Comparative Analysis of ALU Implementation with RCA and Sklansky Adders In ASIC Design Flow

Abdul Rehman Buzdar, Liguo Sun, Abdullah Buzdar
Department of Electronic Engineering and Information Science
University of Science and Technology of China (USTC)
Hefei, People's Republic of China

***Abstract*—An Arithmetic Logic Unit (ALU) is the heart of every central processing unit (CPU) which performs basic operations like addition, subtraction, multiplication, division and bitwise logic operations on binary numbers. This paper deals with implementation of a basic ALU unit using two different types of adder circuits, a ripple carry adder and a sklansky type adder. The ALU is designed using application specific integrated circuit (ASIC) platform where VHDL hardware description language and standard cells are used. The target process technology is 130nm CMOS from the foundry ST Microelectronics. The Cadence EDA tools are used for the ASIC implementation. A comparative analysis is provided for the two ALU circuits designed in terms of area, power and timing requirements.**

***Keywords*—Arithmetic Logic Unit; Ripple Carry Adder; Sklansky Adder; ASIC Design, EDA Tools**

I. Introduction

An Arithmetic Logic Unit (ALU) is key the element of a processor which performs arithmetic and logical operations on binary numbers [1-7]. In this work we have designed an ALU for a 32-bit processor using VHDL hardware description language. The ALU designed can perform four major tasks addition, subtraction, logical and shift operations. Fig. 1 shows the block diagram of the ALU designed, as can be seen that it is made edge-triggered by having flip-flops on its inputs and outputs. Two types of ALUs were designed and the difference in both the ALUs was of adder type. In the first one a ripple carry adder [8-17] was used while in the second one a pre-fix tree of sklansky type adder [18-25] was used. The adder is a very important component in digital systems, so lot of research has been done in past on various types of adders to improve the speed and area requirements [26-41]. Apart from adder both the ALUs consisted of a Logical and Shifter blocks.

In case of ALU-RCA two different designs were used to compare the performance of both the designs in terms of area and power usage. The first design contains a demux for selecting which block of ALU would be used based on the opcode and the blocks which are not used are shutdown to save power. While the second ALU-RCA design does not contain this demux and all the three units of ALU perform their respective operations and only one result out of three units is sent to the output based on the opcode which is selected by the mux which is placed before the output. In both ALUs designs a mux is placed before the output for selecting the result from only one of the three blocks based on the opcode.

The rest of the paper is organized as follow: In the next section we describe the design and verification of ALU circuits, followed by ALU circuit basic Synthesis. Later we describe the process of Design Respin, Power analysis, Place and Route of ALU circuits. Finally, we summarize our conclusions.

II. ALU Design- Verification

Initial verification of both the ALUs i.e. ALU-RCA and ALU-SKL were performed based on the waveform approach using ModelSim software tool [47] as we made both the ALUs generic so we reduced the ALU size to 8-bit just to make initial verification simple. The waveform based approach of verification is only useful during the initial phases of small designs and always requires more comprehensive verification in the later design stages. There were some small bugs found in the code which were later corrected.

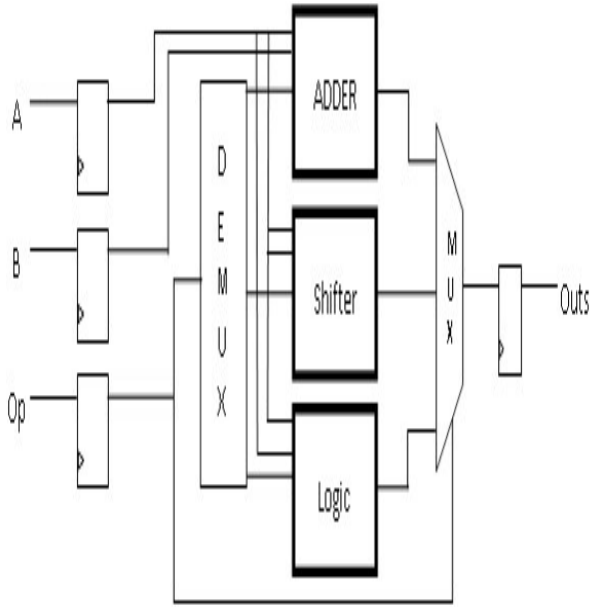

Figure 1: ALU Block Diagram

The next task was to define a VHDL TestBench for more comprehensive verification of ALU, than waveform approach. The TestBench written takes the required input values of A, B and opcode from stimuli file and compares the result to an expected output which is also taken from a stimuli file. If the result is incorrect against any of the testvectors then false is written in a dumped file else true is written. If the results are correct against all the test vectors then a message is displayed that simulation was successful else an error message

