Improved Compression Scheme for FPGA with Hamming Code Error Correction Technique

Chinchu C
PG Scholar
Applied Electronics & Instrumentation
Younus College of Engineering & Technology
Kollam, India

Riyas A N
Assistant Professor
Electronics & Communication
Younus College of Engineering & Technology
Kollam, India

***Abstract*—Theexcess of area taken by the nonvolatile registers can be reduced by the compression strategy. Compression strategy is based on run length encoding which is lossless. Errorslike single bit, multiple bit and bursterrors can occur in the memory storage unit changing the compressed data. These errors are corrected by hamming code, which is a family of linear error correcting codes that can detect up to two bit errors or correct one bit errors. Compression scheme for FPGA is being proposed to be stimulated in Xilinx 14.5.**

***Keywords*—Run length encoding; hamming code; Xilinx; FPGA**

I. INTRODUCTION

Data are stored in either the volatile storage units as in the case of conventional processor, like charge stored in a capacitor, flip flops, registers, and static random access memories (SRAM). The states are lost when the power supply is turned off, because the charge on the capacitor quickly drains without the power supply. Some traditional processors use nonvolatile memories to store the data.Some exascalerandom access memory use off- chip phase change random access memory (PCRAM) for checkpointing applications [2]. PCRAM is a nonvolatile random access memory that has the ability to achieve anumber of distinct intermediary states, thereby having the ability to hold multiple bits in a single cell. Self powered embedded systems use on-chip ferroelectric random access memory (FeRAM) to prevent data loss during power failure [3]. FeRAM is similar in construction to DRAM but uses a ferroelectric layer instead of dielectric layer to achieve non volatility. It offers the same functionality as flash memory.

Nonvolatile processors store data in nonvolatile storage unit. It has got several advantageslike zero standby power, instant on and off feature, high resilience to power failure, fine grain power management. To implement a NV processor, appropriate NV memory technologies are needed for flip-flops and registers. Flash memory is a high density nonvolatile memory but it has got disadvantages like low endurance, slow writing speed, block erasing pattern, high mask cost. Another nonvolatile memory is PCRAM which have disadvantages like asymmetric reading/writing characteristics, limited life time.FeRAM and magnetic random access memory (MRAM) have advantages like unlimited operation cycles, ultrashort access time, easy integration to CMOS technology.

The challenge faced by all the integrated circuits is the area occupied by the memory units such as registers, flip-flops and other storage units. The area of the storage unit can be reduced by the compression technique. We make the following contributions.

1) We develop a design to reduce the number of bits to be stored in the memory units thereby reducing the number and area of the storage units. The bits are reduced by compression.
2) The compression codec is based on parallel run length encoding /decoding scheme. It is a two stage shifting network designed to minimize the codec area.
3) Hamming code is used for error correction.

II. OVERVIEW

A. *Impact of area*

In the case of a nonvolatile flip-flop the area taken by the nonvolatile storage unit is large [1]. The Table I below shows the area taken up by the non volatile storage units.Table shows that the area overhead in different nonvolatile technologies are all nontrivial. It should be noted that the area overhead does not consider the variation and reliability issues, which may further increase the chip area. For ferroelectric processor, the ferroelectric capacitors have a sandwich structure with a ferroelectric film layer between two metal layers. To reduce read and write operations to a low level,the ferroelectric film should be large enough.

TABLE I. AREA CHALLENGE UNDER DIFFERENT NOVOLATILE APPROACHES

Approach	Area of NVFF in DFFs	Area overhead of entire chip
Floating gate	1.4x	19%
Magnetic RAM	2x	40%
FeRAM	4-5x	90%

III. COMPRESSION ALGORITHM

A. *Run length encoding*

The architecture comprises of an encoding and decoding scheme [1]. The data to be stored in the memory unit is first compressed using parallel run length encoding. Then hamming code is used for error correction since single bit, multiple bit and burst errors can occur and change the compressed data. Then comes the decoding stage where the original data is retrieved without any error. The compression scheme used is parallel run length encoding.

