International Journal For Technological Research In Engineering
Volume 4, Issue 7, March-2017 ISSN (Online): 2347 - 4718

HARDWARE IMPLEMENTATION OF ENTROPY ENCODER USING Ψ1,K ENCODING SCHEME ON RECONFIGURABLE LOGIC

Priyanka Pawar[1], Nitesh Dodkey[2], Siddharth Singh Parihar[3]
[1]M.Tech Scholar, [2]HOD ECE, [3]Assistant Professor
Department of Electronics and Communication Engineering, Surabhi Group of Institutions, Bhopal (M.P.) India

Abstract: In this paper, hardware implemented a entropy encoder using Rice PSI 1,k encoding scheme. The design can calculate the value of k in real time and then selects the number of pass through bits. The adder used in this design is serial adder, this reduces the resource usage and also reduces the power consumption of the design. The target device to implement the design is Virtex 5 FPGA (XC5VLX20T-2FF232). Xilinx XST is used to synthesize the design and it is coded in VHDL. The synthesis report shows that 2% of slice registers, 5% of Slice LUTs and 8% of slices were occupied by this design, this suggests area requirement.

Keywords: Rice algorithm, $PSI_{1,k}$, Entropy Encoding, FPGA

I. INTRODUCTION

Real-time lossless compression hardware will be a part of next generation computing chips (very large scale integrated circuits, VLSI) to sustain higher data rates over on-board limited channel and storage capacities. They include processor, network, and storage chips. With a reduced number of bits, limited capacity of transmission channels or storage can be used effectively. Such a reduction has a direct impact on complexity reductions, cost reductions, as well as overall system reliability improvements [1]. Image communication and storage are examples of applications that benefit from image compression, because the compression results in (i) faster (real–time) image transmission through band limited channels and (ii) lower requirements for storage space. Non compressed images require many data bits, making it impossible for real–time transmission through band limited channels—such as 48 kbit per second (kpbs) and 112 kbps integrated services digital network (ISDN), as well as 9.6 kbps voice–grade telephone or radio channels [1]. For example, transmitting a 256×256 pixels grayscale still–image over the voice–grade line would require at least 54.61 s. Furthermore, one HDTV format needs 60 frames of 1280×720 pixels per second. Using 24 bpp colour pixels, this HDTV format would require an impractical channel capacity of 1,440 Mbps. For example, space explorations by National Aeronautics and Space Administration (NASA) missions generate huge science data, requiring real-time compression [2]. Consequently, international bodies such as Committee for Space Data Systems (CCDS) have defined compression standards for real-time systems [3].

II. COUNTER CODE ALGORITHM

To design the architecture, we must first understand the counter code algorithm and then use it as a design requirement. A basic counter coder called Ψ_1 (or sometimes also called PSI–1) works as follows [2]:
Given a block of data samples (for j = 1, ..., J), coder Ψ_1 assumes that each sample takes a symbol si , for i = 1, ..., 256, as shown in Table 1. Coder Ψ_1 treats each sample symbol Si having sample data di as a non-negative integer number xi . The average length of sample data is R = 8 bits per sample.

For every sample in the block (having a symbol si , thus having sample data di), a Ψ_1 encoder converts di into a codeword wi of a length li = Xi + 1, consisting of xi consecutive zero bits '0' followed by a closing one bit '1' (see again Table 1). For example, if a sample happens to be di = 0000 0011, it must have xi = 3, and the Ψ_1 then encodes it using 4 bits, i.e., 3 zeros followed by a one. Hence, the encoding algorithm (converting di into wi) is simply down counting, which is summarized in Table 1.

