 CMC, vol.53, no.1, pp.37-47, 2017

Bus Encoded LUT Multiplier for Portable Biomedical Therapeutic Devices

R. Praveena[1] and S. Nirmala[2]

Abstract: DSP operation in a Biomedical related therapeutic hardware need to be performed with high accuracy and with high speed. Portable DSP hardware's like pulse/heart beat detectors must perform with reduced operational power due to lack of conventional power sources. This work proposes a hybrid biomedical hardware chip in which the speed and power utilization factors are greatly improved. Multipliers are the core operational unit of any DSP SoC. This work proposes a LUT based unsigned multiplication which is proven to be efficient in terms of high operating speed. For n bit input multiplication n*n memory array of 2n bit size is required to memorize all the possible input and output combination. Various literature works claims to be achieve high speed multiplication with reduced LUT size by integrating a barrel shifter mechanism. This paper work address this problem, by reworking the multiplier architecture with a parallel operating pre-processing unit which used to change the multiplier and multiplicand order with respect to the number of computational addition and subtraction stages required. Along with LUT multiplier a low power bus encoding scheme is integrated to limit the power constraint of the on chip DSP unit. This paper address both the speed and power optimization techniques and tested with various FPGA device families.

Keywords: Constant coefficient multipliers, reduced coefficient multipliers, bus encoding, DSP SoC, look up table, barrel shifter, pre-processing.

1 Introduction

Portable Biomedical devices are the state of art invention of modern technology and play a vital role in knowing the state of an organ and helps in treatments. Main consideration of the portable biomedical devices is to provide high speed accurate results by consuming low power. Since almost all of the portable biomedical chips operate with battery/solar backup it is necessary to design the VLSI architecture to address these major constraints. Bus invert coding [Stan and Burleson (1995)] is first used for serial bus architecture in which single bit encoding is achieved for every clock cycle to reduce the dynamic power dissipation. Bus invert coding is further developed into partition invert coding [Jae-Yoon Sim (2007)]. This is for parallel bus architecture in which total bus length will be divided to ensure high throughput and low power utilization. Shift invert bus encoding method introduced by [Natesan and Radhakrishnan (2004)] made multiple bus encoding possible.

[1] Sengunthar college of Engineering, Tiruchengode, e-mail: praveenajuhi@gmail.com
[2] Muthayammal Engineering College, Rasipuram, e-mail: nirmala.ramkamal@gmail.com

It ensures different encoding schemes can be applied for every transaction by comparing the hamming distance between various available encoding scheme (Left shift, Right Shift, Invert). The problem of extra indication bits required for notifying the receiver about the mode of encoding is reduced through authors own research work Phase Encoded Shift invert coding (PESHINV) [Praveena and Nirmala (2017)]. PESHINV uses dual clock edge triggering to encode the multiple indication bits into a single bus line, which provides maximum power and area reduction during on Chip network operations and also achieves high speed operation due to its parallel bus architecture. Various novel architectures for general and application oriented multiplication has been proposed throughout the literature.

Major concern of speed and area utilization in multiplier design has opened a new research era, since multipliers are the basic computation unit in most of the Digital signal processing applications. Many mathematical transformation supporting digital signal processing applications depends upon the computational speed and accuracy of the multiplier unit. A typical multiplier which involves in generation of partial products and static additional steps is proven to be ineffective due to its hardware requirement of n adder stages and shifting cycles after partial product generation. To obtain multiplication result in a single execution cycle, a design of Look up Table based multiplier is proposed with huge memory expenses. These typical LUT architecture requires n*n memory array locations with a bit size of $2n$, even though it achieves efficient multiplication output. It is not practically recommendable due its exponential raise in memory size along with the input bit width. Two novel techniques have been introduced to reduce more than half of the LUT size, Anti-symmetric product coding (APC) is utilized to reduce 50% of LUT memory utilization and an easy method of sign bit exclusion is used to reduce the memory utilization further [Pramod kumar meher (2009)].

