Article

Hardware Optimization for Effective Switching Power Reduction during Data Compression in GOLOMB Rice Coding

G. Neelima[1], T. Srinivas Reddy[2], Sakthivel R[3,*], Vanitha M[4] , Srete Nikolovski[5] and Hossam Kotb[6]

[1] Department of CSE, Vignan Institute of Information Technology, Visakhapatnam, India; gullipalli.neelima@gmail.com
[2] Department of ECE, Malla Reddy Engineering College, Maisammaguda, Secunderabad, 500100, India.; srinivasreddy.thumu@gmail.com
[3] Vellore Institute of Technology, School of Electronics Engineering, Vellore, Tamil Nadu, India; rsakthivel@vit.ac.in / dhillsakthi@gmail.com
[4] Vellore Institute of Technology, School of Information and Technology, Vellore, Tamil Nadu, India; mvanitha@vit.ac.in
[5] Power Engineering Department, Faculty of Electrical Engineering Computing and Information Technology, 31000 Osijek, Croatia; srete.nikolovski@ferit.hr
[6] Department of Electrical Power and Machines, Faculty of Engineering, Alexandria University, Alexandria Egypt; hossam.kotb@alexu.edu.eg

* Correspondence: srete.nikolovski@ferit.hr; rsakthivel@vit.ac.in

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Energies* **2022**, *15*, x. https://doi.org/10.3390/xxxxx

Academic Editor: Firstname Lastname

Received: date
Accepted: date
Published: date

Abstract: Loss-less data compression becomes the need of the hour for effective data compression and computation in VLSI test vector generation and testing in addition to hardware AI/ML computations. Golomb code is one of the effective technique for lossless data compression and it becomes valid only when the divisor can be expressed as power of two. This work aims to increase compression ratio by further encoding the unary part of the Golomb Rice (GR) code so as to decrease the amount of bits used.The algorithm was developed and coded in Verilog and simulated using Modelsim. This code was then synthesized in Cadence Encounter RTL Synthesizer. The modifications carried out show around 6% to 19% reduction in bits used for a linearly distributed data set. Worst-case delays have been reduced by 3% to 8%. Area reduction varies from 22% to 36% for different methods. Simulation for Power consumption shows nearly 7% reduction in switching power. This ideally suggest the usage of Golomb Rice coding technique for test vector compression and data computation for multiple data types, which should ideally have a geometrical distribution.

Keywords: Golomb, compression, Electrocardiogram, Test, Data Compression, Encoding scheme, area, power, delay.

1. Introduction

Data compression refers to use of any particular technique that reduces the number of bits used to represent the same amount of data. This is done using a variety of techniques that have been vastly explored in recent decades. Often, it can be found that the type of compression technique used, usually pertains to the specific application where the said compression technique is intended for.

Golomb Rice coding has been used in data compression for a long time. Image compression seems to be its earliest uses [1]. This technique has been prevalently used for image compression in medical field [2]. This particular compression technique has been fairly popular among researchers working with ECG data.Electrocardiogram (ECG) data is a recording of electrical motion of a living heart over a period of time which is produced using electrodes placed on the patient's body. Different ECG compression techniques like TURNING POINT, AZTEC, CORTES, FFT and DCT have been implemented and studied [3-5]. Several other research works have focussed on Dictionary based ECG data compression [6].

Golomb Rice coding has also been used for ECG data compression using asynchronous time encoding analog to digital converter [7]. While several other researchers have used adaptive linear prediction with adaptive GR coding to improve results [8].

There have been researches in wavelet based methods for ECG compression and their VLSI implementation [9]. Researchers Nicky H. Bellani & Payal Ghutke have used GR codes to implement their own test vector compression in VLSI. [10]. Several others have implemented GR encoding methods for test data compression [11-14]. Beamforming algorithm architectures are used for medical ultrasound imaging [15] [21]. A 15:4 Approximate Compressor based multiplier is used for image processing [16] -22]. Low power low complexity based lossless data compression were proposed and discussed in [17][23] and [18] [24]. IC design algorithm used in the backend require data compression for thermal aware computing and for test data response compaction as discussed in [19] [25].

2. Literature Survey

There are huge volumes of works which have reported on image and signal compression but there are very few works which are being reported on the hardware development for the compression algorithms for test data compression, ECG signal compression etc.

In addition there are works reported on sensitive date lossless compression. Dominik Rzepka [20] discuss about the lossless data compression for an ECG signal using selective linear prediction methodology and it proves to be more effective for the asymmetrical numerical systems. It also well suit for the multichannel signal.

Hang Wang et al. [21] developed an efficient compression based hardware for reducing the on chip memory area requirement by proposing a line buffer architecture. The proposed structure for the compression algorithm provides a good signal to noise ratio. It increases the throughput with reasonable reduction in the hardware cost.

Seongmoon Wang et al. [22] improves the test data compression ratio to a great extent and thereby increases the fault coverage for the Circuit under Test (CUT) in the Automatic Test Engine (ATE). This works aims for compression in test vectors and its response generated in the Linear Feedback shift Register (LFSR) and the ATE respectively.

Xiaoke Qin et al. [23] worked on bitstream compression technique. Bitstream compression is important in reconfigurable system design since it reduces the bitstream size and the memory requirement. It also improves the communication bandwidth and thereby decreases the reconfiguration time.

Wei Jhih Wang et al. [24] worked on modified version of dictionary-based code compression. Memory is a key factor in embedded system design. Code compression is a technique used in embedded systems to reduce the memory usage. Bit Mask-based code compression is applied in this work to increase the compression ratio without increase in the hardware cost.

HarisLekatsas et al. [25] presents a suitable algorithm that will combine approximate compression techniques with bit-toggling reduction and it explores the various tradeoffs. We take advantage of the approximations introduced to modify codes and reduce bit-toggling, while maintaining the compression performance and decoding speed.

2.1. Existing Golomb Rice Coding Technique

The Golomb-Rice (GR) encoding technique is a part of a larger family of prefix codes formed by S.W. Golomb around 1966 as an alternative to the Huffman coding [26] . Golomb Rice coding takes its name from S.W. Golomb and R. F. Rice. Rice described his own modification of the original GR encoding technique where the divisors are a power of two[27].GR codes are optimally suited for encoding symbols from a data set where the probability distribution is exponential (for some parameters of the exponential distribution). However, for a finite alphabet GR codes are neither optimal nor complete. The GR family of codes is characterised by an important parameter 'M'. This parameter is a non-negative integer.In order to encode any input (non-negative integers), the encoded output is framed in two parts namely *unary* and *different* code. First, the unary part is calculated and then the different part is calculated. These two parts are then concatenated to form a single line of code, which is then known as GR code. Table 1 illustrate the GR encoding scheme for M=4. Table 1 illustrate the GR encoding scheme for M=4.

Table 1: GR encoding example for M=4 [2]

M[n]	Quotient	Remainder	Bit stream
0	0	0	0_00
1	0	1	0_01
2	0	2	0_10
3	0	3	0_11
4	1	0	10_00
5	1	1	10_01
6	1	2	10_10
7	1	3	10_11
8	2	0	110_00
9	2	1	110_01
10	2	2	110_10
11	2	3	110_11
12	3	0	1110_00
13	3	1	1110_01
14	3	2	1110_10
15	3	3	1110_11
16	4	0	11110_00

2.2. *Algorithm*

The GR code algorithm is well explained by figure.1. The code assigned for unary and Golomb code based on the value of M with appropriate bit assignment and the same is being explained below.