The reconstruction is obviously simple counting too. Given a codeword wi , a Ψ_1 decoder just counts the number of zero bits until a one appears. The counting result is the sample value xi . Using Table 1, it determines that the codeword belongs to a symbol si , hence it produces the sample data di as its output.
We then propose to use the counter coder as an alternative and more computationally efficient entropy coder. One basic counter coder called Ψ_1 (or PSI–1) works as follows [2]. Given a block of data samples (for j = 1, ..., J), Ψ_1 replaces every sample in the block with a number of consecutive zero bits '0' followed by a closing one bit '1' (see Table 1). For example, if a sample happens to have a value of 55, the Ψ_1 encodes it using 56 bits, i.e., 55 zeros followed by a one. Hence, the encoding algorithm is simple. The reconstruction is obviously simple too. A Ψ_1 decoder just counts the number of zero bits until a one appears. The counting result is the sample value.
Define now a function $i(j)$ such that a particular data block consists of symbols $\{S_{i(1)}, S_{i(2), \ldots\ldots,} S_{i(j)}\}$. The total number of bits required to encode a data block is (for index j and block size J)

$$L = J + \sum_{j=1}^{j} X_{i(j)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Notice that this code basically allocates more codeword bits to data samples with higher sample values, i.e. $X_i \geq X_j \leftrightarrow l_i \geq l_j$. This means Ψ_1 code is optimal if the statistical distribution of data samples is monotonically decreasing, i.e., $X_i \geq X_j \leftrightarrow P(Si) \geq P(Sj)$. The average codeword length of Ψ_1 code is:

$$R = \sum_{i=1}^{q}(P(s_i)i \quad \text{.....(2)}$$

It has been studied elsewhere [3] that this code is optimal for a (monotonically decreasing distribution) source with a first-order entropy around 1, i.e., $1.5 < H < 2.5$. It should be clear that in a monotonically decreasing distribution, the probability of a sample has a large value is low. As a result, the probability of long codeword length Eq. (11) is low. Thus it is optimal. We say that range is the natural entropy range of Ψ_1. For sources with entropies outside that range, there are several variations of Ψ_1.

For example, if $H > 2.5$, it is safe to assume that the LSBs of sample data d*i*(*j*) are completely random. In this case there is no need to perform any compression on those LSBs. We can then split d*i*(*j*) into two portions: *k* LSBs and (8-*k*) MSBs. The MSBs are coded using Ψ_1 code before being sent to bitstream, while the LSBs are sent uncoded. A decoder first recovers the MSBs using Ψ_1 decoder, and then concatenates the results with the uncoded LSBs, resulting in the desired d*i*(*j*). It can be shown that this approach has a natural entropy range $1.5 + k < H < 2.5 + k$. This code is called $\Psi_{1,k}$.

Table 1: Code Word of input data

i	Symbol Si	Sample Xi	Sample Data di	Codeword Wi	Length Li
1	S1	0	0000 0000	1	1
2	S2	1	0000 0001	01	2
3	S3	2	0000 0010	001	3
4	S4	3	0000 0011	0001	4
..	…	…	…	…	…
256	S256	255	1111 1111	0000…01	256

For $0.75 < H < 1.5$ range, we can first use the Ψ_1, resulting in a bitstream with many '1' bits. To exploit this, we can cascade it using another simple code. The simple code first complements the results of Ψ_1 (i.e., converting '0' into '1' and vice versa), and group the resulting bit into binary 3-tuples. It finally codes each binary 3-tuples in a way that a tuple with many '0' bits will use short codewords. This code is called Ψ_0.

For sources with entropy $H < 0.75$, we can cascade the coder with another coder, such as ZRL coder described above. A ZRL encoder processes samples, and provides its outputs to a counter coder. In effect the ZRL coder brings the data entropy into counter code natural range.

Figure 1 shows the flow chart of PSI-1,k encoder.

III. HARDWARE IMPLEMENTATION

A. Data Path Structure

It has been studied elsewhere that the PSI- 1 code discussed in chapter 1 is optimal for monotonically decreasing distribution source with first order entropies outside this, Rice introduced a concept of word splitting. If H > 2.5, it is safe to assume that the LSBs of the sample data di(j) are completely random. In this case there is no need to perform compression on those LSBs. An encoder can split data di(j) into two parts: k LSBs and 8-k MSBs as shown in figure 2.