Reduced coefficient multipliers (RCM) and Constant coefficient multipliers (KCM) has been analysed in detail to derive a highly efficient limited range working multiplier. The designed multiplier is tested under Xilinx Virtex FPGA platform with an application of DCT poly phase filter [Turner and WOODS (2004)]. A Modified approach of APC and odd multiple storage (OMS) technique is designed for improvising DSP applications. An optimised LUT memory utilization is achieved by a combined approach of APC, OMS and a selective sign reversal [Pramod kumar meher (2010)]. A high precision shift-save-accumulation controller with a memory based multiplication approach is introduced to achieve a 42% area and 38% area-delay product. This method is modelled and tested with synopsis design ware with 0.18 microns library [Pramod kumar meher (2009)]. High precision input coding LUT multiplier is compared with APC technique and the synopsis tested experimental results show the proposed input coding utilize significantly less area and time overheads than APC method [Meher (2010)]. A performance evaluation between 4 input and 6 input LUT FPGA architecture is carried with signed adders, multipliers and FIR filter implementation and the results shows that the number system representation of the previous generation FPGA architectures fits optimally with the LUT new generation FPGA architectures [Cardarilli, pontarelli and Salsano (2008)]. A low power encoded multiplier is proposed in [Prabhu, Mangalam and Karthick (2013)] in which zero partial products are removed using bypassing techniques. The selection of encoder and decoder by storing the partial products in a register. A modified version odd multiple storage (OMS) is introduced in [Indrajit Shankar Acharya and Ruhan Bevi (2013)] through which

signed and unsigned multiplications are performed with reduced memory size. A typical design of cryptographic multiplier is detailed in [Dani George and Bonifus (2013)] by employing ancient vedic mathematics and the required adders in the multiplier is reduced to half. A high speed parallel multiplier design with redundant binary encoding is published in [Keote and Karule (2016)] and proved that redundant binary multiplier allows carry-free addition. The proposal is done in 64x64 multiplier structure with only 132 LUT size in Xilinx Spartan FPGA.

2 Background

Portable Biomedical devices in the form of system on chip architecture consists of basic units like Sensors, Power Management circuit, Digital Signal Processor, Inbuilt Flash Memories, Transmitter, Input & Output Devices. They use any type of transmission techniques [Alain and Mika (2006)]. Sensors act as interface medium to acquire real time patient data and feed it to the Digital Processing unit for decision. In a portable Bio device various types of sensors are incorporated for measuring pulse rate, heart rate, insulin level etc. Digital signal processor. Microprocessor consists of program memory to hold calculation formulas and I/O controller to operate based on input configuration. The Major portion of the DSP controller is used by the Mathematical processing unit consists of Multiplier, Shifter and ALU circuits.

Most of the power utilization is spent for DSP control unit to execute complex mathematical calculations back and forth using on chip Bus architecture. The speed of the Result is depending of the calculation speed of the arithmetic control unit. Various Multiplier architectures are proposed for high speed and power optimization, this work explores the LUT multiplier with bus encoding architecture. Anti-symmetric product coding (APC) and Odd multiple storage (OMS) are the popular techniques which set the reference for any LUT based memory management multiplication. These techniques works as a compressor algorithm to short the memory requirement of the LUT in half. LUT's are the basic building blocks in modern FPGA design and has the capability to render itself in any form of logical gates. Implementing Multipliers in the form of LUT rather than conventional adder circuit is much preferable for FPGA circuits where memory can be afforded but the speed of operation need to be instant.

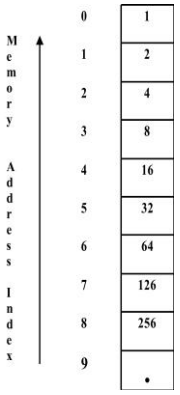

Figure 1: LUT for 2n radix storage

Lookup table is nothing but a memory structure (Fig.1) in which a pre-defined reference values will be stored which will act as a Look up reference for standard inputs. In this proposal the LUT memories will be the filled with 2^n vector values in sequence. The power coefficient will be used to decide the shifting level of the barrel shift multiplier. Multiply using shift addition is a popular technique where the multiplicand can be expressed as a polynomial equation of base power 2. The value of 16 will be represented as 2^4 and the value 17 will be represented as 2^4+1. This is to find the no of shift operation required for performing the multiplication.