Step 1	Fix parameter M to an integer value.
Step 2	For input N to be encoded, find • Quotient, q= int[N/M] • Remainder, r = N modulo M
Step 3	Code word generation: Code format = <Unary Code><remainder code>
Step 4	Unary code: Represent quotient in unary coding i.e. "q" strings of 1's followed by 0.
Step 5	Remainder Code: If (M is power of 2)

```
            Remainder is coded as binary format using $\log_2(M)$ bits.
        Else if (M is not a power of 2)
        Set x = \$ceiling{$\log_2$ (M)}
        If ($r < 2x – M$)
                    Code remainder as plain binary using x-1 bits.
            Else if ($r >= 2x – M$)
Code the number (remainder + $2x – M$) in binary representation using x
bits.
```

Figure 1. Algorithm

2.3. Technical Gaps with the existing system Architecture

In Golomb-Rice code, the M -parameter greatly affects the encoding efficiency. In order to efficiently encode the data, the distribution of the data needs to be studied and accordingly the M - parameter needs to be selected. In this case, the value of M is assumed to be 128 and the input data to be encoded is assumed to be 10-bit in size. This setup enables the 'q' value to vary from as low as zero to as high as 7. The variable 'q' holding the value 7 means that the unary code will be 8 bits at its maximum length. This method does allow a certain compromise on data compression by decreasing the bit length of most used input symbols. However, it also increases the length of lesser-used symbols to a very large extent. This makes GR code unsuitable for use when the data is not completely geometrically distributed. This is well illustrated in Figure.2 with an input bit size of 10.

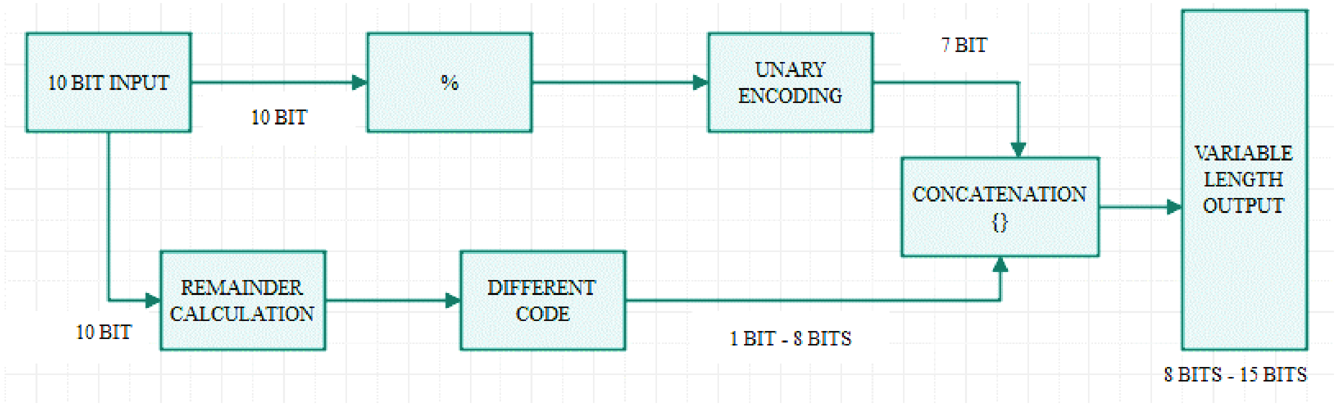

Figure 2. Architecture for 10-bit input

3. Proposed Methodology

This work aims to reduce the bit length used for unary code. The selected value of M as 128 allows us an ideal case to show various types of methods to reduce the length of unary code while not being too large to handle in the scope of this paper. The implementation of a normal GR encoding scheme was carried out using Verilog coding. The outputs were observed and matched with expected results in order to verify whether the code worked with all possible 10-bit inputs. Further, an encoding method was applied which converted all the possible unary codes to fixed length 3 bit code. This helped in reducing the complexity of the code but the bit-length of the output was back to 10-bits for all the inputs, which meant there was no compression in the end. The input was 10 bits and the output was 10 bits as shown in figure.3.

However, this allowed another modification where there was a separate encoding scheme for unary codes. Next, the algorithm was modelled using Verilog, simulated using modelsim and synthesized using Cadence Encounter RTL compiler.This meant including an entire division module and a loop to find the log values for different values of parameter M. These codes were then synthesised. Various reports like timing, area, delay and power were obtained for all proposed 3 schemes and the original GR encoding scheme as shown in figure .4. Comparison of all the data was visualised using graphs. Final conclusion was drawn as to which method was better for which data distribution.