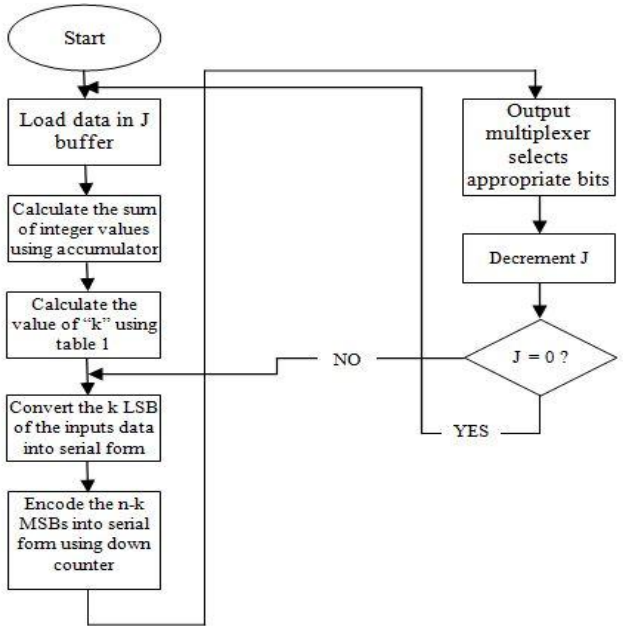

Figure 1: Flow Chart – PSI-1,k

The MSBs are encoded using PSI - 1 coder before being sent to output bit stream. While the LSBs are sent un-coded (or said to encoded using PSI – 3). At the receiver the decoder concatenates the data received.

This counter compression of word splitting is called PSI – 1,k, the entropy range of this enhanced encoder is $1.5 + k < H < 2.5 + k$. The PSI – 1,k encoder is shown in figure 4.2. Now a mechanism of appropriate word splitting is needed at the encoder side.

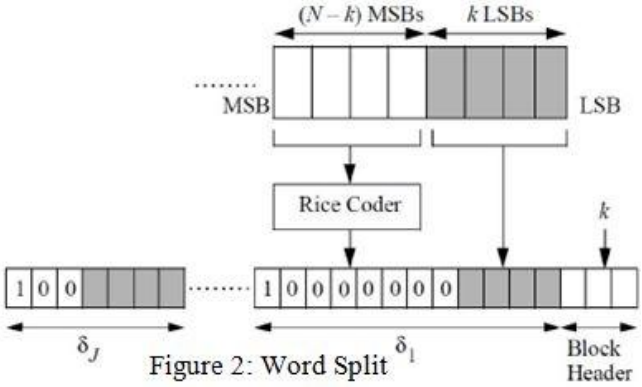

Figure 2: Word Split

B. Splitter

Given a block of data samples (for $j = 1, ..., J$), the $\Psi 1,k$ coder must then have a mechanism to estimate the entropy of the block to ensure it uses the optimal *k*. Rice has come with an estimation rule of thumb based on sum of *xi* values in the block [2]. Since we assume that the data samples has a source according to Table 1 with statistics of a monotonically decreasing distribution, *xi* with low values are likely to occur. The lower the entropy, the more the distribution is skewed toward lower value *xi* (i.e., lower *L*). As a result, a block with lower entropy will have a smaller sum of *xi* values in the block. In other words, average entropy in a block can be estimated from L / J . This rule of thumb is shown in Table 2. A small value of L / J is reflected in a smaller sum of *xi*, corresponding to a low entropy value,

hence a small selected *k*. As mentioned before if the samples do not have such a characteristic, i.e., a nonnegative and monotonically decreasing distribution, there are several simple preprocessing schemes available to preprocess the samples to comply with the characteristics.