For example, 5*8 can be done be left shifting the 5 for 3 times 5<<3, where 3 is the base representation of $8(2^3)$. i.e. 5*8 can be written as a 8 bit representation: $00000101_2 << 3$, then result will $00101000_2 (40_{10})$ which is same as 5*8. The major issue of this method is it cannot operate other than 2^n sequence. A polynomial sequence need to be derived to add further steps of arithmetic operation after shifting. Power optimization techniques in VLSI architecture mostly concentrate on reducing the switching activity of the chip. Bus encoding schemes also follows the same principle of reducing the switching activity using various encoding mechanisms [Abinesh et al. (2009); Park and Kim (2014); Gu and Guo (2009); Arnavut (2000)]. Shift invert Bus encoding mechanism is one of the popular encoding methods where multiple encoding schemes are employed to achieve maximum power reduction during on chip network transfer. SHINV mechanism can also be used for parallel bus encoding schemes for faster transmission.

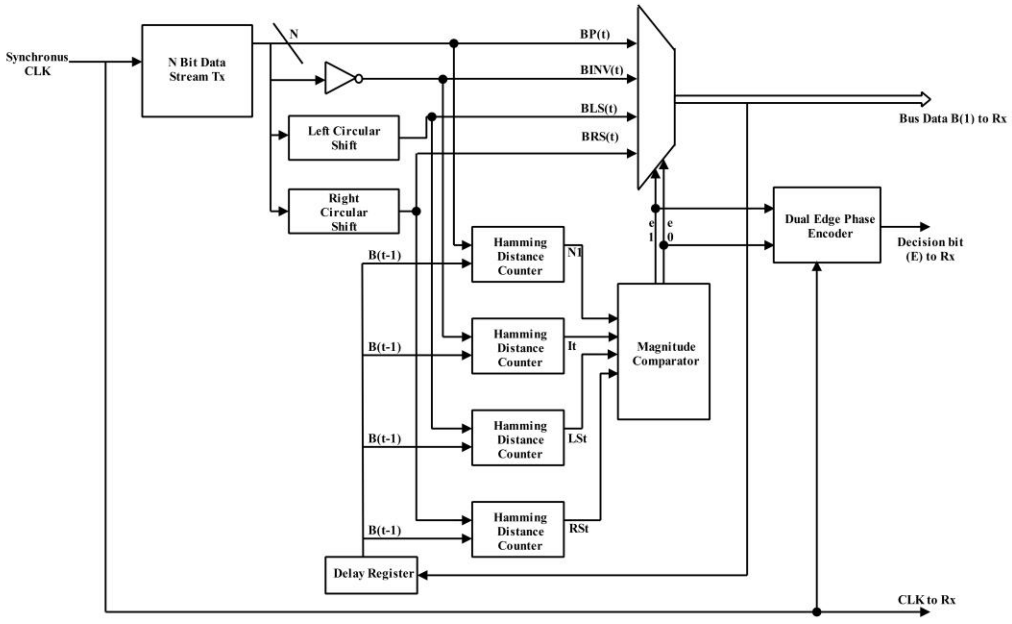

Figure 2: Phase encoded shift invert bus encoder

Encoding scheme will be decided by comparing the hamming distance between the present and previous bus data's. Hamming Distance counting is shown as: In case of two input binary datas we can calculate hamming distance by doing XOR operation on the two strings and by adding the number of 1's present in the result will be the hamming distance.

1 0 0 1 0

1 1 0 0 0

---

0 1 0 1 0 (Hamming distance is two as number of 1 in the result is 2)

The arithmetic expression for the Selecting Control signals based on lowest Hamming distance score is given in eq.1.

$$(B^{(t)}, C_1^{(t)} C_0^t) = \begin{cases} (b^{(t)},00) H_{\min} = H \\ (b^{INV(t)},01) H_{\min} = H^{INV} \\ (b^{LS(t)},10) H_{\min} = H^{LS} \\ (b^{RS(t)},11) H_{\min} = H^{RS} \end{cases} \quad (1)$$

Two extra encoding bits need to be transmitted for informing the receiver about the mode of encoding. Using the phase encoder SHINV encoding (Fig.2) the extra encoding bits need to be transmitted along with the data bits will be reduced with dual edge triggered clock gated circuit.