First, a compression algorithm with superior compression ratio should be designed and its hardware feasibility and processing speed should be taken into consideration. Second, the hardware design of the codec should be implemented. In the hardware design, the area overhead must be reduced to improve the performance.

A proper compressionalgorithm should have the following features: 1) it should be lossless, that is, at the decoding stage the data must be recovered without any error. 2) it should provide efficient hardware implementation, that is, the area overhead introduced bythe hardware realization of the algorithm should be as small aspossible. One widely used lossless encoding algorithm is

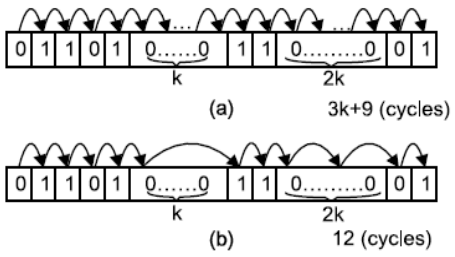

Fig. 1. Example of *k* bit parallel RLE. (a) Serial RLE (b) Parallel RLE

Huffman coding [4], which is a lossless encoding algorithm that provides a unique decodable entropy code. However it cannot be used, as it requires a predefined probability distribution of each symbol. In this case it is unobtainable. All the algorithms are software based so it requires an additional hardware to store code. So these are not used for compression. Some efficient binary compressionalgorithms are used in memory data compression, such asthe LZRW series, which achieve good compressionspeed and ratio. However, those algorithms are software-basedand require significant additional hardware to store code the dictionary.

Run length encoding (RLE) is a lossless algorithm and can be easily implementedin hardware. Unlike Huffman coding it does not require a probability distribution function and code mapping. Furthermore, RLE offers a good compression ratio for consecutive 0/1 sequences. Therefore, we choose RLE encoding as thecompression algorithm.

RLE is of two types, serial and parallel. In this paper parallel RLE is used. The shortcoming of the traditional RLE is that it processesbit streams serially, which incurs a large time overhead.Trein*et al*. [5] proposed a parallel input-based RLE. Instead ofexamining a bit stream bit by bit, our parallel RLE (PRLE)observes *k* bits in parallel. If they are all 0 or 1, all the *k* bitsare bypassed in one clock cycle. Fig. 1shows that for serial RLE it requires 3k+9 cycles to bypass the bit stream but for parallel RLE it takes only 12 clock cycles. K bits are called observation width window (OWW) for parallel RLE. However the value of k affects the speed.

Disadvantage of RLE is that, it cannot operate well when there are short0/1's chain. We can use adaptive codec to improve the efficiency of short chains. The challenges faced by the adaptive codec are:1) the improvement that can be done on adaptive codec is limited because of the infrequent appearance of 0/1's chain.2) adaptive codec requires complex circuits which requires large area.In order to improve the efficiency of PRLE, we use a threshold value of length, only if the bit chains of 0/1's exceed that length PRLE will be performed.

B. *Hardware design*

Fig. 2 shows the hardware design of the codec. It has an input end shifting network which shifts the 'n' input bits from the registers to encoder. Output end shifting network shift 'm' bit to registers. There is also a 0/1 detector, which is for the observation of 'k' bits. The detector generates a bypass signal to RLE encoder and length controller. The function of the length controller is that it provides the length that has to be shifted, depending on 'OWW' and bypass signal. RLE encoder compresses the 'k' bit serially when the bypass signal is enabled. If it is disabled it bypasses the 'k' bit. Microcontroller control unit (MCU) provides OWW 'k' and threshold value. After compressing the output is 'q' bit which is given to the output end shifting network.