is displayed in the end of simulation. The first task in the verification was to functionally verify our 32-bit ALU meaning that whether our design conforms to specification and does this proposed design do what is intended. We can view the test vectors as a functional specification. The NCSIM logic simulator was used to logically verify our designs. The logic simulation is the use of some computer program to stimulate the operation of digital circuit before it is actually built. The designs were simulated against the 1000 randomized reference test vectors and found no errors. To check the correctness of our TestBench we changed some of the vectors in the testvectors stimuli files and again simulated our design this time it gave errors against those vectors which we had changed as expected. We cannot say that this verification is complete as we checked our design using only 1000 randomized testvectors and the possible combinations in our case is 268, but it is a good starting point of verification process and as they say verification is never complete.

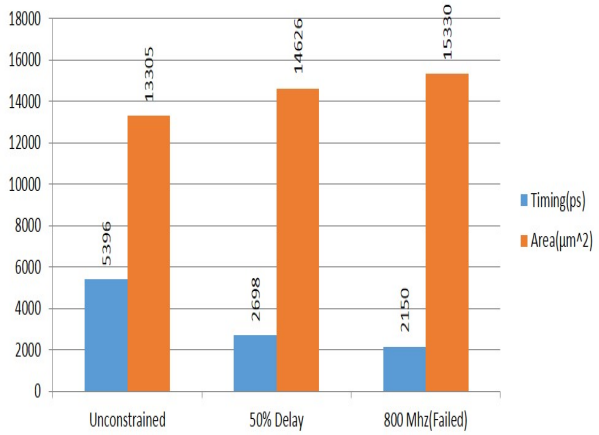

Figure 2: Timing and Area results of ALU-RCA

III. ALU Design- Basic Synthesis

The next task was to synthesize the VHDL code of ALUs using Cadence RTL compiler [42-45]. The VHDL descriptions of ALU will be mapped to certain process technology which in this case is 130-nm technology provided by STMicroelectronics [46]. After starting the RTL compiler the technology files were added by giving path and file name of library files. The next step in synthesis process was to read the VHDL files of ALU-RCA. The important thing to be noted here is that the TestBench was not included here since it is not synthesizable rather just behavioral description. The RTL compiler was instructed to assemble the VHDL descriptions of ALU-RCA into an internal representation i.e. network of logic gates by typing the elaboration command. The VHDL code of ALU-RCA was found synthesizable as it was kept in mind while writing it. The VHDL code of ALU-RCA can be called RTL since it is found to be synthesizable. Some time was spent in studying the gate level netlist produced during elaboration and later using GUI. Up till now we have done initial logic synthesis in which no process technology is used rather the RTL compiler makes use of a virtual gate library. The next step was to assign our hardware descriptions to real standard cells, which is commonly known as technology mapping. The initial synthesis was done without any timing constraint and using low effort to study the intrinsic behavior of implementation. The reason for using low effort here is that we do not want to optimize the timing at all to study the actual behavior of our design which is also known as Static Timing Analysis (STA). The timing and area of our design was documented by giving the appropriate commands. The worst-case delay value and estimated area of the implementation was found to be $5396ps$ and $13305\mu m^2$ respectively.

The worst-case signal propagation path of ALU-RCA was found to be between input and output registers through the chain of adders starting from index 0 to 31, this was observed by looking at the GUI window. The design was re-synthesized using the new timing goal of 50% of the delay we obtained in the last task which comes to be $2689ps$ using medium effort. Here we are using medium effort because we want the Compiler to put some more effort to meet this timing constraint. The worst-case delay value and estimated area of the implementation was found to be $2698ps$ and $14626\mu m^2$ respectively.