3 Proposed system design

This proposal builds the principle of multiplication can also be done through shifting. If any one of the multiplicand is in 2^n sequence then multiplication can be take place by shifting another multiplier with 'n' times. This type of multiplication required a shifting sensing logic along with a parallel shifter. The address generator logic of the proposed LUT is modified from existing one which predicts not only the matching address but also finds the lower and higher boundaries for unmatched address.

Direct LUT coefficient multiplication is not always possible if the multiplicand is not a multiplicand of 2. i.e. 5*14 cannot be directly represented using 2^x series, So, 14 can be represented as 2^3+6 this will evaluated as 3 shift and 6 addition operations in the DSP unit. Arithmetic operation can be reduced in some cases if subtraction is performed instead of addition. Example 14 can also be represented as 2^4-2. Which requires 4 shifting but two subtraction. So, with a trade-off of one shifting operation we can gain 4 reduced arithmetic steps. The ultimate aim of the proposed work is to find the nearest polynomial conversion to reduce arithmetic and shifting operations.

5*14 can be written as (5<<3) + (5*6)

00000101_2 << 4 + (5 * 6)

$00101000_2 + 00011110_2 = 01000110_2 = 70$

Another speculation is that instead of focusing only multiplicand we can alter the multiplier i.e for 5*14 instead of tweaking 14 can be express the multiplier 5 as 2^2+1(two shift and 1 addition). Which greatly reduces the total number of shifting and arithmetic steps. The polynomial equation can be word in either addition or subtraction. Main objective of the proposed work is to find the minimal number of steps (shifting and addition) required to achieve effective multiplication. Serial adders is preferred over parallel adders since the number of addition/subtraction steps are dynamic in nature

corresponding to the multiplier or multiplicand.

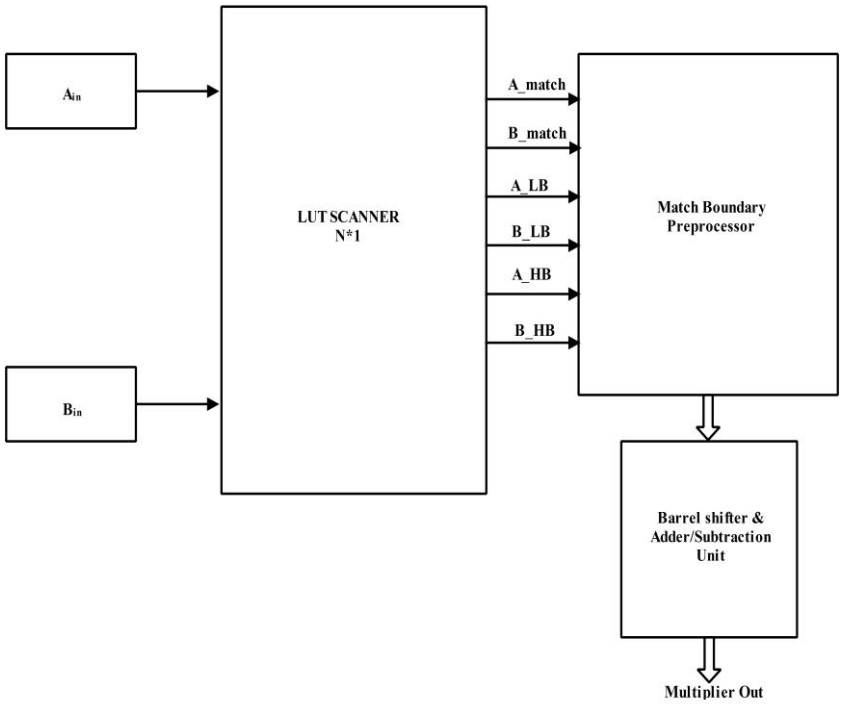

Figure 3: Block diagram of Proposed LUT Multiplier

Ain, Bin are the two input Multiplicands, the boundary selector architecture (Fig.3) scans for exact matching of 2^n Series in both multiplicands. In case of no direct matching in multiplicands, then match boundary processer, find the lower and higher boundary of both the multiplicands. The lower and higher boundaries servers as a reference to find the number of addition / subtraction steps needed to achieve multiplication. In actual cases the by increasing a shifting operation many number of serial arithmetic operation can be reduced.