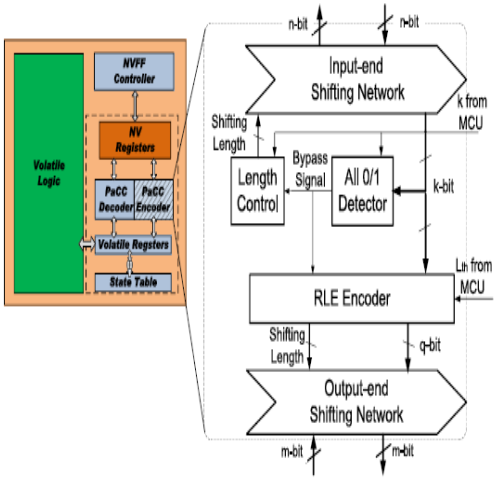

Fig. 2. Block diagram for the encoder in the architecture

In the decoder stage, the input and output shifting networks are interchanged. Instead of RLE encoder, *there* is an RLE decoder. Data flows in the opposite direction. Encoding and decoding are opposite operations, hence the decoder can reuse the two shifting networks. The only difference in the decoder is that it contains anRLE

decoding module instead of the RLE encoding module. Thus we omit detailed discussion on the decoder part.

The two shifting is a challenge because the shifting length changes. Barrel shifter can be used but it consumes large area. In order to overcome this we use hierarchical shifting network. It is a new area efficient shifting structure.

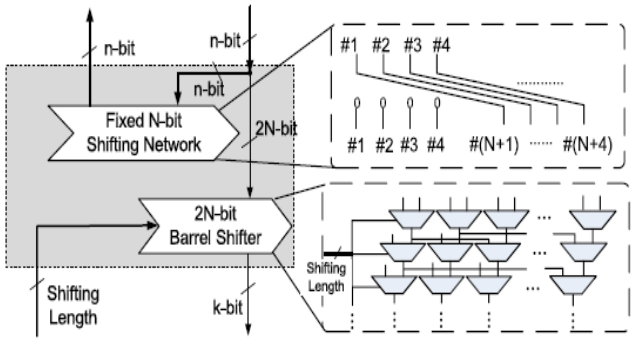

Fig. 3. Structure of the two stage shifting network

Fig. 3 shows a two stage hierarchical shifting network. First stage consists of a coarse grained shifting network. It has a fixed shifting length 'N'. Input is shifted by 'N'. Advantage of the first stage is that it is area efficient and no multiplexer is required. Second stage consist of a '2N' bits barrel shifter which takes first '2N' bits of the 'n' bit input. Second stage shift the data to be compressed. First stage update the volatile registers. There are $2N \times \log_2(2N)$ multiplexers of which '2N' is used to increase the performance. If the value of 'N' is small, then area becomes small.

C. Threshold Based RLE Algorithm

Threshold based RLE algorithm is used to record the length of 0/1's chain [7]. It provides a good compression ratio. Algorithm 1 represents threshold based variable run length encoding. The input of the algorithm is a data bit stream$v_{diff}$and a threshold value 'm', the output is the compressed data streamv_{comp}. Threshold value 'm' is selected to deal with short 0/1's chains.

Line 1 of the algorithm checks if the encoding is complete or not by comparing vector current index *k* with the vector bound *n*. Line 2 determines if a transition (0 *to* 1 or 1 *to*0) happens in the sequence. If the number of consecutive zeros or ones are smaller than *m*,*Process_ShortSEQ*is called to copy the segment intov_{comp}. Otherwise, we call *Process_LongSEQ*to encode the segment intov_{comp}. The encoded segments and copied segments are regularly constructed according to Fig. 4.

Fig. 4. Shows encoded and copied segment structure. The first bit (flag bit) in the structure indicates the segment category. The next four bits indicate the length of the body part. The body part records the length of consecutive 0/1 for the encoded segments or the copied segments. For a good compression, the threshold value 'm' must be properly chosen. In real applications the threshold value 'm' varies for different input bit stream. The value of *m* is set to balance the length of the encoded segments and the copied segments.

