International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Design of High Speed Rounding Technique Based Approximate Multiplier

*[1]M Naveen Kumar Reddy, [2]E Balakrishna**

[1]M Tech, Student of ECE, CRIT College Anantapur, Andhra Pradesh, India
[2]Assistant Professor, ECE, CRIT College, Anantapur, Andhra Pradesh, India

A B S T R A C T

Approximate computing is one of best suited efficient data processing for error resilient applications, such as signal and image processing, computer vision, machine learning, data mining etc. Approximate computing reduces accuracy which is acceptable as a cost of increasing the circuit characteristics depends on the application. Desirable accuracy is the threshold point for controlling the trade off, between accuracy and circuit characteristics under the control of the circuit designer. In this work, the rounding technique is introduced as an efficient method for controlling this trade off. In this regard multiplier circuits as a critical building block for computing in most of the processors have been considered for the evaluation of the rounding technique efficiency. The impact of the rounding method is investigated by comparison of circuit characteristics for multipliers. In addition, to improve efficacy ofthe proposed approximate multiplier is done by using parallel prefix adder.

Project will be developed using verilog HDL. Xilinx ISE tool is used to perform the Simulation and Synthesis.

Keywords: HDL, XILINK, PREFIX ADDER

INTRODUCTION

Decreased alertness is among the most critical design criteria in virtually any electronic system, particularly the mobile ones including mobile phones, tablets and various gadgets. This minimum output (speed) penalties is required to be obtained extremely. Digital Signal Processing Blocks (DSP) are key components for the production of various communication systems of these mobile devices. The arithmetic and logic operations is the centre of the computation method, and supplies the greatest proportion of multipliers for all the operations carried out on such DSP systems. The improvement of multiplier speed and power / energy-efficient features plays an important role in enhancing processor efficiency.

Many DSP core devices incorporate audio / video algorithms with final outputs made for animal feed. pictures or video clips. This helps one to use estimations for speed energy consumption change. This is attributed to human beings' poor visual capacity to see an picture or a recording. Many fields where the accuracy of the mathematical functions does not significantly impact the efficiency of the device are included in contrast to the based on image processing framework. The model is capable of creating choices between precision, speed and energy usage through being willing to use estimated computation.

Used to approximate the arithmetic units, the loop, logic, but instead architectural design levels as well as algorithms and software layers may take different levels of design Abstraction. Estimation may be carried out using a range of methods such as the authorizing of such timing abuses (such as increasing voltage or overclocking) and feature approximations (e. g. alteration of a part of the circuit boolean method) or variations. A variety of approximating math basic components, including adders or multipliers, have been introduced in the field of feature approximations on various architecture stages. We are working on a high-speed low energy multiplier for error adaptable DSP applications. The suggested, even area-efficient estimated multiplier is built by amending the standard algorithm multiplication method to assume round input values

MULTIPLER

2.1 Introduction to Multiplier

Most high-performance devices, like FIR filters, microprocessors, optical signal processors, etc. multipliers shape main components. A complex method is a written arithmetic. The number must not be taken into account with unauthenticated multiplying. The same method can not be used in signed multiplication, since the sign number is in a 2 shape, which, if similiar with non- signed multiplication, will yield an inaccurate conclusion

2.9 Approximate Multiplier Importance

Some of previous works on directly proportional to the concentration are described briefly. Proposed in an approximate multiplier and approximate adder based on a technology known broken-array multiplier (BAM). By implementing the BAM approximation approach to the standard adjusted booth multiplier, the estimated signed booth multiplier produced energy consumption savings of 28% to 58.6% and region reductions of 19.7% to 41.8% for various term lengths relative to a traditional booth multiplier indicated an estimated multiplier.

An roughly signed 32-bit multiplier for prediction purposes in pipeline processors was optimised to be 20% quicker than a full-adder-based leaf multiplier, with a likelihood of error of about 14%. In an error-tolerant multiplier, which determined the approximation result by splitting the multiplier into one accurate and one estimated component, the accuracies for varying data widths were recorded. For a 12-bit multiplier, power savings of more than 50% were reported. Designed and evaluated two estimated 4:2 compressors for use in a standard Dadda multiplier.