3.1 Algorithm steps

- Scan be Direct matching of Multiplicand
- Scan be Direct matching of Multiplier
- If both of the above statements are true compare the number of shifting operations required for multiplier and multiplicand. Pass control signal to the DSP control unit to perform shifting with lower coefficient multiplicand/multiplier.
- If one of the above (1&2) statement is true perform shifting with direct multiplication coefficient.
- If both of the above (1&2) statement is false, expand the multiplicand with nearest Shif+ADD polynomial and also with SHIFT-SUB Polynomial (Ex: 15 can be represented as 2^3+6 and 2^4-1).
- Find the most nearest Coefficient with reduced shifting and addition operation and

assume it as P1.

- Repeat the steps 5 & 6 for multiplier and assume the result as P2.
- Compare P1&P2 to find the optimal result solution with lowest shift and arithmetic operation.
- Pass these results to memory or to Transmitter through PESHINV encoded Bus architecture.

The straightforward matching might not tangible in all the input cases. An example of non-matching scenario is explained with operative details of the proposed LUT Multiplier.

3.2 Proposed LUT example

The pseudo code for the method is given below

```
functioncalculate_Score()
        Input  ain;
        Output [add,sub,match_flag,shift,addshift,
subshift,alower_index,ahigher_index];
initializemax_size=2048
initialize lut={2,4,8,16,32,64,128,256,1024,…max_size}
fori in 0 to lutsize
        if(lut[i]==ain)
                shift=i;
                add=-1; //not used
                sub=-1;
                match_flag=1;
                alower_index=-1;
                ahigherindex=-1;
                addshift=-1;
                subshift=-1;
                break;
else if(lut[i]>ain&&i>0)
                addshift=lut[i-1];
                subshift=lut[i];
                alower_index=i-1;
                ahiger_index=i;
                add=ain-lut[i-1];
                sub=lut[i]-ain;
                matchflag=0;
        end
function main()
```

```
        inputain,bin;
        outputmul_out;
readain,bin
[a_add,a_sub,a_match_flag,a_shift,a_addshift,a_subshift,a_lower_index,a_higher_index]
=calculate(ain)
[b_add,b_sub,b_match_flag,b_shift,b_addshift,b_subshift,b_alower_index,b_ahigher_ind
ex]=calculate(bin);
if(a_match_flag==1 &&b_match_flag==1)
        if(a_shift<b_shift)
                mul_out=shift(bin,a_shift);
        else
                mul_out=shift(ain,b_shift);
else if(a_match_flag==1)
        mul_out=shift(bin,a_shift);
else if(b_match_flag==1)
        mul_out=shift(ain,b_shift);
else
        [aflag,bflag]=compare(a_add,a_sub,a_add_shift,a_sub_shift,b_add,b_sub,b_add_
shift,b_sub_shift);
        if(aflag==1)
                tmp=shift(bin,a_addshift);
                mul_out=barallel_add(tmp,ain,a_add);
        else if(aflag==2)
                tmp=shift(bin,a_subshift);
                mul_out=barallel_sub(tmp,ain,a_sub);
        else if(bflag==1)
                tmp=shift(ain,b_addshift);
                mul_out=barallel_add(tmp,bin,b_add);
        else if(bflag==2)
                tmp=shift(ain,b_subshift);
                mul_out=barallel_sub(tmp,bin,b_sub);
```

*X= 7 * 9 =63;*

For multiplicand '7' polynomial Expression is 2^2+3 for 2 SHIFT + 4 ADD and 2^3-1 for 3 SHIFT- 1 SUB. Then P1 will be 2^3-1 for 3 SHIFT - 1 SUB. Which include 1 shift loss but 2 arithmetic savings.

For multiplier '9' polynomial Expression is 2^3+1 for 2 SHIFT + 1 ADD and 2^4-7 for 4 SHIFT-7 SUB. Then P2 will be 2^{3+}1 for 3 SHIFT + 1 ADD. Which include 1 shift loss but 2 arithmetic savings.