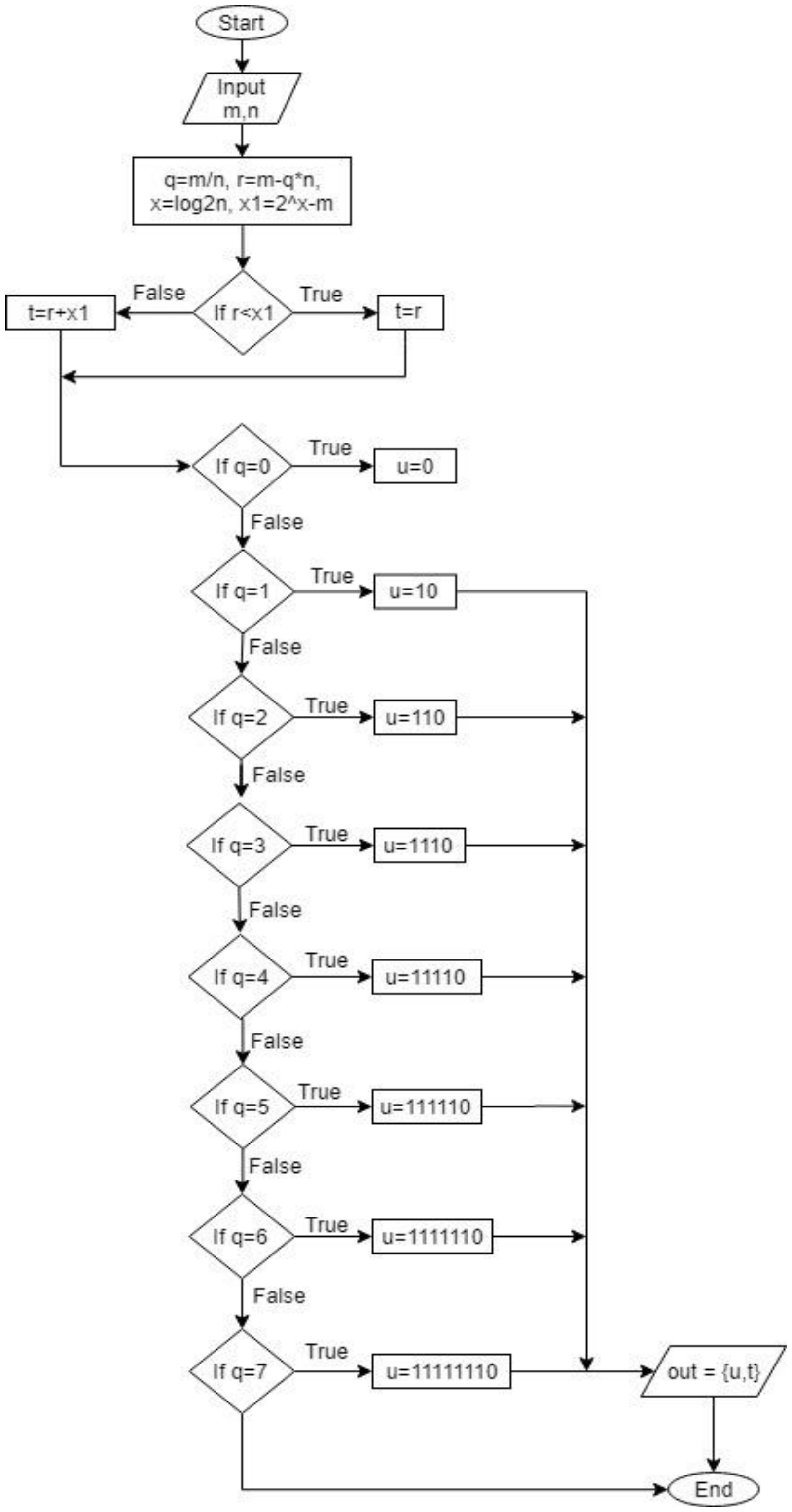

Figure 3. **Flowchart for GR codes**

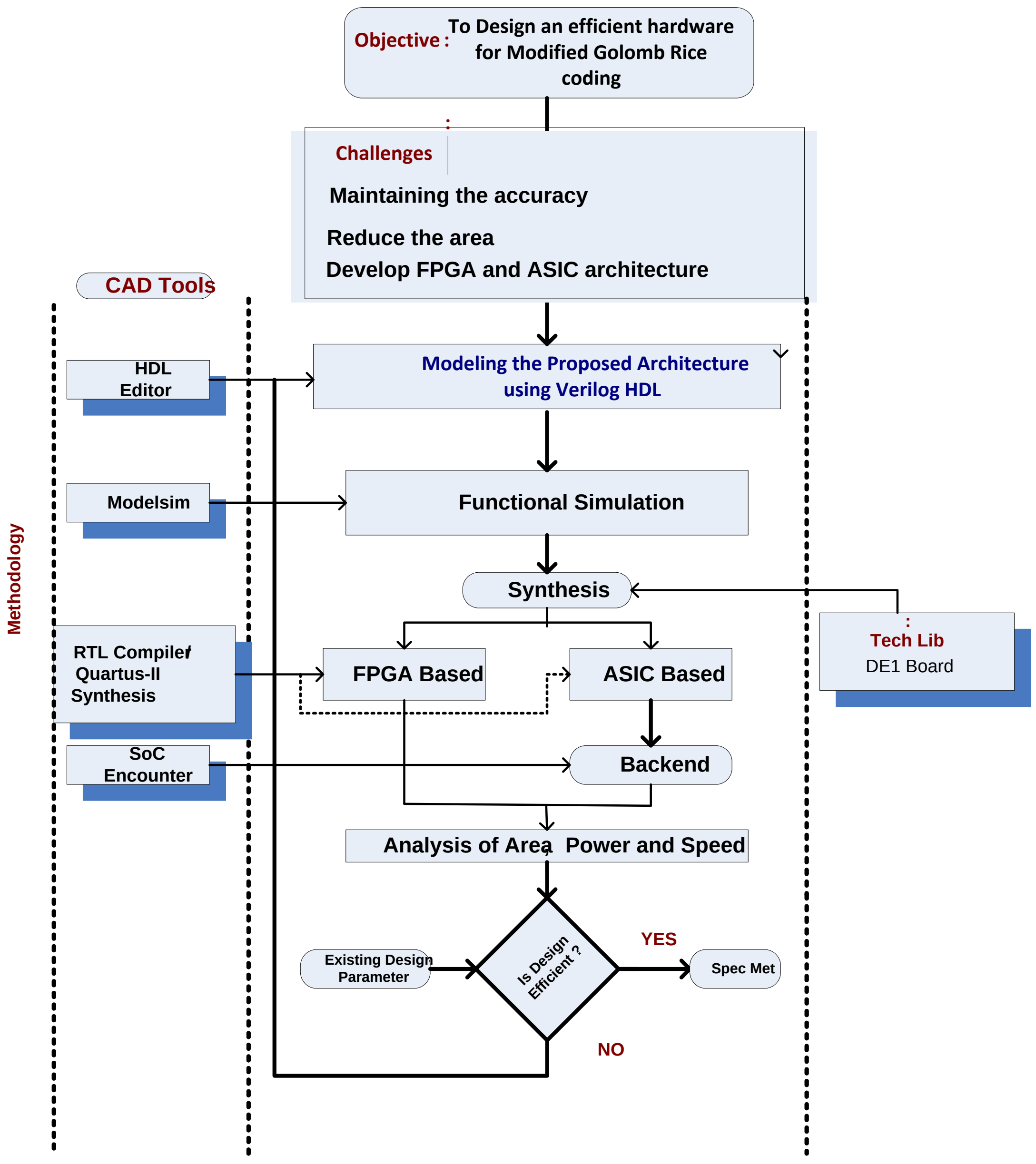
Objective : To Design an efficient hardware for Modified Golomb Rice coding
Challenges
Maintaining the accuracy
Reduce the area
Develop FPGA and ASIC architecture
CAD Tools
HDL Editor
Modeling the Proposed Architecture using Verilog HDL
Modelsim
Functional Simulation
Synthesis
Methodology
RTL Compiler Quartus-II Synthesis
FPGA Based
ASIC Based
Tech Lib
DE1 Board
SoC Encounter
Backend
Analysis of Area Power and Speed
Existing Design Parameter
Is Design Efficient ?
YES
Spec Met
NO

Figure.4. Design methodology followed for the hardware.

3.1. *Proposed schemes*

This scheme tries to reduce the three largest bit sized unary codes by encoding them using 2 bits. This allowed the unary part of the code to shorten its bit length. The codes which were 6,7,8 bit in size were 2 bit each now. The maximum possible bit length of the unary code was 5 bits now. This further led to another two schemes, which used encoding scheme to fix their lengths. In order to decode this code, the bit length needs to determined and accordingly further steps are taken. The possible lengths of unary codes are 1,2,3,4,5 bits. This means that the final combined GR code will be 8,9,10,11,12 bit long. This means whenever the codes are 8,10,11,12 bit long , normal GR decoding will take place which involves finding the leading 0. Whereas for 9 bit long codes the first 2 bits will always determine their unary codes and the remaining 7 bits will correspond to the respective remainder. Similarly for schemes 2 and 3, their unary codes were changed to reduce the bits used.

3.2. *High speed Golomb Rice Code (HSCRC) Proposed scheme-1*

The unary part of the code is modified to decrease the amount of bits used. Here, we have taken the example of a 10-bit input. The value of M is 128. This gives us a q value ranging from [0,7]. The maximum bit size of output will be 15 bits [8 unary bits + 7 remainder bits]. In this, 7 remainder bits are definite in size and are not changed. The unary representation according to GR code will be as shown in Table.2.