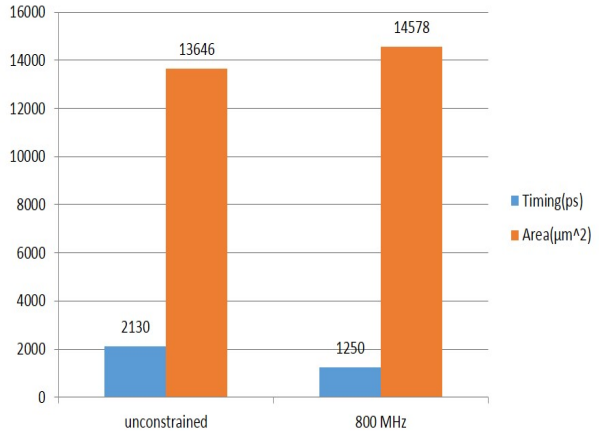

Figure 3: Timing and Area results of ALU-SKL

The RTL Compiler was able to meet this timing constraint at the expense of increased area because it has to put more effort to meet this timing constraint. The worst-case timing path was still passing through the chain of adders. The ten longest signal propagation paths were documented and they all were passing through the chain of full adders. The data books were studied to see what standard cells are being used. In the last task when we synthesized the design without timing constraint it was using one bit fuller adder cells but with timing constraint of $2698ps$ it is now using half adders those having less area than full adders. But as the half adders are used in more numbers than the full adders, so the total area of design has increased.

Later it was found out that the original intension of our ALU design is actually to put in inside a 800-MHz processor. So the ALU-RCA was re-synthesized using $1250ps$ timing constraint and with medium effort. The ALU-RCA was unable to meet this timing constraint as there was negative slack time, so we cannot use ALU-RCA for 800-MHz processor. The worst-case path was still passing through the chain of adders.

This time the standard cells used in implementation were different from the previous task as Compiler tried its best to meet the timing constraint using fast standard cells. The worst-case delay value and estimated area of the implementation was found to be $2150ps$ and $15330\mu m^2$ respectively.

The compiler was only able to reach the timing of $2150ps$ with increased area as it has to put extra effort in trying to meet this stricter timing constraint but was unsuccessful. The next step in the ASIC design flow is the verification of synthesized netlist of ALU-RCA. The VHDL description of ALU-RCA was synthesized again with new timing constraint of $2698ps$ for which function is guaranteed using medium effort. The TestBench was used with test vectors for the verification of synthesized netlist. The clk period used in the TestBench was 50% of timing constraint i.e. $1349ps$ meaning that clock used in the TestBench would be high for $1349ps$ and low for $1349ps$. The netlist was successfully verified without any errors, which proves the idea that netlist has the same functionality as that of VHDL description of design. Fig. 2 shows the timing and area results of ALU-RCA at different settings.

IV. ALU Design- Design Respin and Power analysis

The ALU-SKL was synthesized without timing constraint and using low effort to study the intrinsic implementation. The worst-case delay value and estimated area of the implementation was found to be $2130ps$ and $13646\mu m^2$ respectively. The ALU-SKL was re-synthesized using stricter timing constraint corresponding to 800-MHz using medium effort. The worst-case delay value and estimated area of the implementation was found to be $1250ps$ and $14578\mu m^2$ respectively. The ALU-SKL was able to meet this timing constraint at the expense of increased area which means that we can use ALU-SKL for 800-MHz processor. The worst-case timing path of ALU-SKL was found to be passing through shifter block. The main reason of using Sklansky adder is that it is faster than RCA based adder. The worst-case path also proves this as in the case of ALU-RCA the worst case was passing through the chain of adders and now it is through the shifter block because of high performance of Sklansky adder in terms of speed. The data books were studied again to see which standard cells Sklansky adder is using and it was found out that it was using completely different cells than RCA e.g. 4 Input NOR gate etc and that's why it has more speed at the expense of larger area. The ten longest paths were documented and it found that first nine paths were passing through the shifter while the tenth path was passing through the Sklansky adder. Fig. 3 shows the timing and area results of ALU-SKL at different settings.