The usage of estimated multipliers for computer vision applications, leading to lower power consumption, latency, and transistor compared to those with an exact multiplier nature, was discussed in the literature. Multiplier architecture (ACMA) has been suggested for mistake-resilient systems. To increase its output, the ACMA used a technique called carry-in prediction that worked on a pre-computation logic. Relative to the actual version, the suggested estimated calculation resulted in approximately 50 % reduction in latency by rising the crucial path. Estimated Wallace tree multiplier (AWTM). Once, to the the crucial path, it used carry-in estimation. Throughout this study, AWTM has been used in a real-time comparison picture test showing around 40 percent and 30 percent power and region declines, respectively, with no image quality degradation relative to using a direct Wallace tree multiplier (WTM) framework.

Proposed in roughly unsigned multiplication dependent on conditional operand logarithm. In the suggested multiplication, summing the estimated logarithms decides the operation product. Multiplier is then generalised to a change, introducing operations. Proposed in a process to increase the precision of multiplication strategy. It was focused on input operand decomposition. This approach considerably increased the average failure at the price of rising estimated multiplier hardware almost twice.

Presented in a Dynamic Segment Method (DSM), which performs multiplication on an m-bit segment starting from the top one bit of input operands. A dynamic range unbiased multiplier (DRUM) multiplier was introduced, which selects an m-bit section beginning from the leading one bit of input operands and sets the least important bit of truncated values to one. In this system, shortened values are multiplied and left to produce the final output. Approximately 4 WTM has also been suggested using an incorrect 4:2 measure. Additionally, an error detection device was proposed to fix outputs. This unreliable Wallace multiplier may be used in an array layout to create larger multipliers.

Almost all of the previously suggested estimated multipliers was focused on either changing a particular precise multiplier 's structure or rising difficulty. Similarly, we suggest conducting the estimated multiplication by simplifying the operation. The distinction between our research and is that while the concepts of both works are now almost identical for unsigned numbers, our suggested approach's mean error is smaller. We also propose certain approximation methods when calculating with ’s square.

EXISTING SYSTEM

4.1 Introduction of Wallace Multiplier

The computer scientist Luigi WALLACE invented the multiplier of the WALLACE hardware during 1965. The multiplier WALLACE is derived as a multiplier parallel[5]. It's marginally quicker and needs fewer steps. Parallel multipliers are used with various forms of schemes. The WALLACE method for the summation of input signals is one of the simultaneous multiplication schemes that effectively minimises the amount of adder stage needed.

To do this, total and half adders are used to reduce the amount of line numbers in the matrix of bits at increasing point of summation. While a normal, less complex structure exists in the WALLACE multiplication, the mechanism is slower due to the serial multiplier. The multiplier WALLACE is even less expensive relative to the multiplier for wallace vine. The report also develops and analyses the WALLACE multipliers using full adders of various reasoning types. The essay discusses specific approaches.

4.2. Implementation of Wallace Multiplier

The WALLACE multiplier algorithm is based on the shape below. Throughout the first stage

AND stages form the partial product matrix.

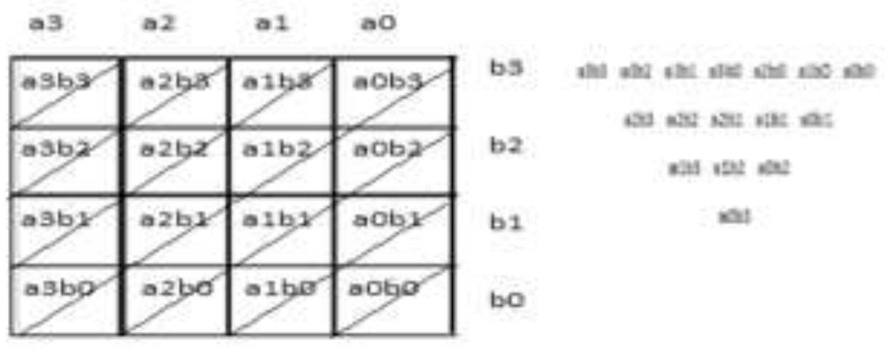


Fig 4.14x4 WALLACE Algorithm

Steps involved in WALLACE TREE multipliers Algorithm

Multiply each bit (that's-AND) of one argument, generating N outcomes, by each bit of the other. The wires bear various weights based on the location of the weighted bits.

Reduce to two complete layers of adders the partial products.

Group the wires and connect them to a traditional adder in two digits. A set of AND gates created product names.