Table 2. (HSCRC) Encoding Scheme

• Third bul

Value of "q"	Unary representation	Further encoding	Bits saved
0	0	000	-2
1	10	001	-1
2	110	010	0
3	1110	011	1
4	11110	100	2
5	111110	101	3
6	1111110	110	4
7	11111110	111	5

Use of this encoding technique reduces the number of bits used but it also neutralizes the effect of GR coding. Since, the input was 10 bits in size and the output also remains 10 bit in size, there was no actual compression. This is nothing but just a direct binary representation of the input albeit in a different way.

We needed to encode the input in such a way that the reduction in bit used is significant even in comparison to direct binary representation.Here, we propose a new method to encode the unary part. The last 3 unary codes are represented using two bits instead of their original codes in Table 3. As a result, the concatenated string will be smaller in size as compared to the original as shown in figure 5.

Table 3. (HSCRC)Scheme- 1 Encoding

Value of "q"	Unary representation	Further encoding	Bits saved
0	0	0	0
1	10	10	0
2	110	110	0
3	1110	1110	0
4	11110	11110	0
5	111110	00	4
6	1111110	01	5
7	11111110	11	6

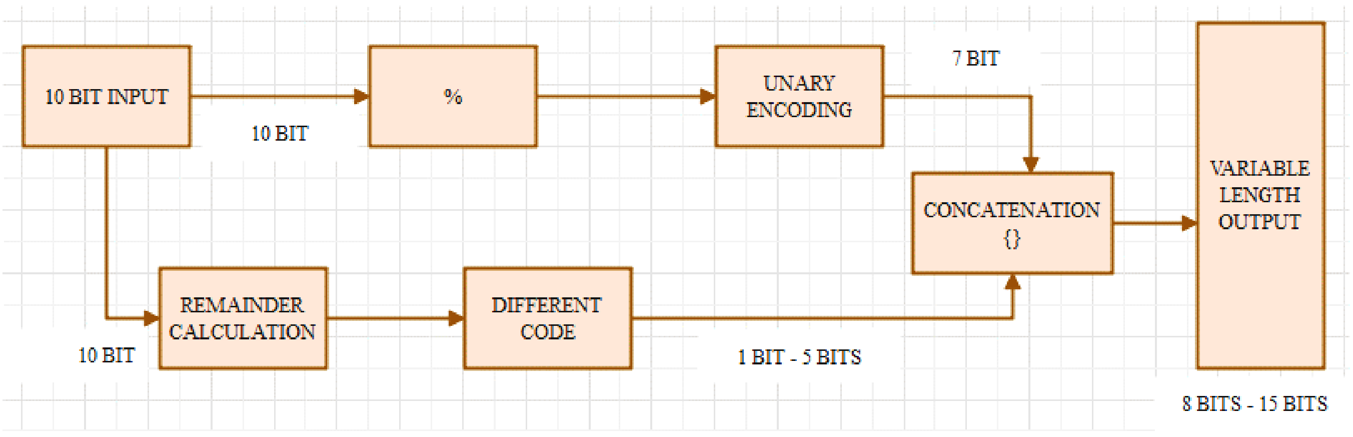

Figure 5. Architecture for (HSCRC)Scheme-1 taking in a 10-bit input

3.3. Low Power Golomb Rice Code (LPCRC) Proposed Scheme-2

Also, we can use the representation according to our needs as the data distribution differs. If the data distribution is linear, then the above method saves most bits. However, if the data distribution is geometric in nature the distribution below is better. The modified encoding scheme is shown in Table 4.

Table 4. LPCRC-Scheme- 2 Encoding

Value of "q"	Unary representation	Further encoding	Bits saved
0	0	0	0
1	10	10	0
2	110	00	1
3	1110	01	2
4	11110	11	3
5	111110	111110	0
6	1111110	1111110	0
7	11111110	11111110	0

As a result, the architecture of the encoding scheme changes and is shown in the figure 6

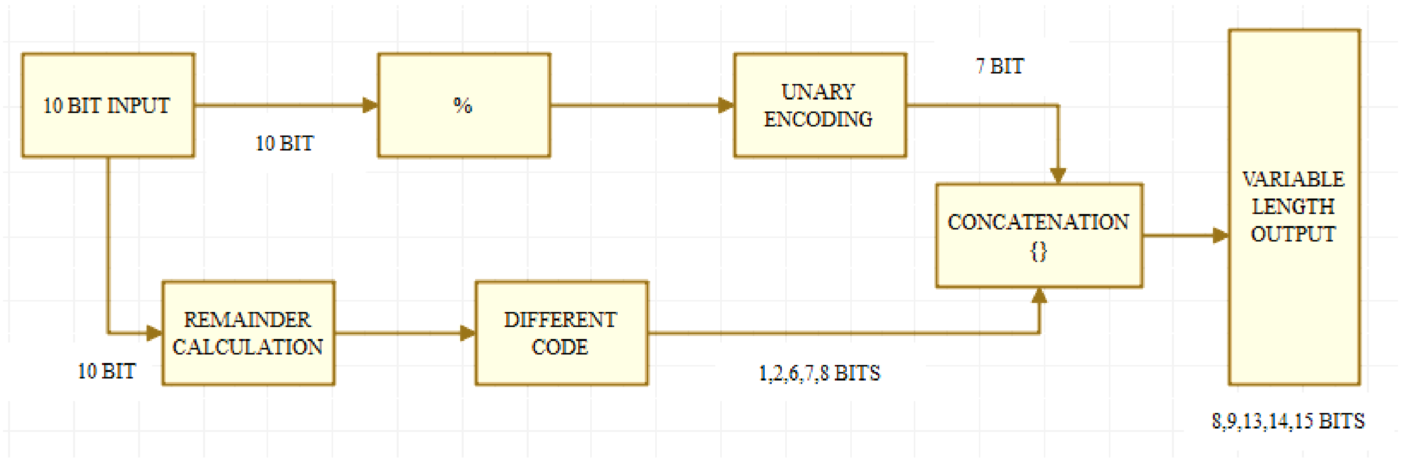

Figure 6. Architecture for LPCRC-Scheme- 2 taking in a 10-bit input

3.4. Efficient bit Reduction Golomb Code (EBRGC)-Proposed Scheme-3

Further, we can make use of different sized registers by encoding the inputs even more judiciously. If the output registers are defined for 3 different bit sizes i.e. 8,9,10 bits, we can effectively use as shown in Table 5.

Table 5. (EBRGC)-Scheme- 3 Encoding

Value of "q"	Unary representation	Scheme-3 encoding	Bits saved
0	0	0	0
1	10	10	0
2	110	00	1
3	1110	01	2
4	11110	11	3
5	111110	000	3
6	1111110	001	4
7	11111110	010	5

As a result, the architecture of the encoding scheme changes and is shown in the figure 7.