Algorithm 1

Input: V_{diff}, *m*
Output::V_{comp}
1: while *k* < *n* do
2: if thena$_k$ = a$_{k+1}$
3: if *counter* ≤ *m* then
4: *Process_ShortSEQ( );*
5: end if
6: if *counter* > *m* then
7: *Process_LongSEQ( );*
8: end if
9: end if
10: end while

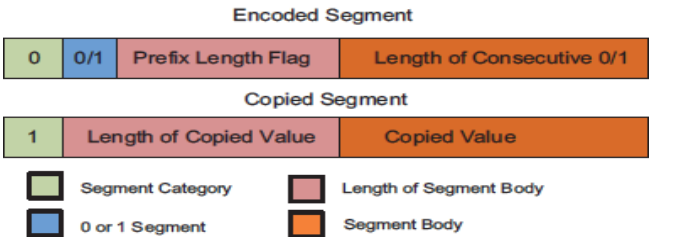

Fig. 4. Encoded and copied segment structure

D. Hamming code

Hamming codes are linear error correcting codes that can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors. Hamming codes achieve highest possible rate for codes, with block length and minimum distance of three.

Hamming codes are binary linear codes. For each integer*r* ≥ 2 there is a code with block length n=2r-1 and message length k=2r-r-1. Rate of Hamming code is R=k/n=1-r/(2r-1), which is the highest possible for codes with minimum distance of three and block length 2r-1. Simple parity code cannot correct errors, and can detect only odd number of bits in error.

Parity- check matrix of Hamming code is constructed by listing all columns of length 'r' that are non –zero, which means the dual code of the Hamming code is the Punctured Hadamard code. In parity-check matrix any two columns are pair wise linearly independent. Hamming codes add only limited redundancy to the data so they can only detect and correct errors when the error rate is low. Extended hamming codes achieve a hamming distance of four so they are said to be single-error correcting and double error detecting codes (SECDED).

Parity adds a single bit indicating whether the number of one's (bit- positions with values of one) in the preceding data was even or odd. The error can be detected if odd number of bits are changed in transmission, the message will change parity. However the bit that changed may have been the parity itself. If the parity is one, it indicates that there are odd numbers of one in the data, and a parity value of zero indicates that there are even numbers of one in the data. Error will not be detected if the number of bits

changed is even and check bit will be valid. Parity can only detect the error, it cannot indicate which bit contained the error. Hence, the data must be discarded entirely and re-transmitted.

If more error-correcting bits are included with a message,and if those bits can be arranged such that different incorrectbits produce different error results, then bad bitscould be identified. In a seven-bit message, there areseven possible single bit errors, so three error control bitscould potentially specify not only that an error occurredbut also which bit caused the error.

Hamming studied the existing coding schemes and developed a nomenclature to describe a system, including the number of data bits and error correction bits in a block. Hamming described this as an (8,7) code , with eight bits in total, of which seven are data. The repetition example will be (3,7), following the same logic. For repetition example 1/3 , codec rate is the second number divided by the first.

Hamming studied the existing coding schemes, includingtwo-of-five, and generalized their concepts. To start with,he developed a nomenclature to describe the system, includingthe number of data bits and error-correction bitsin a block. For instance, parity includes a single bit forany data word, so assuming ASCII words with seven bits.Parity has adistance of 2, so one bit flip can be detected, but notcorrected and any two bit flips will be invisible. The(3,1) repetition has a distance of 3, as three bits needto be flipped in the same triple to obtain another codeword with no visible errors. It can correct one-bit errorsor detect, but not correct two-bit errors. A (4,1) repetition(each bit is repeated four times) has a distance of 4,so flipping three bits can be detected, but not corrected.When three bits flip in the same group there can be situationswhere attempting to correct will produce the wrongcode word. In general, a code with distance k can detectbut not correct $k - 1$ errors.

E. Algorithm of hamming code

The following general algorithm generates a single-errorcorrecting (SEC) code for any number of bits.

1. Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.