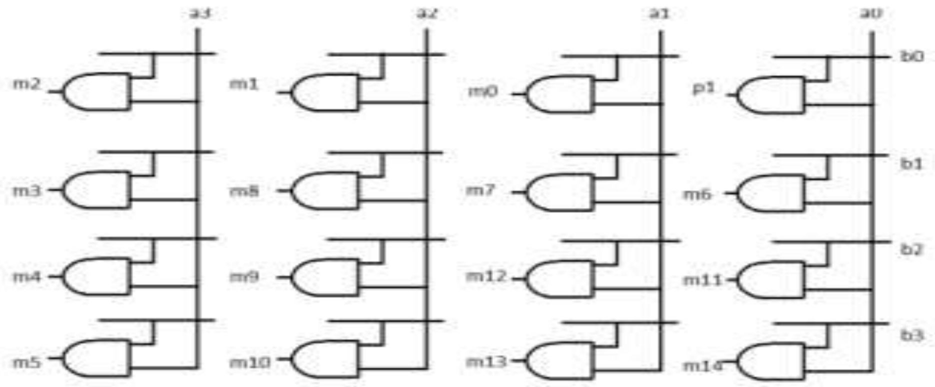


Fig 4.2 Product terms generated by a collection of AND gates

5.1 Introduction to Approximate multiplier

We are at the threshold of an explosion in new data, produced not only by large, powerful scientific and commercial computers, but also by the billions of low-power devices of various kinds. While traditional workloads including transactional and database processing continue to grow modestly, there is an explosion in the computational footprint of a range of applications that aim to extract deep insight from vast quantities of structured and unstructured data. There is an exactness implied by traditional computing that is not needed in the processing of most types of these data. Yet today, these cognitive applications continue to be executed on general purpose (and accelerator) platforms that are highly precise and designed with reliability from the ground up. Approximate computing aims to relax these constraints with the goal of obtaining significant gains in computational throughput - while still maintaining an acceptable quality of results.

A primary goal of research in approximate computing is to determine what degrees of approximations in the several layers of the system stack (from algorithms down to circuits and semi-conductor devices) are feasible so that the produced results are acceptable, albeit possibly different from those obtained using precise computation. Approximate computing techniques studied by various researchers have focused primarily on optimizing one layer of the system stack and have shown benefits in power or execution time. In this work we set out to investigate if combining multiple approximation techniques spanning more than one layer of the system stack compounded the benefits, and if these compounded benefits are widely applicable across different application domains.

In order to provide a concrete demonstration, we focused on three approximation categories: skipping computations, approximation of arithmetic computations themselves, and approximation of communication between computational elements. As representatives of each category we evaluated loop perforation, reduced arithmetic precision, and relaxation of synchronization. We selected applications that are computationally expensive but have the potential to significantly impact our lives if they became cheap and pervasive. Our applications spanned the domains of digital signal processing, robotics, and machine learning. Across the set of applications studied, our results show that we were able to perforate hot loops in the studied applications by an average of 50%, with proportional reduction in overall execution time, while still producing acceptable quality of results. In addition, we were able to reduce the width of the data used in the computation to 10-16 bits from the currently common 32 or even 64 bits, with potential for significant performance and energy benefits. In the parallel applications we studied, we were able to reduce execution time by 50% through partial elimination of synchronization overheads.

Finally, our results also demonstrate that the benefits from these techniques are compounded when applied concurrently. That is, combined judiciously, the multiple techniques do not significantly lessen the effectiveness of one another. As the benefits of approximate computing are not restricted to a small class of applications these results motivate a re-thinking of the general purpose processor architecture to natively support different kinds of approximation to better realize the potential to approximate computing.

Rounding technique is one of the most efficient methods for packing the input data before processing. This method has a potential to improve the circuit characteristics such as power and energy consumption, speed and area which is suitable method for the approximate computing. Approximate computing works very well to most of error resilient applications in the field of computer vision, image processing, pattern recognition, signal processing, scientific computing, and machine learning. Over past decade, research on these areas has given lots of opportunities in research. A multiplier is a fundamental block of computation and one of the most resource-consuming operation. We see innumerable research on this front with a significant tradeoff on accuracy and power-delay-energy. Fundamental building blocks of the multiplier are partial product generation, partial product reduction, and packing. This paper proposes rounding technique as a new method for input block prior to partial product generation. Accuracy curve as a criteria plays a critical role in controlling and minimizing the error range to be

considerable depending on applications. Different algorithms are implemented on different levels of multiplier blocks. Input block has a rounding technique for both 16bit and 32bit based on accuracy levels. Generated partial products are divided into either active or inactive partial products. Inactive partial products are all zeros and hence are not needed to be considered in the reduction process with compressors.