The Fig. 4 and 5 shows that the area of ALU-RCA changes more rapidly than ALU-SKL as the ALU-RCA has to put more effort to meet the stricter timing constraint and its area increases, while ALU-SKL is fast adder easily meets the stricter timing constraint without increasing the area. The netlist of ALU-SKL was also successfully verified.

The next task was to perform power analysis of both designs for a timing constraint that both ALUs satisfy. So Both the ALUs were synthesized with the timing constraint of $2500ps$ using medium effort. The estimated area of the implementation was found to be $14828\mu m^2$ for ALU-RCA and $13825\mu m^2$ for ALU-SKL. The ALU-RCA with DeMux was also synthesized using this timing constraint to compare its area and timing with the ALU-RCA without the DeMux.

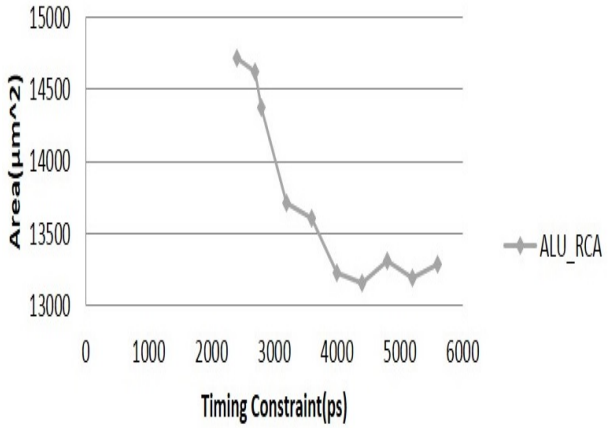

Figure 4: Scaling of Area of ALU-RCA with Timing Constraint

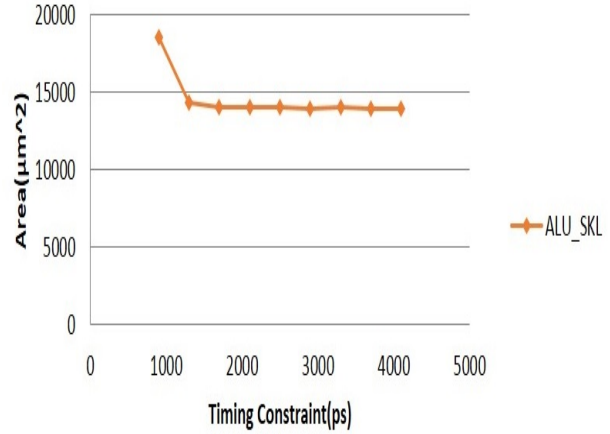

Figure 5: Scaling of Area of ALU-SKL with Timing Constraint

The initial power analysis was performed by assigning some switching probabilities on the primary inputs. This is common practice as initially the the test vectors are not available but the correctness of the power analysis highly depends on the test vectors used for power analysis. The Table I and II clearly show that increasing the toggling probability increases the switching power where as the leakage power almost remains the same. Table III shows the clk power and Capacitance of ALU-RCA and ALU-SKL. The ALU-RCA is consuming more power and has more area compared to ALU-SKL for stricter timing constraint of $2500ps$. This is because ALU-RCA has to put more effort to meet this timing constraint which result in more area and high power, while ALU-SKL has no problem in meeting this timing constraint which result in more area and power efficient design. The individual power for the three blocks was also compared and it was found out that adder was consuming the most power then logical block and shifter was consuming the least power. The power

dissipated in the clock net for both the ALU-RCA and ALU-SKL was documented and was found to be in agreement with the common expression $f * V_{DD}^2 * C$, it should be noted here that the RTL compiler shows wrong unit for capacitance.