2. Write the bit numbers in binary: 1, 10, 11, 100, 101,etc.

3. All bit positions that are powers of two (have onlyone 1 bit in the binary form of their position) areparity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)

4. All other bit positions, with two or more 1 bits in thebinary form of their position, are data bits.

5. Each data bit is included in a unique set of 2 or moreparity bits, as determined by the binary form of itsbit position.

(a) Parity bit 1 covers all bit positions which havethe least significant bit set: bit 1 (the parity bititself), 3, 5, 7, 9, etc.

(b) Parity bit 2 covers all bit positions which havethe second least significant bit set: bit 2 (theparity bit itself), 3, 6, 7, 10, 11, etc.

(c) Parity bit 4 covers all bit positions which havethe third least significant bit set: bits 4–7, 12–15, 20–23, etc.

(d) Parity bit 8 covers all bit positions which havethe fourth least significant bit set: bits 8–15,24–31, 40–47, etc.

(e) In general each parity bit covers all bits wherethe bitwise AND of the parity position and thebit position is non-zero.

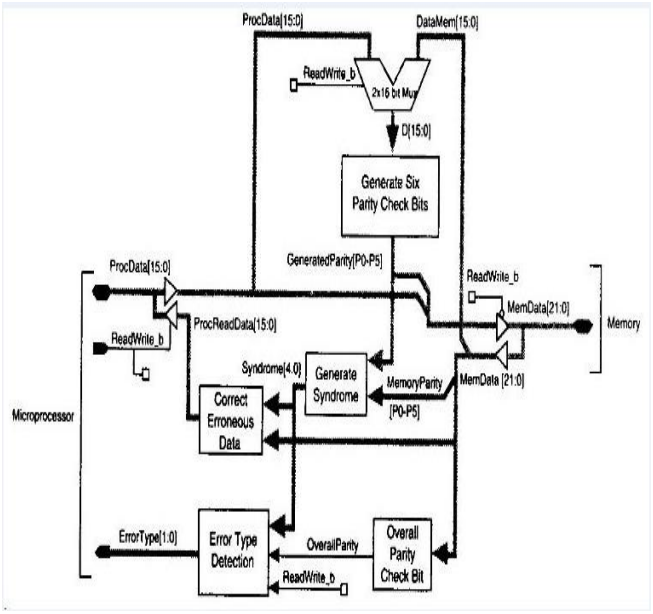

Fig. 5. Block diagram of hamming code

IV. SIMULATION RESULTS

The compression scheme for FPGA with hamming code error correction technique can be designed in Xilinx 14.5 software. By using this software we can generate thewaveforms of the compressed data at the encoding stage and decompressed data at the decoding stage.