7.5 XILINX Design Process

Step 1: Design entry

- HDL (Verilog or VHDL, ABEL x CPLD), Schematic Drawings, Bubble Diagram

Step 2: Synthesis

- Translates .v, .vhd, .sch documents into a netilist record (.ngc)

Stage 3: Implementation

- FPGA: Translate/Map/Place and Route, CPLD: Fitter

Stage 4: Configuration/Programming

- Download a BIT document into the FPGA
- Program JEDEC document into CPLD
- Program MCS document into Flash PROM

Recreation can happen after stages 1, 2, 3

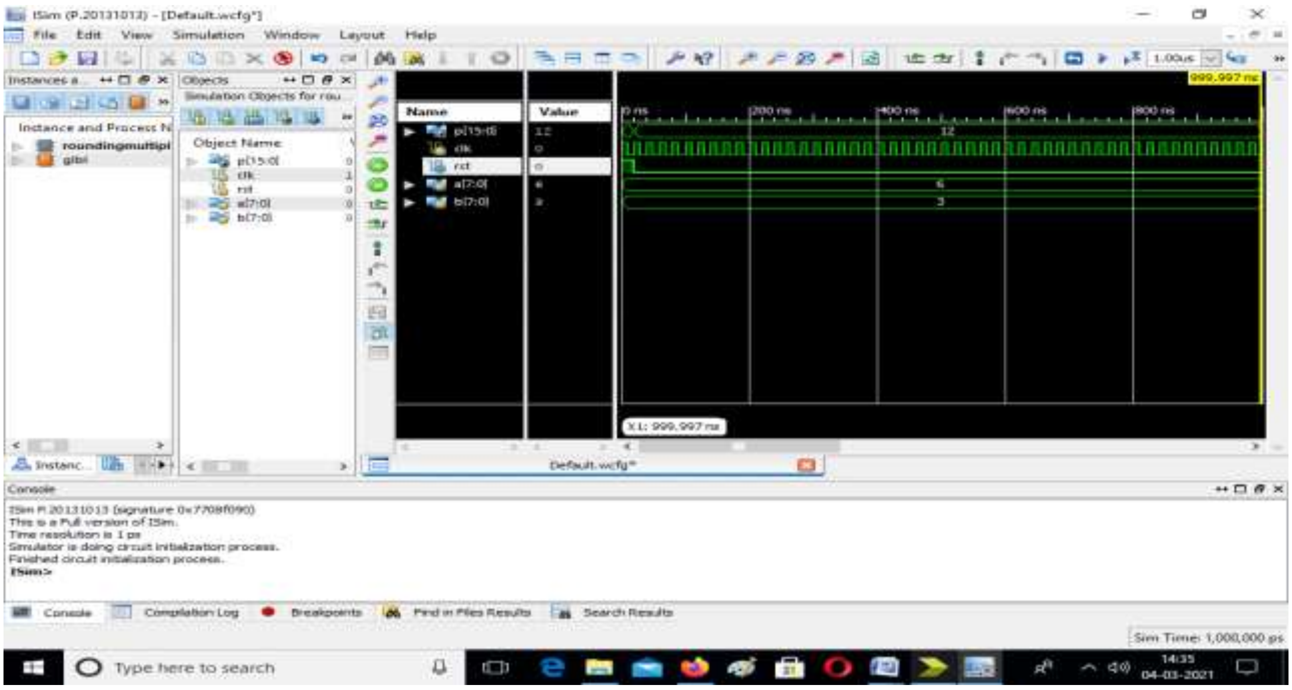
XILINX ISE 13.2 is used for regeneration and synthesis as part of this principle. The programs are dialectally structured.

8.1 SIMULATION RESULTS

This section evaluates the performance of the proposed multiplier and shows the simulation results. For the implementation of this project the software tool required is Xilinx ISE 14.7. The simulation and synthesis results are executed by using Xilinx ISE 14.7 This project uses the spartan3e for synthesis of the design. The code is developed using the Verilog HDL Language.