Table I: Power results of ALU-RCA and ALU-SKL

	Toggling Probability(ns)	Dynamic Power(nW)	Leakage Power(nW)
ALU-RCA	0.02	2005282	377251
ALU-SKL	0.02	1806029	325786

Table II: Power results of ALU-RCA and ALU-SKL

	Toggling Probability(ns)	Dynamic Power(nW)	Leakage Power(nW)
ALU-RCA	0.1	5039712	377858
ALU-SKL	0.1	4443770	326528

Table III: Synthesis results

	CLK Power(nW)	Capacitance(fF)
ALU-RCA	219974	381
ALU-SKL	178560	310

The next task was to relax the timing constraint for both the ALUs to see the impact of this on area and power. The timing constraint was set to $4500ps$ and both the designs were re-synthesized using medium effort. The estimated area of the implementation was found to be $13468\mu m^2$ for ALU-RCA and $13819\mu m^2$ for ALU-SKL. Now it can be seen that the ALU-RCA using less area and power as compared to ALU-SKL. It means that it is better to use ALU-RCA if the timing constraint is not high and we require area and power efficient ALU. Table IV and V shows the power results for timing constraint of $4500ps$.

Table IV: Power results of ALU-RCA and ALU-SKL

	Toggling Probability(ns)	Dynamic Power(nW)	Leakage Power(nW)
ALU-RCA	0.02	1315083	318043
ALU-SKL	0.02	1332294	325224

Table V: Power results of ALU-RCA and ALU-SKL

	Toggling Probability(ns)	Dynamic Power(nW)	Leakage Power(nW)
ALU-RCA	0.1	3877682	317800
ALU-SKL	0.1	3974884	325085

Table VI shows the power and area results for two different ALU-RCA designs. The design with DeMux is more power efficient as the switching is only taking place in the block that is needed at that time depending on the opcode but has little area overhead compared to the other design which is without DeMux. The power analysis was performed using toggling probability of 0.1 ns on the primary inputs.

The next task of power analysis was to do power analysis of ALU-RCA using three different set of test vectors, as discussed earlier that the correctness of power analysis highly depends on the test vectors used for power analysis. The TestBench

Table VI: Synthesis results of ALU-RCA

ALU-RCA Type	Dynamic Power(nW)	Area(μm^2)
With DeMux	4887000	14931
Without DeMux	5039712	14828

was used to generate the VCD file for each set of test vectors using the synthesized netlist generated from the design with correct timing constraint, here timing constraint was set to $2500ps$ with medium effort. One important thing to note is that the clock period used in the TestBench should be 50% of timing constraint. Table VII shows the power results using test vectors. The result of power analysis using test vectors shows that the toggling probability of Random test vectors is higher as compared to Regular and Real trace because random test vectors has highest switching power where as the toggling probability of regular and real trace seems to be almost same and seems to be close to 0.1 ns. As instructed the TCF files were checked to compare the signals A[16] and B[15] in all the three TCF files. It was found that in the regular trace test vectors the signal A[16] was changing state from 0 to 1 and vice versa all the time and has the high state toggling probability of almost 0.5 ns, where as the signal B[15] was all the time zero and has the high state toggling probability of almost 0.0 ns.

Table VII: Power results of ALU-RCA

	Random	Regular	Real Trace
Dynamic Power(nW)	9524859	4578158	4509525
Leakage Power(nW)	378506	379258	387047

V. ALU Design- Place and Route

The final task in the ASIC design flow is place and route step which takes considerable amount of experience to make good place-and-route. The first step was to generate netlist file of our own design using timing constraint of 3.2 ns also need to produce the constraint file as we decided to work with our ALU-RCA design. Then these files were placed in the proper directories as directed. The partitioning step was performed followed by the Floorplanning, we placed the input registers on the left and the output registers on the right side of core. Then pin placement and power routing was done. The standard cell placement was the next step, it was found out that the placement of cells was done according to our pre-placement constraint. Then Clock Tree Synthesis (CTS) step was performed, after Pre-CTS optimization timing was checked and the timing constraint was not met. The Pre-CTS step can be explained as mapping the design to logic gates, without mapping to actual cells i.e. buffers. The actual CTS step was performed which is like mapping the design to actual cells. The positions of clock buffers and clock tree were checked and it was found that our design has one level of buffers. The timing was checked again after this step and we were still unable to meet the constraint. The last step of CTS i.e. post-CTS optimization was performed to do the optimization based on existing clock tree. The timing was checked again and it was found that timing was improved a lot and the slack time was -0.466, much better than before when the slack time was -2.259. The routing and post-route

optimization was performed and the clock and reset signals should have highest priority because these signals have to be provided to every block in the design and therefore are critical. The Filler cells were used to fill the gaps and to connect them to the power rails. The layout verification was done, four MinCut violations were found which were later removed using the fixMinCutVia command. The final timing analysis was performed and the slack time was found to be -0.395.