Table 2: A rule of thumb to estimate block entropy for $J = 8$ and $n = 8$ for 8 options of coders (adapted from [2])

Decision range	Sum of x_i values	Entropy	k
$LJ \le 5/2$	0 – 12	1.5	0
$5/2 < LJ \le 9/2$	13 – 28	2.5	1
$9/2 < LJ \le 17/2$	29 – 60	3.5	2
$17/2 < LJ \le 33/2$	61 – 125	4.5	3
$33/2 < LJ \le 65/2$	126 – 252	5.5	4
$65/2 < LJ \le 129/2$	251 – 508	6.5	5
$129/2 < LJ \le (128n–831)/2$	509 – 764	7.5	6
$(128n–831)/2 < LJ$	765 – 1023	8	8

C. Proposed Accumulator

It is observed form the base paper adder design that the adder is not in use for most of the duration, so we can replace the 7 RCA structure by a single RCA structure shown in figure 3.

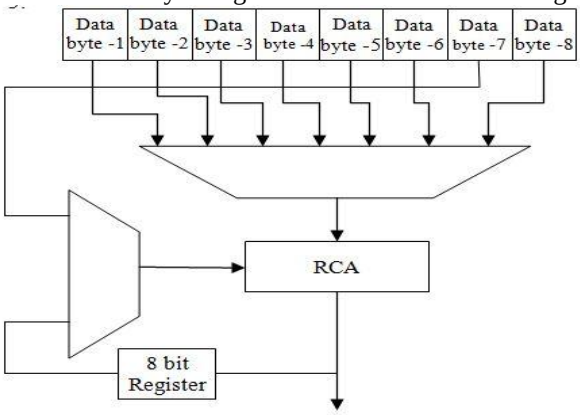

Figure 3: Proposed Accumulator

The proposed adder uses a single RCA, in the first clock cycle the data byte 8 and data byte 7 are added and the result is stored in 8 bit register. In the next clock cycle the data byte 6 is added with the value stored in 8 bit register and so on. This will reduce the area and power consumption of the design.

D. Selection Logic

The selection logic splits the input data bytes into two halves, k LSB bits and n-k MSBs. The proposed architecture of selection logic is shown in figure 4. The comparator shown in figure 4 compares the value of the sum evaluated from the adder shown in figure 3 with internal reference value mentioned in table 2 in this chapter. Then the appropriate value of k is found out, then the input data byte is divided into two halves using a multiplexer.

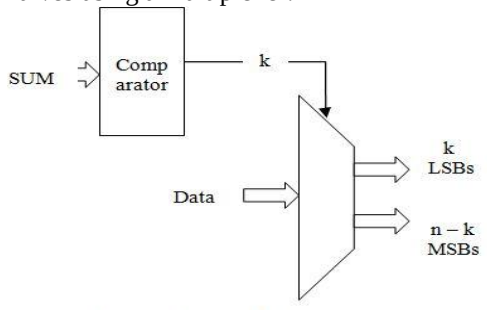

Figure 4: Proposed Selection Logic

E. MSB Encoder

The n-k MSBs encoder is implemented using a simple down counter shown in figure 5.

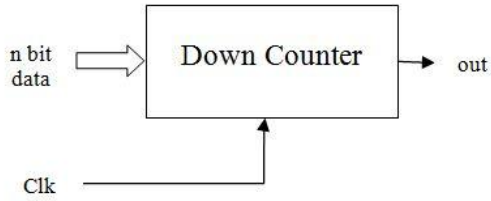

Figure 5: MSB Encoder

The remaining n-k bits are encoded using the down counter shown in figure 5. The counter has n bit of input lines, in our case the value of n is 8, so the input data line is 8 bit wide. The n-k bit data is appended with k zeros and the integer value is loaded in the down counter, now at every rising clock pulse the down counter is decremented by one and the output line ”out” is kept at zero, when the down counter reaches zero, the output line “out” is made 1. This implements the PSI – 1 encoder in the PSI – 1,k rice algorithm.

F. Output Multiplexer

The output multiplexer connects the encoder with serial bit oriented channel. Figure 6 shows the output multiplexer.