Family	sparten3e
Device	XC3s1200E
Package	FG320
Speed	-5
Synthesis tool	XST (VHDL/Verilog)

Simulator	ISIM (VHDL/Verilog)
Preferred language	Verilog



```
ISim P.20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

8.2 Block diagram

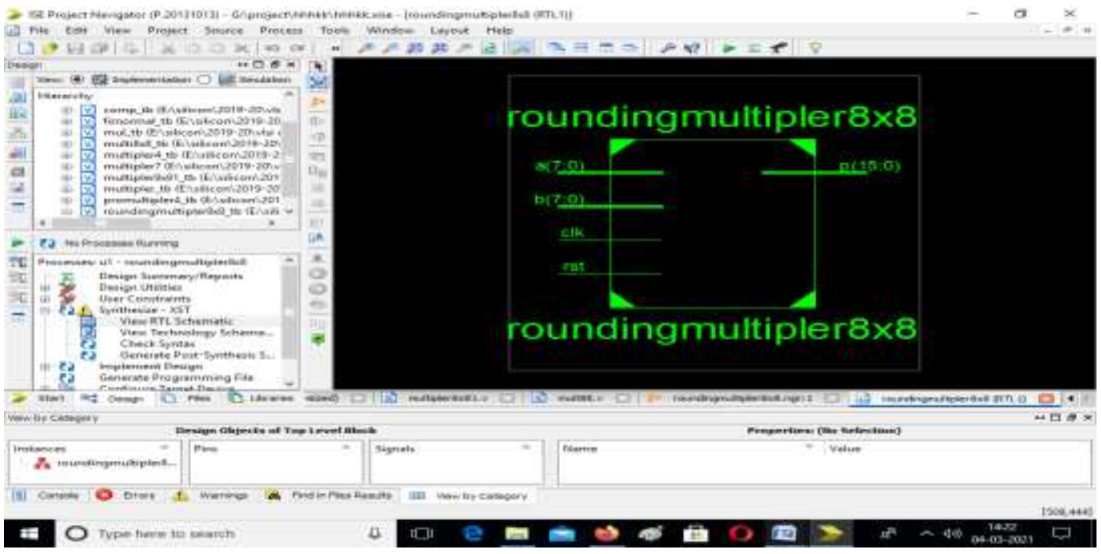


Fig 8.2. Showing Block diagram of proposed adder

The above block diagram shows the proposed adder . It consists of a, b, clk, rst as input and p as output

Device utilization summary:

Selected Device : 3s1200efg400-5

Number of Slices:	85	out of	8672	0%
Number of Slice Flip Flops:	3	out of	17344	0%
Number of 4 input LUTs:	147	out of	17344	0%

Number of IOs:	35			
Number of bonded IOBs:	34	out of	250	13%
Number of GCLKs:	2	out of	24	4%

8.5 Timing Summary:

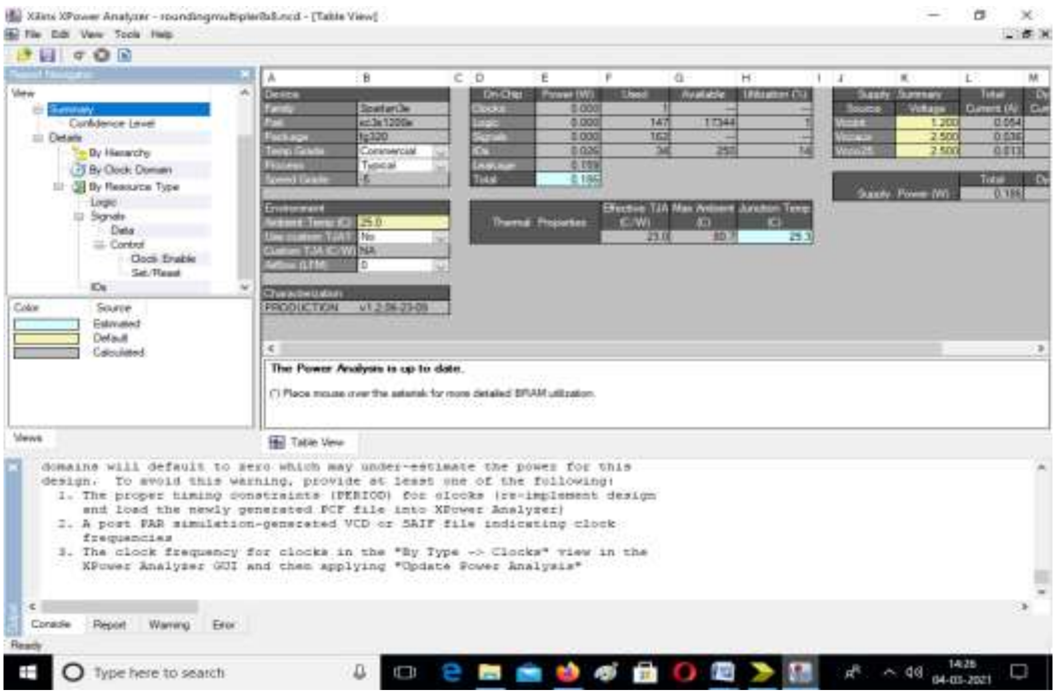
Minimum period: No path found

Minimum input arrival time before clock: 4.128ns

Maximum output required time after clock: 23.598ns

Maximum combinational path delay: 22.389ns

8.6 Power report



Xilinx XPower Analyzer - roundingmultiplier8x8.ncd - [Table View]

Device	
Family	Spartan3e
Part	xc3s1200e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-5

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	147	17344	1
Signals	0.000	162	---	---
IOs	0.026	34	250	14
Leakage	0.159			
Total	0.186			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	23.0	80.7	29.3

Supply Source	Summary Voltage	Total Current (A)
Vccint	1.200	0.054
Vccaux	2.500	0.036
Vcco25	2.500	0.013

	Total
Supply Power (W)	0.186

Color	Source
▭	Estimated
▭	Default
▨	Calculated

The Power Analysis is up to date.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