VI. Conclusion

The aim of this research was to design a Arithmetic Logic Unit (ALU) for a 32-bit processor using two different adder circuits. The two ALU units were implemented in VHDL using Ripple Carry Adder and Sklansky Adder circuits. After the VHDL implementation synthesis was performed using Cadence RTL compiler to compare the performance of both the ALU units in terms of area, power and timing requirements. The VHDL descriptions of ALU were mapped to 130-nm process technology provided by STMicroelectronics. The synthesis results shows that the area of ALU-RCA changes more rapidly than ALU-SKL as the ALU-RCA has to put more effort to meet the stricter timing constraint at the expense of more area. While ALU-SKL which is a fast adder easily meets the stricter timing constraint without increasing the area and power consumption. It was also observed that the ALU-RCA uses less area and power as compared to ALU-SKL, so it is better to use ALU-RCA if the timing constraint was not high so in this way we can get more area and power efficient ALU Design.

Acknowledgment

This work is partially supported by the Chinese Academic of Sciences and The World Academy of Sciences CAS-TWAS President's Fellowship 2013-2017.

References

[1] Ravindran, N.; Lourde, R. Mary "An optimum VLSI design of a 16-BIT ALU", Information and Communication Technology Research (ICTRC), 2015 International Conference on, On page(s): 52-55

[2] Larsson-Edefors, P.; Jeppson, K. "Timing- and power-driven ALU design training using spreadsheet-based arithmetic exploration", Microelectronics Education (EWME), 10th European Workshop on, On page(s): 151 154

[3] Kaur, H.; Singh, H. "Advanced ALU with inbuilt selection modules for Genetic Algorithm processor", Signal Processing, Computing and Control (ISPCC), 2015 International Conference on, On page(s): 405 - 410

[4] Kaur, Harmeet; Kaur, Manpreet; Singh, Haramardeep "Genetic Algorithm processor with advanced ALU, memories and control unit", India Conference (INDICON), 2015 Annual IEEE, On page(s): 1 - 5

[5] H. Singh, "Design of enhanced arithmetic logical unit for hardware genetic processor", Advances in Computing, Communications and Informatics (ICACCI), 2014 International Conference on, pp. 549-553

[6] M. Suzuki, N. Ohkubo, T. Yamanaka, A. Shimizu, and K. Sasaki, "A 1.5 ns 32 b CMOS ALU in double pass-transistor logic", IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, vol. 267, pp. 90-91, 1993

[7] Parameswar, A.; Hara, H.; Sakurai, T. "A swing restored pass-transistor logic-based multiply and accumulate circuit for multimedia applications", Solid-State Circuits, IEEE Journal of, On page(s): 804 - 809 Volume: 31, Issue: 6, Jun 1996

[8] T. -Y. Chang and M.-J. Hsiao, "Carry-select adder using single ripple-carry adder", Electronics Letters, vol. 34, no. 22, pp. 2101-2103, 1998

[9] Chung-Hsun Huang; Jinn-Shyan Wang; Chingwei Yeh; Chih-Jen Fang "The CMOS carry-forward adders", Solid-State Circuits, IEEE Journal of, On page(s): 327 - 336 Volume: 39, Issue: 2, Feb. 2004

[10] Senejani, M. Nadi; Ghadiry, M. Hossein "Low dynamic power high performance adder", CAD Systems in Microelectronics, 2009. CADSM 2009. 10th International Conference - The Experience of Designing and Application of, On page(s): 242 - 245

[11] Senejani, M. Nadi; Hosseinghadiry, M.; Miryahyaei, M. "Low Dynamic Power High Performance Adder", Future Computer and Communication, 2009. ICFCC 2009. International Conference on, On page(s): 482 - 486