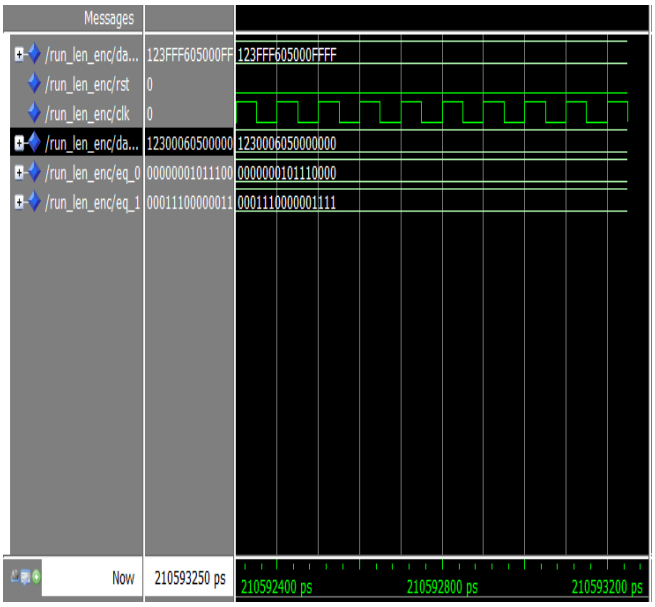

Fig. 6.Output at the encoding stage

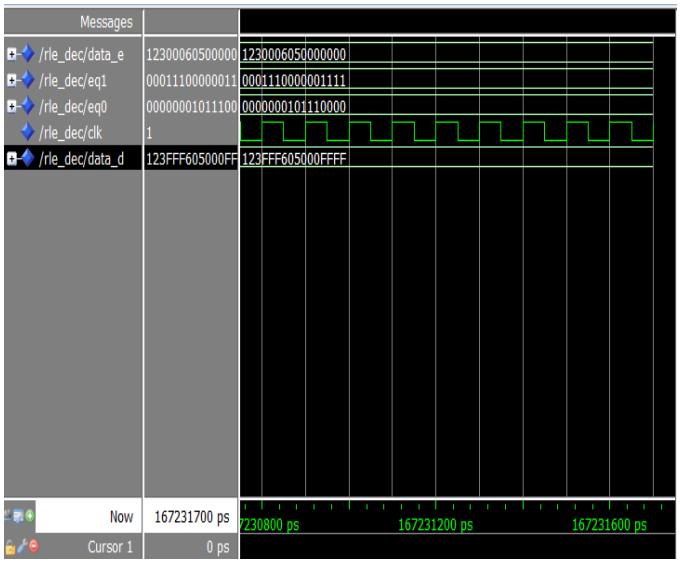

Fig. 7. Output at the decoding stage

V. CONCLUSION

Compression technique using hamming code opens a new domain for area efficient, power saving and other attractive applications. This paper proposes a compression strategy for reducing energy and the number of bits stored in a memory unit. The compressed data is prone to errors like single bit, multiple bit and burst errors which is corrected by hamming code. Our future work will focus on data compressing of more than 64 bits.

ACKNOWLEDGMENT

I would like to thank, the Head ofthe Department of Electronics and Communication Mr. Rajeev S Kwho were always ready to help me with ideas and suggestions for rectifying themistakes that crept up time to time during the completion of this venture. I would also like to thank myfriends and last but not the least the staff of ECE department for their whole hearted support andencouragement. Above all, I am thankful to the god almighty.

REFERENCES

[1] PaCC: A Parallel Compare and Compress Codec for Area Reduction in Nonvolatile Processors YiqunWang, *Student Member, IEEE, Yongpan Liu, Member, IEEE,Shuangchen Li, Xiao Sheng*, Daming Zhang, Mei-Fang Chiang, BaikoSai, Xiaobo Sharon Hu, *Senior Member, IEEE,*andHuazhong Yang, *Senior Member, IEEE*, july 2014.

[2] W. M. Jones, J. T. Daly, and N. DeBardeleben, "Application monitoring and checkpointing in HPC: Looking towards exascale systems," in *Proc. 50th Annu. Southeast Regional Conf.*, 2012, pp. 262–267.

[3] M. Zwerg, A. Baumann, R. Kuhn, M. Arnold, R. Nerlich, M. Herzog, R. Ledwa, C. Sichert, V. Rzehak, P. Thanigai, and B. Eversmann, "An 82 μA/MHz microcontroller with embedded FeRAM for energy harvesting applications," in *Proc. ISSCC*, Feb. 2011, pp. 334–336.

[4] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952

[5] J. Trein, A. Schwarzbacher, B. Hoppe, and K. Noff, "A hardware implementation of a run length encoding compression algorithm with parallel inputs," in *Proc. ISSC*, Jun. 2008, pp. 337–342.

[6] G. Beenker and K. Immink, "A generalized method for encoding and decoding run-length-limited binary sequences (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 29, no. 5, pp. 751–754, Sep. 1983.

[7] Y. Wang, Y. Liu, Y. Liu, D. Zhang, S. Li, B. Sai, M.-F. Chiang, and H. Yang, "A compression-based area-efficient recovery architecture for nonvolatile processors," in *Proc. DATE*, Mar. 2012, pp. 1519–1524.