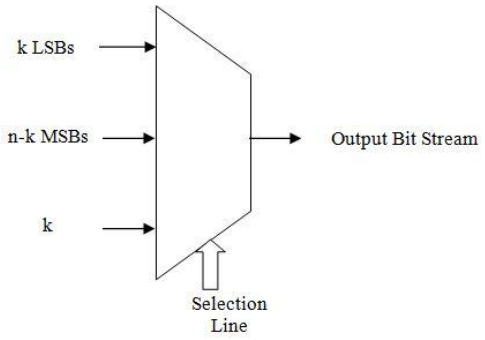

Figure 6: Output Multiplexer

IV. RESULTS

Figure 7 shows the simulation of PSI – 1,k encoder with the transmission value of 4,3,3,3,2,2,2,2.

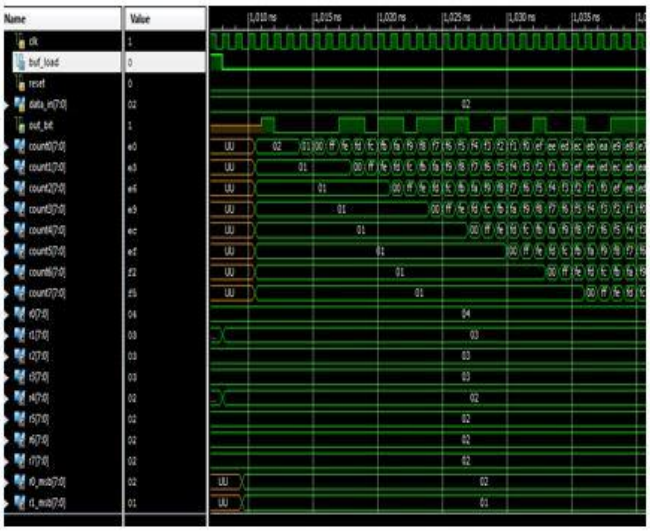

Figure 7: Simulation of PSI – 1,k Encoder

Table 3 shows the resource usage of proposed design of PSI – 1,k encoder. We have implemented the design on Virtex 5 FPGA device.

Table 3: Resource Usage Summary

S.no	Resources	Quantity	% use Proposed Design
1	Number of Slice Registers	305	2%
2	Number of Slice LUTs	660	5%
3	Number of occupied Slices	277	8%

Table 4 shows the power consumption summary of proposed PSI – 1,k encoder, Figure 8 shows the characteristics of power consumption with respect to operating frequency.

Table 4: Power Consumption Summary

S.no	Frequency (Mhz)	Static Power Consumption (mW)	Dynamic Power Consumption (mW)	Total Power Consumption (mW)
1	30	321	11	332
2	50	321	14	335
3	80	321	19	340
4	100	321	22	343
5	150	321	30	350
6	200	321	37	358
7	247 (MOF)	321	43	365

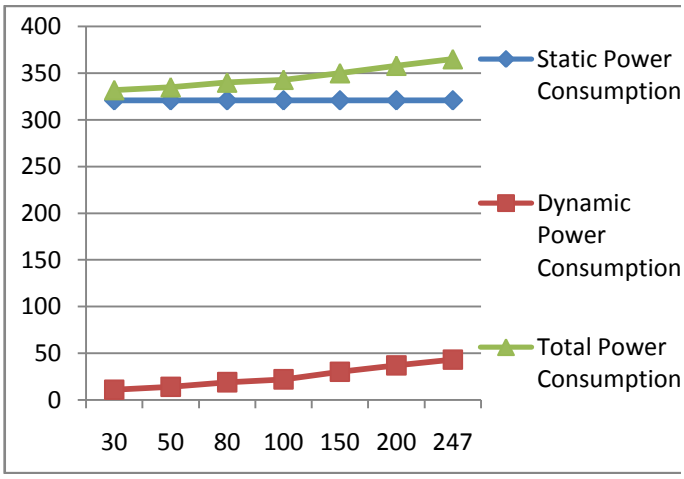

Figure 8: Power Consumption characteristics with frequency

REFERENCES

[1] Armein Z. R. Langi "An FPGA Implementation of a Simple Lossless Data Compression Coprocessor" 2015 International Conference on Electrical Engineering and Informatics 17-19 July 2015, Bandung, Indonesia.