[12] C.-J. Fang, C.-H. Huang, J.-S. Wang, and C.-W. Yeh, "Fast and compact dynamic ripple carry adder design ", Proc. 3rd IEEE Asia-Pacific Conf. ASIC, pp. 25-28, 2002

[13] G. A. Ruiz, "Evaluation of three 32-bit CMOS adders in DCVS logic for self-timed circuits", IEEE J. Solid-State Circuits, vol. 33, pp. 604-613, 1998

[14] W. Hwang, G. Gristede, P. Sanda, S. Y. Wang, and D. F. Heidel, "Implementation of a self-resetting CMOS 64-bit parallel adder with enhanced testability", IEEE J. Solid-State Circuits, vol. 34, pp. 1108-1117, 1999

[15] A. Rothermel, "Realization of Transmission-Gate Conditional-Sum (TGCS) Adders with Low Latency Time", IEEE J. Solid-State Circuits, vol. 24, pp. 558-561, 1989

[16] N. Burgess, "Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI", 20th IEEE Symposium on Computer Arithmetic, pp. 103-111

[17] Burgess, Neil "Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI", Computer Arithmetic (ARITH), 2011 20th IEEE Symposium on, On page(s): 103 111

[18] M.Moghaddam and M. B. Ghaznavi-Ghoushchi ,"A New Low-Power, Low-area, Parallel Prefix Sklansky Adder with Reduced Inter-Stage Connections Complexity",IEEE Computer society,2011

[19] V. S. Veeravalli and A. Steininger, "Architecture for monitoring set propagation in 16-bit sklansky adder," in Quality Electronic Design (ISQED), 2014 15th International Symposium on, March 2014, pp. 412-419.

[20] N. Burgess, "New Models of Prefix Adder Topologies", J. VLSI Signal Processing Systems, Vol. 40, (June 2004), pp. 125 - 141.

[21] Roy, S.; Choudhury, M.; Puri, R.; Pan, D.Z. "Towards Optimal Performance-Area Trade-Off in Adders by Synthesis of Parallel Prefix Structures", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, On page(s): 1517 - 1530 Volume: 33, Issue: 10, Oct. 2014

[22] Choi, Y.; Swartzlander, E. E. "Speculative Carry Generation With Prefix Adder", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, On page(s): 321 - 326 Volume: 16, Issue: 3, March 2008

[23] Choi, Y.; Swartzlander, E.E., Jr. "Parallel prefix adder design with matrix representation", Computer Arithmetic, 2005. ARITH-17 2005. 17th IEEE Symposium on, On page(s): 90 98

[24] Liu, Jianhua; Zhu, Yi; Zhu, Haikun; Cheng, Chung-Kuan; Lillis, John "Optimum Prefix Adders in a Comprehensive Area, Timing and Power Design Space", Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific, On page(s): 609 615

[25] Chen, Jun; Stine, James E. "Enhancing parallel-prefix structures using carry-save notation", Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on, On page(s): 354 357

[26] Yingtao Jiang, Abdulkarim Al-Sheraidah, Yuke Wang, Edwin Sha and Jin-Gyun Chung, "A novel Multiplexer-based low power Full Adder", IEEE Transactions on Circuits and systems-II, vol. 51, 2004

[27] R. Zlatanovici, S. Kao and B. Nikolic, "Energy-delay optimization of 64-bit carry look-ahead adders with a 240 ps 90 nm CMOS design example", JSSC, vol. 44, no. 2, pp. 569-583, 2009

[28] H. Ling, "High-Speed Binary Adder", IBM J. Research and Dev., vol 25, p.156-166 (May 1981) J. Grad and J.E. Stine, "New algorithms for carry propagation", Proc. 15th ACM Great Lakes symposium on VLSI, Chicago, 2005, pp. 396 - 399

[29] Harris, D.; Sutherland, I. "Logical effort of carry propagate adders", Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on, On page(s): 873 - 878 Vol.1 Volume: 1, 9-12 Nov. 2003