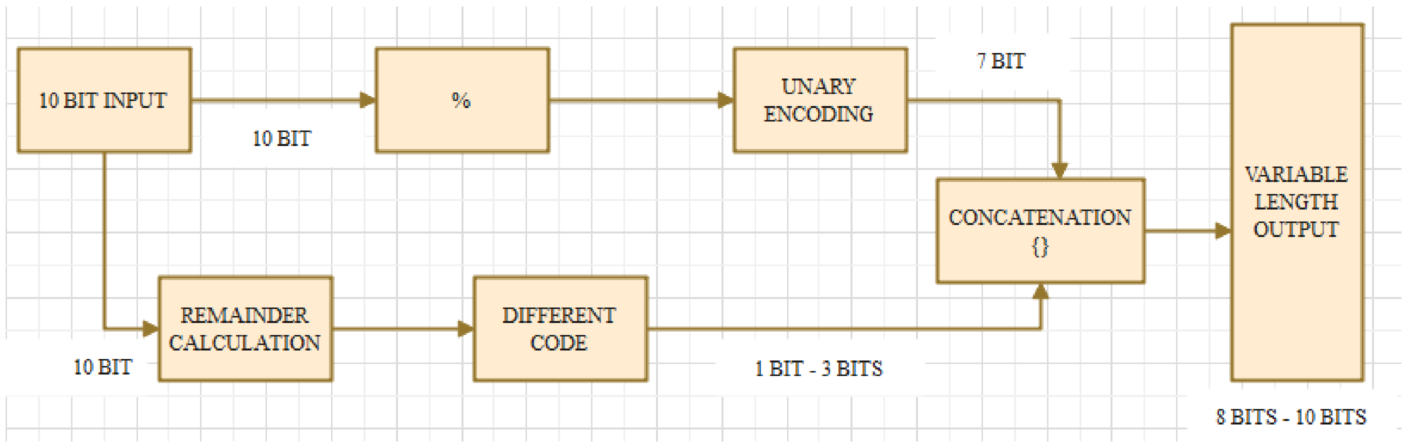

Figure 7. Architecture for (EBRGC)Scheme- 3 taking in a 10 bit input

The output is taken through 5 registers of defined bit length. The registers are of 12,11,10,9,8 bits in size respectively. This makes sure that during decoding, it's possible to decode the value without any error.

4. Performance Analysis

The proposed system is mathematically worked out to verify the validity of the changes which we have made and the same was designed for its hardware architecture. The hardware modelling was done using Verilog and the same was simulated using Modelsim. In order to get better optimization with design metrics like area, power and timing, the proposed design was synthesized using 45nm TSMC design lib with cadence RTL compiler. The synthesized results are compared with the existing architecture in terms of area, power and timing and the same has been shown in table and graphs.

4.1. Bit Reduction

This bit reduction is for linear test input. Total bits used were calculated while keeping in mind that every data has same occurring frequency and comes once. A total of 1024 different inputs are considered to fed into the system one by one. All the other methods use the same inputs but with a different way to encode unary part of the code.Table 6 shows the percentage of bit reduction for the three methods proposed. Figure 10 shows the graphical display of bit usage for the conventional and proposed GR coding techniques.

Table 6. Data showing number of bits used

Method Used	Bits Used	Bits saved	% Reduction
Original GR coding	11776	-	-
HSGRC(Scheme- 1)	9856	1920	16.30 %
LPCGRC(Scheme- 2)	11008	768	6.52 %
EBRGRC(Scheme- 3)	9472	2304	19.56 %

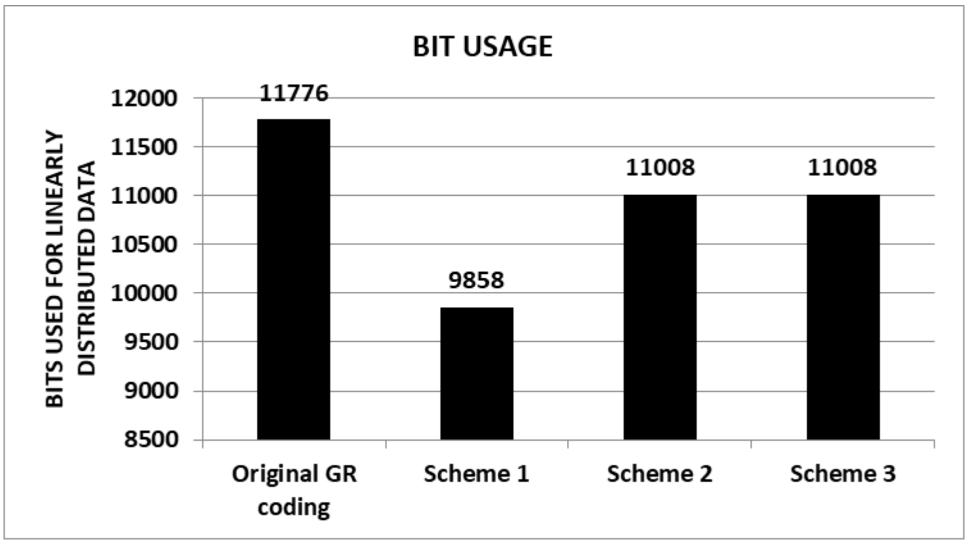

Figure 8. Comparative graph depicting bit usage.

4.2. *Timing Report*

The synthesized report gives the shows the worst case delay for original algorithm and the proposed methods .Table 7 compares the worst case timing for all 3 modified ways of encoding the same input data. Figure.9 shows the graphical representation of the same.

Table 7. Data showing change in worst case delay

Method Used	Worst Case Time Delay(ps)	Time reduction (ps)	% Reduction
Original GR coding	1486	-	-
(HSGRC)Scheme- 1	1260	151	10.16%
(LPCGRC)Scheme- 2	1428	58	3.9%
(EBRGRC)Scheme- 3	1353	133	8.95%

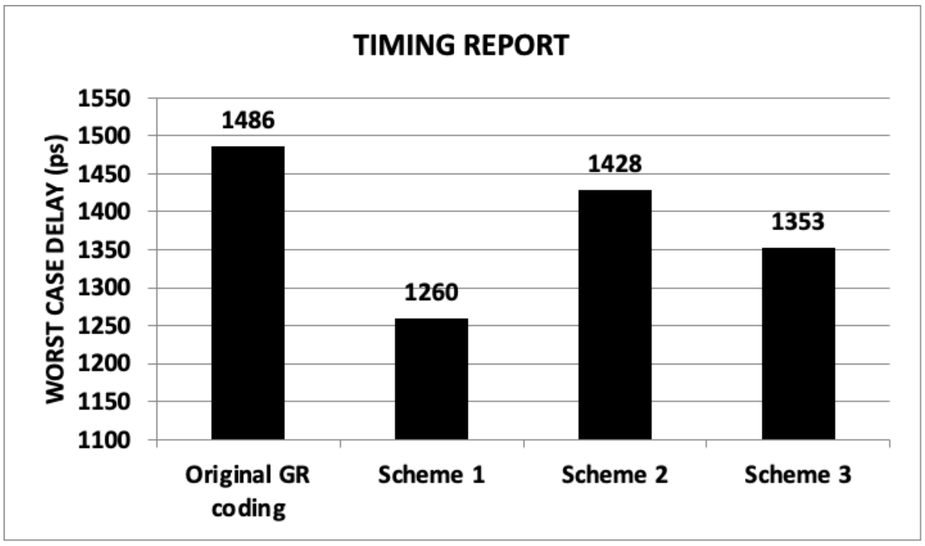

Figure9. Comparative graph depicting worst case delay

4.3. *Area Report*

Area analysis is being made on the area report obtained on the 90nm TSMC synthesis. Table 8 shows the total area used for synthesizing the original algorithm and compares the data for all 3 modified ways of encoding the same input data. Figure 10 shows the graphical representation of the same.