By comparing P1 & P2 both doesn't differ anything so P1 expression will be executed for default.

Thus, DSP processor write these expressions dynamically and improves the multiplication speed 20% better than existing method.

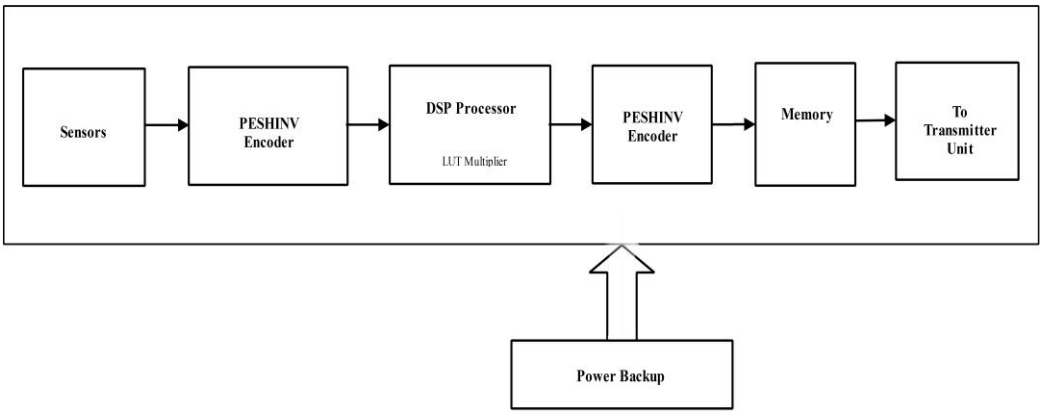

Figure 4: Block diagram of the proposed portable DSP SoC

The final architecture of the proposed LUT multiplier along with PESHINV bus encoding method is shown in fig.4. After completion of every successful mathematical operations the result of the proposed LUT multiplier need to be stored in memory or it can be transmitted to any Internet of Thing based receiver for live patient monitoring. Each interface between the DSP unit and the other Bio circuit components will be encoded with proposed PESHINV architecture which greatly controls the power dissipation. The power dissipation and area utilization of the proposed architecture is modelled using VHDL SoC design and the result evaluated with FPGA families.

The power dissipation of the normal LUT and proposed method is tabulated in table.1. For the analysis Cyclone II, Cyclone III, Cyclone IV GX, Stratix II and Stratix III. Table 2 shows the time delay comparison of proposed architecture with various existing bit multiplier. The performance comparison of existing and proposed DSP processor is presented in table.3.

Table 1: Power dissipation comparison in various device families.

Device / Attributes	Power Dissipation	
	Normal LUT	Proposed
Cyclone II	67.86 mW	32.73 mW
Cyclone III	55.92 mW	74.25 mW
Cyclone IV GX	144.29 mW	158.29 mW
Stratix II	323.82 mW	158.29 mW
Stratix III	397.35 mW	408.00 mW

Table 2: Time delay comparison of proposed architecture with various existing bit multiplier

Multiplier n Bit / Attributes	Setup Time Delay (tsu)	
	Existing	Proposed
8 Bit	2.20 ns	3.446 ns
16 Bit	4.11 ns	3.595 ns
32 Bit	7.41 ns	4.472 ns

Table 3: Performance comparison of existing and proposed DSP processor

Attributes	Time	Power
DSP Processor without LUT & Encoding	7.54 ns	144.29 mW
DSP Processor with LUT & Without Encoding	4.20 ns	152.73
DSP Processor with LUT & with Encoding	5.1 ns	91.3mW

4 Conclusions

A hybrid architecture of optimized LUT multiplier with Bus encoding mechanism is designed using HDL code behavioural verification of the encoding and multiplier operations are done using Testbench waves using Modelsim Tool. Hardware Synthesis and Netlist generations are analysed using Altera Quarts II synthesizer. Match boundary pre-processor calculates the computation and power efficiency to decide which way of multiplication will be efficient for power and time savings. Analytical result for various FPGA families has been tabulated for switching power dissipation and setup time delay units. With the statistical conclusions our proposed approach is proven to be better efficient in higher order multiplication circuits. A 8 bit delay comparison between APC and proposed method shows that 1 ns lag in processing delay but greatly leads in 16 and 32 bit multiplication. The complex setup time delay is unnecessary for scanning small order multiplicands and can be operated better only in higher order multiplicands. The overall proposed delay product is improved 30% from existing APC techniques.