```
domains will default to zero which may under-estimate the power for this
design.  To avoid this warning, provide at least one of the following:
  1. The proper timing constraints (PERIOD) for clocks (re-implement design
     and load the newly generated PCF file into XPower Analyzer)
  2. A post PAR simulation-generated VCD or SAIF file indicating clock
     frequencies
  3. The clock frequency for clocks in the "By Type -> Clocks" view in the
     XPower Analyzer GUI and then applying "Update Power Analysis"
```

Fig 8.4 Power Report

The above block diagram shows the power report of proposed adder. Maximum power consumption is 0.286 w

8.7 compression table

	EXISTING ADDER	PROPOSED ADDER
DELAY	32.681ns	21.389ns
AREA	143	147
POWER	0.304w	0.186w
SPEED	30.598Mhz	46.753Mhz

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

Proposed algorithm proves to be best in terms of power-area delay and PDP efficiency when compared to other algorithms for both signed and unsigned data (16-bit and 32-bit). This is the primary investigation of rounding technique on approximate multiplier by having one method of rounding pattern which are fixed active partial product rows. With this rounding pattern, we see potential areas of less accuracy and areas with better accuracy corresponding to probability of rounding value. Based on accuracy required, rounding patterns are changed with a little extra expense of hardware.

Rounding pattern can be modified to have fixed or dynamic partial product rows and yet have fewer active partial product rows for compression. The proposed algorithm can be used in the wide range of applications in image processing, machine learning and signal processing. Thus, different weights based on the bit position of '1' plays an important role to keep accuracy relatively near to the conventional method. With flexible reduction of partial products, proposed algorithm produces great hardware characteristics, when compared to DRUM. The modified multiplier, which had high accuracy based on roundingof the inputs in the form of $2n$. In this way, the computational intensive part of the multiplication was omitted improving speed and energy consumption at the price of a small error.

9.2 Future scope

This multipliers plays a very important role in our day to day life. In future the multipliers are going to play a major role. The speed of the multipliers are increased by using carry save adders, carry look ahead adder, and so on. Rounding patterns will be optimized based on required accuracy and different compression techniques. The area and delay can be reduced in future by using advanced technology.

BIBLIOGRAPHY

[1] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[3] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. Int. Conf. Comput.-Aided Design, Nov. 2011, pp. 667–673.

[5] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, "New approximate multiplier for low power digital signal processing," in Proc. 17th Int. Symp. Comput. Archit. Digit. Syst. (CADS), Oct. 2013, pp. 25–30.

[6] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Int. Conf. VLSI Design, Jan. 2011, pp. 346–351.

[7] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in Proc. Conf. Design Archit. Signal Image Process., 2009, pp. 97–104.

[8] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC), Dec. 2010, pp. 1–4.

[9] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[10] K. Bhardwaj and P. S. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip, 2013, pp. 1–6.

[11] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in Proc. 15th Int. Symp. Quality Electron. Design (ISQED), 2014, pp. 263–269.

[12] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electron. Comput., vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

[13] V. Mahalingam and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," IEEE Trans. Comput., vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[14] Nangate 45nm Open Cell Library, accessed on 2010. [Online]. Available: http://www.nangate.com/

[15] H. R. Myler and A. R. Weeks, The Pocket Handbook of Image Processing Algorithms in C. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.