Table 8. Data showing change in area

Method Used	Area(nm)	Area reduction (nm)	% Reduction
Original GR coding	909.2	-	-
HSGRC(Scheme- 1)	707.2	202	22.21%
LPCGRC(Scheme- 2)	689	220.2	24.21%
EBRGRC(Scheme- 3)	577	332.2	36.53%

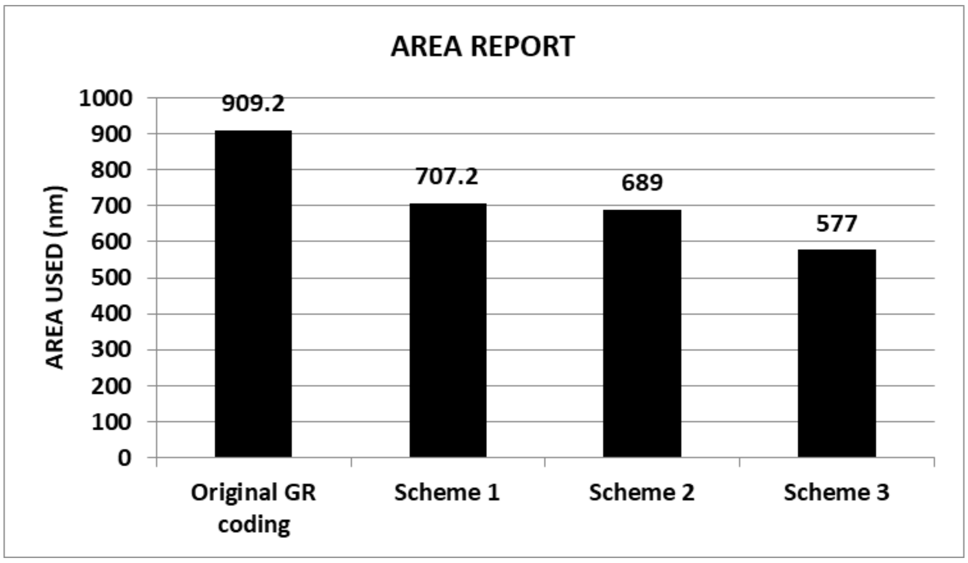

Figure 10. Comparative graph depicting area used

4.4. Power report

Table 9 shows the total power consumed by the proposed circuits made with the original algorithm and compares the data for all 3 modified ways of encoding the same input data. Figure 11 shows the graphical representation of the same. Figure 12 shows a comparative bar graph diagram for the proposed methods.

Table 9. Data showing reduction in power usage

Method Used	Power consumed(nW)	Power saved(nW)	% Reduction
Original GR coding	12904.65	-	-
HSGRC(Scheme- 1)	12054	850.65	6.59%
LPCGRC(Scheme- 2)	11899	1005.65	7.79%
EBRGRC(Scheme- 3)	12838.5	521.15	4.03%

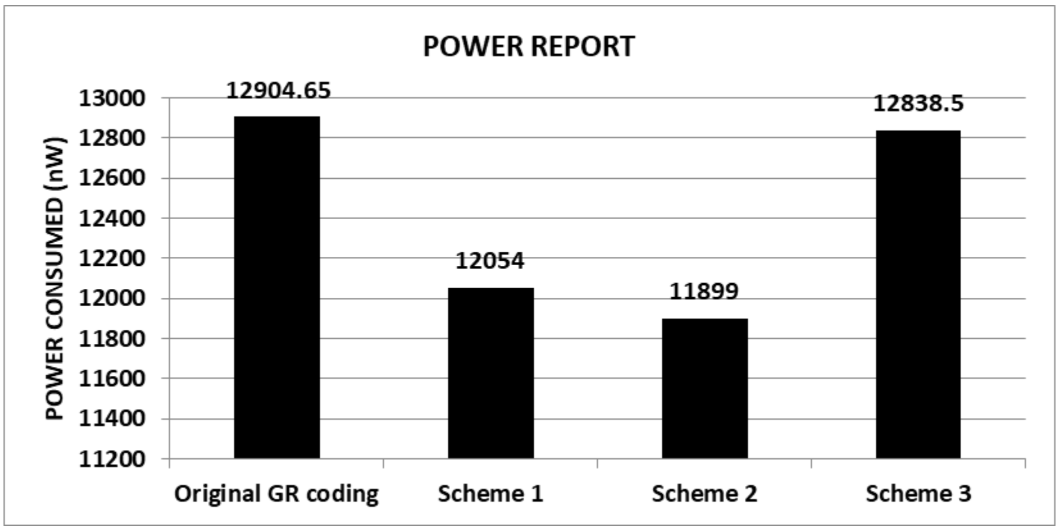

Figure 11. Comparative graph depicting power consumption

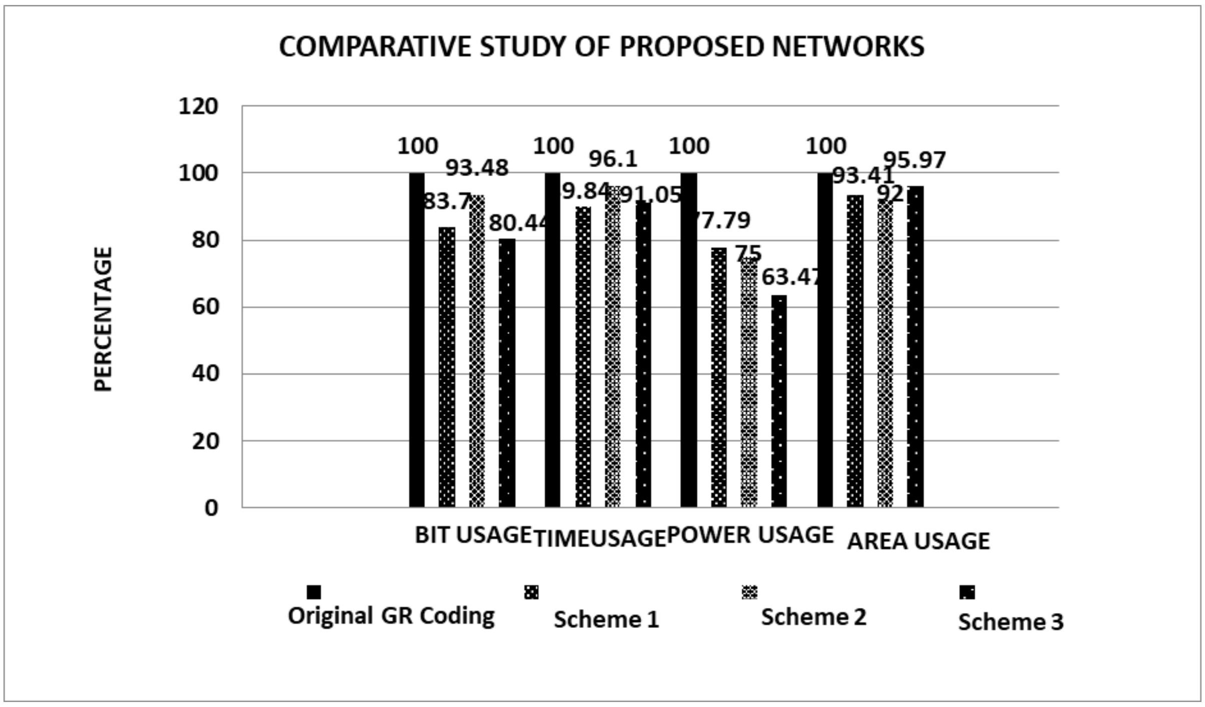

Figure 12. Comparative bar graph depicting difference between the methods.

