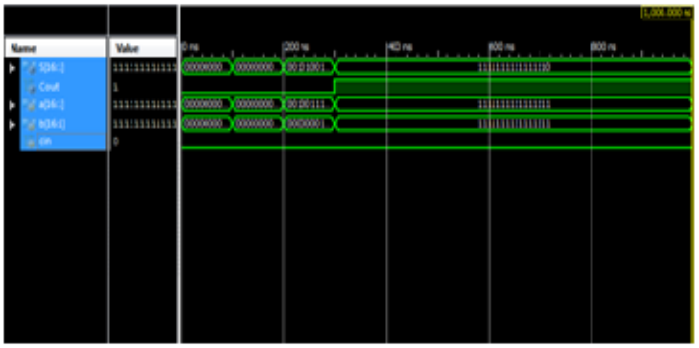


Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	34	9112	0%
Number of fully used LUT-FF pairs	0	34	0%
Number of bonded IOBs	50	232	21%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 11.255ns
```

Fig.11.Kogge Stone Adder wave form, area & delay



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	34	9112	0%
Number of fully used LUT-FF pairs	0	34	0%
Number of bonded IOBs	50	232	21%

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 10.578ns
```

Fig.12.Sparse-Kogge Stone Adder wave form, area & delay.

VI. CONCLUSION

From the study of analysis done on area and power, we have concluded that the efficiency is improved by 5.77 % in ours delay for RCA, The area of the adder designs is measured in terms of look up tables (LUT) and input output blocks (IOB) taken for Xilinx Spartan 6 FPGA is plotted in the figure. As per reference [1], ISE software doesn't give exact delay of the adders because it is not able to analyze the critical path over the adder [1]. From the figure 11, the CSA has more delay when compared to other adders. Out of all adders, RCA has less delay. SKA adder and BKA has about the same delay, where as KSA and STA has same delay. According to the synthesis reports, out of four parallel prefix adders, Sparse - KOGGE STONE adder has better delay.

REFERENCES

[1] David H.K.Hoe, Chris Martinez and Sri Jyothsna Vundavalli", Design and Characterization of Parallel Prefix Adders using FPGAs", 2011 IEEE 43rd Southeastern Symposium in pp. 168-172, 2011.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[2] N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson–Addison-Wesley, 2011.

[3] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp. 260-264, 1982.

[4] D. Harris, "A Taxonomy of Parallel Prefix Networks," in Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–7, 2003.

[5] 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.

[6] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, "Easily Testable Cellular Carry Lookahead Adders," Journal of Electronic Testing: Theory and Applications 19, 285-298, 2003.

[7] T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," IEEE Trans. on Computers, vol. 41, no. 8, pp. 931-939, Aug. 1992.

[8] Beaumont-Smith, A, Cheng-Chew Lim ,"Parallel prefix adder design", Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium,pp. 218 – 225,2001.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989

[9] K. Vitoroulis and A. J. Al-Khalili, "Performance of Parallel Prefix Adders Implemented with FPGA technology," IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007. 172.

[10] 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