[30] Patel, R. A.; Benaissa, M.; Powell, N.; Boussakta, S. "Novel Power-Delay-Area-Efficient Approach to Generic Modular Addition", Circuits and Systems I: Regular Papers, IEEE Transactions on, On page(s): 1279 - 1292 Volume: 54, Issue: 6, June 2007

[31] Roy, S.; Choudhury, M.; Puri, R.; Pan, D.Z. "Polynomial Time Algorithm for Area and Power Efficient Adder Synthesis in High-Performance Designs", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, On page(s): 820 - 831 Volume: 35, Issue: 5, May 2016

[32] Ge Yang; Seong-Ook Jung; Kwang-Hyun Baek; Soo Hwan Kim; Suki Kim; Sung-Mo Kang "A 32-bit carry lookahead adder using dual-path all-N logic", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, On page(s): 992 - 996 Volume: 13, Issue: 8, Aug. 2005

[33] Esposito, D.; De Caro, D.; Napoli, E.; Petra, N.; Strollo, A. G. M. "Variable Latency Speculative Han-Carlson Adder", Circuits and Systems I: Regular Papers, IEEE Transactions on, On page(s): 1353 - 1361 Volume: 62, Issue: 5, May 2015

[34] Zlatanovici, R.; Kao, S.; Nikolic, B. "EnergyDelay Optimization of 64-Bit Carry-Lookahead Adders With a 240 ps 90 nm CMOS Design Example", Solid-State Circuits, IEEE Journal of, On page(s): 569 - 583 Volume: 44, Issue: 2, Feb. 2009

[35] Zamanlooy, Babak; Novak, Ashley; Mirhassani, Mitra "Complexity Study of the Continuous Valued Number System Adders", Multiple-Valued Logic (ISMVL), 2012 42nd IEEE International Symposium on, On page(s): 116 121

[36] Yuke Wang; Parhi, K.K. "A unified adder design", Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on, On page(s): 177 - 182 vol.1 Volume: 1, 4-7 Nov. 2001

[37] Kwak, Sanghoon; Har, Dongsoo; Lee, Jeong-Gun; Lee, Jeong-A "Design of Heterogeneous Adders Based on Power-Delay Tradeoffs", Embedded Computing, 2008. SEC '08. Fifth IEEE International Symposium on, On page(s): 223 226

[38] Roy, Subhendu; Choudhury, Mihir; Puri, Ruchir; Pan, David Z. "Polynomial time algorithm for area and power efficient adder synthesis in high-performance designs", Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific, On page(s): 249 254

[39] Baliga, Akansha; Yagain, Deepa "Design of High Speed Adders Using CMOS and Transmission Gates in Submicron Technology: A Comparative Study", Emerging Trends in Engineering and Technology (ICETET), 2011 4th International Conference on, On page(s): 284 289

[40] Vergos, H.T.; Efstathiou, C.; Nikolos, D. "Diminished-one modulo 2n+1 adder design", Computers, IEEE Transactions on, On page(s): 1389 - 1399 Volume: 51, Issue: 12, Dec 2002

[41] Sun Xu-guang; Mao Zhi-gang; Lai Feng-chang "A 64 bit parallel CMOS adder for high performance processors", ASIC, 2002. Proceedings. 2002 IEEE Asia-Pacific Conference on, On page(s): 205 208

[42] Cadence EDA Tools. [Online]. Available: http://www.cadence.com

[43] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, "Distance and Speed Measurements using FPGA and ASIC on a high data rate system" International Journal of Advanced Computer Science and Applications(IJACSA), 6(10), 2015, 273-282.

[44] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, "Instruction Decompressor Design for a VLIW Processor", Informacije MIDEM-Journal of Microelectronics, Electronic Components and Materials Vol. 45, No. 4 (2015), 225-236.

[45] Abdul Rehman Buzdar, Azhar Latif, Liguo Sun and Abdullah Buzdar, "FPGA Prototype Implementation of Digital Hearing Aid from Software to Complete Hardware Design" International Journal of Advanced Computer Science and Applications(IJACSA), 7(1), 2016, 649-658.

[46] STMicroelectronics. [Online]. Available: http://www.st.com

[47] ModelSim. [Online]. Available: https://www.mentor.com