References

Abinesh, R. et al. (2009): Transition inversion based low power data coding scheme for synchronous serial communication. *Proc. IEEE Comput. Soc. Annu. Symp. VLSI Conf.*, Tampa, FL, pp.103-108.

Arnavut, Ziya (2000): Move-to-Front and Inversion Coding. *Data Compression Conference*, Snowbird, UT, pp.193-202.

Alain, J. Martin; Nystrom, Mika (2006): Asynchronous techniques for system-on-chip design. *Proceedings of the IEEE*, vol. 94, no.6, pp. 1089-1120.

Cardarilli, G.C.; Pontarelli, S.; Re, m.; Salsano, A. (2008): *On the Use of Signed Digit Arithmetic for the New 6-Inputs LUT Based FPGAS*. IEEE, transactions 2008.

George, Dani; Bonifus, P. L. (2013): RSA encryption system using encoded multiplier and vedic mathematics. Advanced Computing and Communication Systems (ICACCS). *2013 International Conference on*, pp.19-21.

Gu, Ji; Guo, Hui (2009): A Segmental Bus-Invert Coding Method for Instruction Memory Data Bus Power Efficiency. *Proceeding of the 2009 IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, pp.137-140.

Keote, R.S.; Karule, P.T. (2016): VLSI Design of 64bit×64bit High Performance Multiplier with Redundant Binary Encoding, IEEE.

Meher, Pramod kumar (2009): *New Look-Up-Table Optimizations for Memory-Based Multiplication*. IEEE.

Meher, Pramod Kumar (2010): Lut Optimization for Memory-Based Computation IEEE, *Transactions on circuits and systems-ii: express briefs*, vol.57, no.4.

Meher, Pramod Kumar (2009): New approach to LUT implementation and accumulation for memory-based multiplication, *Transactions*. IEEE.

Meher, Pramod Kumar (2010): Novel Input Coding Technique for High-Precision LUT-Based Multiplication for DSP Applications. IEEE.

Natesan, J.; Radhakrishnan, D. (2004): Shift invert coding (SINV) for low power VLSI, *Euromicro Symposium on Digital System Design*, pp. 190-194.

Prabhu, E.; Mangalam, H; Karthick, S. (2013): A Low Power Multiplier Using Encoding and Bypassing Technique. *Journal of Theoretical and Applied Information Technology*, vol. 57, no.2, pp.251-260.

Praveena, R.; Nirmala, S. (2017): An Efficient Low Power Encoding Methodology for Data Transmission in Biomedical System on Chip Architecture. *International Journal of Printing, Packaging & Allied Sciences*, vol. 5, no. 1, pp.795-806.

Park, Daejin; Tag Gon, Kim (2014): Built-In Binary Code Inversion Technique for On-Chip Flash Memory Sense Amplifier with Reduced Read Current Consumption. *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 22, no. 5, pp. 1187-1191.

Shankar Acharya, Indrajit; Bevi, Ruhan (2013): LUT Optimization for Memory Based Computation using Modified OMS Technique. *International Journal of Advanced Electrical and Electronics Engineering*, vol.2, no.5, pp.71-75.

Sim, Jae-Yoon (2007): Segmented Group-Inversion Coding for Parallel Links. *IEEE Transactions on circuits and systems-ii: express briefs*, vol.54, no.4, pp. 328-332.

Stan, M.R.; Burleson, W.P. (1995): Bus-Invert Coding for low-power I/O. *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, pp. 49-58.

Turner, H.; WOODS, R. F. (2004): Highly Efficient, Limited Range Multipliers for Lut-Based Fpga Architecture. *IEEE transactions on very large scale integration (vlsi) systems*, vol.12, no.10.