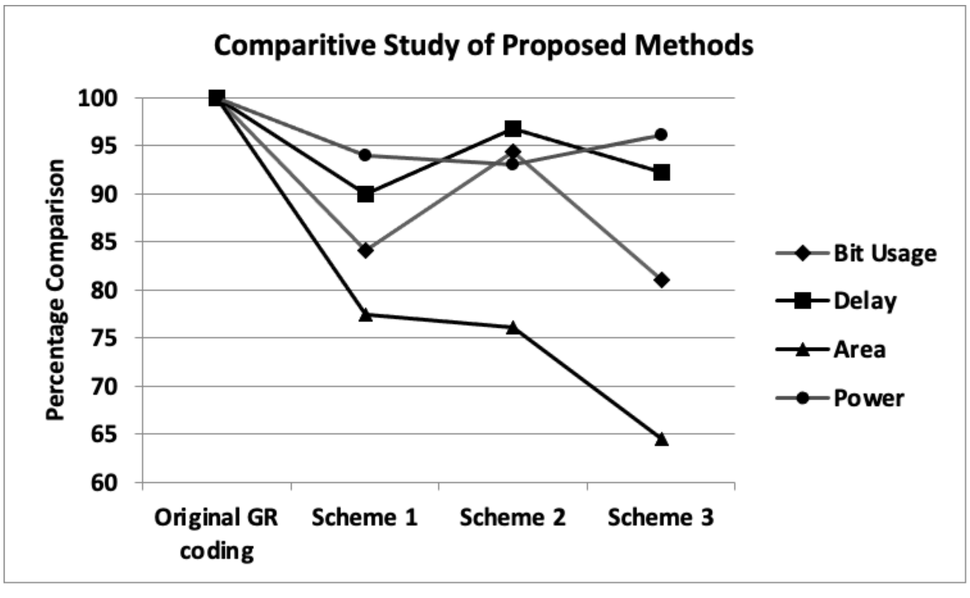

Figure 13. Comparative line graph depicting variations in the proposed methods

Then, we have a different way altogether to show this data where we compare the relative improvement in terms of percentage. Figures 13 and 14 shows a line graph comparison of different methods and metrics used respectively. This is a relative comparison because of which all the readings are compared relatively to each other on ascale of 100 where 100% is the reading for original GR encoding algorithm.

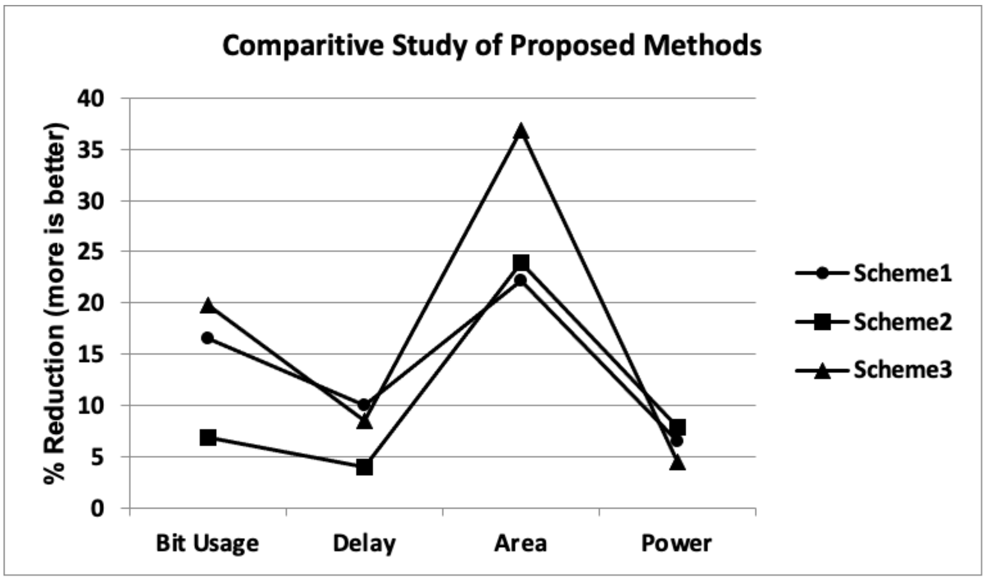

Figure 14. Comparative line graph depicting improvements in proposed parameters

The readings for original GR encoding algorithm is considered as the benchmark and hence it is shown as 100% while others are compared relatively to the 100% value.

To benchmark our results with the existing state of art design we made a comparative analysis with those existing hardware design for data compression. The comparison is made in the aspect of hardware metrics like area, power, frequency of operation with the original GR coding and our proposed designs and also with state of art design with other researchers. Table.10 shows the comparative results.

Table 10. Comparative results with state of art design

Method Used	Area	Power consumed(nW)	Bits used	Speed(MHz)
Original GR coding	909.2 nm^2	12904.65	11776	672
HSGRC(Scheme -1)	707.2 nm^2	12054	9856	793
LPCGRC(Scheme -2)	689 nm^2	11899	11008	700
EBRGRC(Scheme -3)	577 nm^2	12838.5	9472	739
Wei Jhih et al.[33]	6.26mm^2	405900000	*	255
Xiaoke Qin [32]	250 slices	*	*	195
Hang wang et al.[30]	13.5 Logic gates	*	*	600

5. Application and Future scope

The synthesized results of the proposed design shows an appreciable improvement of around 15% reduction in bit size, around 25% reduction in area, 6 % reduction in power consumption and 9% increase in speed. Since there is a reasonable amount of bit reduction it finds a great application in test vector compression and response compaction of VLSI testing. It immensely reduces the testing power because the switching power get reduced due to the bit reduction followed for coding [27 - 28] . The wide application of AI and IoT requires a huge data manipulation and storage which essentially demands an effective hardware architecture for lossless data compression coding and decoding [29-30].

Biomedical applications and neuromorphic computing require huge data computation and lossless compression for analysis, prediction and classification. This majorly demands a dedicated efficient hardware's for data compression [31-32].

This work can be further extended by applying signal statistics based modified GR codes architecture for Machine learning algorithms and this could also be optimized based on stochastic computing based architecture which could be applied in robotics and brain computing.

6. Conclusion

In this work we try to come up with three different algorithmic level modification in GR coding and its corresponding hardware architecture was designed, simulated and synthesized. The synthesized results were compared which shows an effective increase of metrics compared with GR compression algorithm. These modifications offered some significant improvements. These improvements have been quantified and summed up below:

HSGRC (Scheme -1) displays 16.30 % reduction in bit usage for linearly spread data. Worst-case delay has been reduced by 10.16%, while area has been reduced by 22.21%. In addition, power consumption is down by 6.59%.

LPCGRC (Scheme -2) displays 6.52 % reduction in bit usage for linearly spread data. Worst-case delay has been reduced by 3.9%, while area has been reduced by 24.21%. In addition, power consumption is down by 7.79%.

EBRGRC (Scheme -3) displays 19.56 % reduction in bit usage for linearly spread data. Worst-case delay has been reduced by 8.95%, while area has been reduced by 36.53%. In addition, power consumption is down by 4.03%.

Based on the comparison it suggest for the usage in test vector compression, biomedical image compression and in AI with IoT applications.

Author Contributions: Conceptualization, G.N, T.S, S.R, V.M.; data curation, V.M.; formal analysis, H.K, S.N.; funding acquisition, S.N, H.K.; investigation, S.R.; methodology, G.N.; project administration, V.M.; resources, V.M.; software, G.N.; supervision, S.N, H.K.; validation, S.N. and A.F.; visualization, H.K.; writing–original draft, G.N.; writing–review and editing, G.N, S.R, V.M. All authors have read and agreed to the published version of the manuscript.

Funding: Please add: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest to declare.

References

1. Howard PG, Vitter JS. Fast and efficient lossless image compression. In[Proceedings] DCC93: Data Compression Conference 1993 Mar 30 (pp. 351-360). IEEE.
2. Starosolski R, Skarbek W. Modified Golomb-Rice codes for lossless compression of medical images. InProceedings of International Conference on E-health in Common Europe, Cracow, Poland 2003 Jun (pp. 423-37).
3. Malik A, Kumar R. Compression Techniques for ECG Signal: A Review. Int. J. Modern Electron. Commun. Eng.. 2016;4(4):1-4.
4. Singh B, Kaur A, Singh J. A review of ECG data compression techniques. International journal of computer applications. 2015 Jan 1;116(11).
5. Batista LV, Carvalho LC, Melcher EU. Compression of ECG signals based on optimum quantization of discrete cosine transform coefficients and Golomb-Rice coding. InProceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439) 2003 Sep 17 (Vol. 3, pp. 2647-2650). IEEE.
6. Brito M, Henriques J, Carvalho P, Ribeiro B, Antunes M. An ECG compression approach based on a segment dictionary and bezier approximations. In2007 15th European Signal Processing Conference 2007 Sep 3 (pp. 2504-2508). IEEE.

7. Marisa T, Niederhauser T, Haeberlin A, Goette J, Jacomet M, Vogel R. Asynchronous ECG time sampling: Saving bits with Golomb-Rice encoding. In2012 Computing in Cardiology 2012 Sep 9 (pp. 61-64). IEEE.
8. Tsai TH, Kuo WT. An efficient ECG lossless compression system for embedded platforms with telemedicine applications. IEEE Access. 2018 Jul 23;6:42207-15.
9. Chan HL, Chiu YC, Kao YA, Wang CL. VLSI implementation of wavelet-based electrocardiogram compression and decompression. Journal of Medical and Biological Engineering. 2011 Oct 1;31(5):331-8.
10. Bellani NH, Ghutke P. A Modified GOLUMB Encoder and Decoder for Test Vector Compression.
11. Balakrishnan KJ, Touba NA. Improving linear test data compression. IEEE transactions on very large scale integration (VLSI) systems. 2006 Dec 4;14(11):1227-37.
12. Kalode P, Khandelwal R. Test data compression based on Golomb coding and two-value Golomb coding. Signal & Image Processing. 2012 Apr 1;3(2):171.
13. Karthik B, Kumar TV, Selvaraj A. Test data compression architecture for low power VLSI testing. World Applied Sciences Journal. 2014 Jan;29(8):1035-8.
14. Volkerink EH, Khoche A, Mitra S. Packet-based input test data compression techniques. In Proceedings. International Test Conference 2002 Oct 10 (pp. 154-163). IEEE.
15. Sreejeesh SG, J. U Kidav, Sakthivel R. Beam forming Algorithm Architectures for Medical Ultrasound. International Journal of Innovative Technology and Exploring Engineering 2019 Oct ; 8(12):2452-2459.
16. Krishna TS, Riyas KS, Premson Y, Sakthivel R. 15–4 Approximate Compressor based multiplier for image processing. In2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI) 2018 May 11 (pp. 671-675). IEEE.
17. Şimşek C, Kaya İ, Albayrak C. Low complexity, losless ECG data compression algorthims for Wireless Sensor Network. In2013 21st Signal Processing and Communications Applications Conference (SIU) 2013 Apr 24 (pp. 1-4). IEEE.
18. Joseph B, Acharyya A, Rajalakshmi P. A low complexity on-chip ECG data compression methodology targeting remote health-care applications. In2014 36th annual international conference of the IEEE engineering in medicine and biology society 2014 Aug 26 (pp. 5944-5947). IEEE.
19. Karmakar R, Chattopadhyay S. Thermal-aware test data compression using dictionary based oding. In2015 28th International Conference on VLSI Design 2015 Jan 3 (pp. 53-58). IEEE.
20. Rzepka D. Low-complexity lossless multichannel ECG compression based on selective linear prediction. Biomedical Signal Processing and Control. 2020 Mar 1;57:101705.
21. Wang H, Wang T, Liu L, Sun H, Zheng N. Efficient compression-based line buffer design for image/video processing circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2019 Jun 26;27(10):2423-33.
22. Wang S, Wei W, Wang Z. A Low Overhead High Test Compression Technique Using Pattern Clustering With $ n $-Detection Test Support. IEEE transactions on very large scale integration (VLSI) systems. 2009 Nov 10;18(12):1672-85.
23. Qin X, Muthry C, Mishra P. Decoding-aware compression of FPGA bitstreams. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2009 Dec 8;19(3):411-9.
24. Wang WJ, Lin CH. Code compression for embedded systems using separated dictionaries. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2015 Feb 4;24(1):266-75.
25. Lekatsas H, Henkel J, Wolf W. Approximate arithmetic coding for bus transition reduction in low power designs. IEEE transactions on very large scale integration (VLSI) systems. 2005 Jun 27;13(6):696-707.
26. Golomb and Rice coding Juan Francisco Rodríguez Herrera Vicente González Ruiz {https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Text/03-symbol_encoding/09-Golomb_coding/index.html}
27. Golomb-Rice Coding {https://urchin.earth.li/~twic/Golomb-Rice_Coding.html}.
28. Data Compression https://searchstorage.techtarget.com/definition/compression

29. Granberg T. Handbook of digital techniques for high-speed design. Pearson India; 2004.
30. Khalgui M, Hanisch HM, editors. Reconfigurable Embedded Control Systems: Applications for Flexibility and Agility: Applications for Flexibility and Agility. IGI Global; 2010 Nov 30.
31. Akhter S. Digital Hardware Design. Laxmi Publications, Ltd.; 2008.
32. Domnic S. A new method for Golomb-Rice parameter estimation. In2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) 2017 Nov 13 (pp. 1-5). IEEE.