[2] Moussalli, Roger, et al. "A high throughput no-stall golomb-rice hardware decoder." Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on. IEEE, 2013.

[3] Kim, Hong-Sik, et al. "A lossless color image compression architecture using a parallel Golomb-Rice hardware CODEC." IEEE transactions on circuits and systems for video Technology 21.11 (2011): 1581-1587.

[4] Sukhwani, Bharat, et al. "High-throughput, lossless data compresion on FPGAs." Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on. IEEE, 2011.

[5] Meira, M., J. Lima, and L. Batista. "An FPGA implementation of a lossless electrocardiogram compressor based on prediction and Golomb-rice coding." Proc. V Workshop de Informática Médica. 2005.

[6] Malvar, Henrique S. "Adaptive run-length/Golomb-Rice encoding of quantized generalized Gaussian sources with unknown statistics." Data Compression Conference, 2006. DCC 2006. Proceedings. IEEE, 2006.

[7] Seroussi, G.; and Weinberg, m.j. "On Adaptive Strategies for an Extended Family of Golomb-type Codes." Proceedings of the Data Compression Conference, Snowbird, p.131-140, March, 1997.

[8] Rice, Robert F. "Some practical universal noiseless coding techniques, part 3, module PSl14, K+." (1991).

[9] A. Langi, "Review of data compression methods and algorithms", Technical Report, DSP-RTG–2010–9, InstitutTeknologi Bandung, Sep. 2010.

[10] Langi and W. Kinsner, "Wavelet compression for image transmission through bandlimited channels", ARRL QEX Experimenters'sEchange, (ISSN: 0886–8093, USPS 011–424), No. 151, pp. 12–21, Sep. 1994.

[11] Langi, "Lossless Compression Performance of a Simple Counter-Based Entropy Coder", ITB Journal of information and communication technology, submitted Dec 2014.

[12] Langi, "A VLSI Architecture of a Counter-Based Entropy Coder", ITB Journal of Engineering Science, submitted Feb 2013.

[13] Wang, Zhengrong, and Paul Steven Houle. "Data compression using adaptive bit allocation and hybrid lossless entropy encoding." U.S. Patent No. 6,049,630. 11 Apr. 2000.

[14] Wilkinson, James H. "Apparatus for compressing image data employing entropy encoding of data scanned from a plurality of spatial frequency bands." U.S. Patent No. 5,537,493. 16 Jul. 1996.

[15] Ruttimann, Urs E., and Hubert V. Pipberger. "Compression of the ECG by prediction or interpolation and entropy encoding." IEEE Transactions on Biomedical Engineering 11 (1979): 613-623.

[16] Acharya, Tinku. "MMX optimized data packing methodology for zero run length and variable length entropy encoding." U.S. Patent No. 6,195,026. 27 Feb. 2001.

[17] Belyaev, Evgeny, Karen Egiazarian, and Moncef Gabbouj. "A low-complexity bit-plane entropy coding and rate control for 3-D DWT based video coding." IEEE Transactions on Multimedia 15.8

(2013): 1786-1799.

[18] Najmabadi, Seyyed Mahdi, et al. "High throughput hardware architectures for asymmetric numeral systems entropy coding." Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on. IEEE, 2015.

[19] Hu, Nan, and En-Hui Yang. "Fast mode selection for HEVC intra-frame coding with entropy coding refinement based on a transparent composite model." IEEE Transactions on Circuits and Systems for Video Technology 25.9 (2015): 1521-1532.

[20] Rojals, Joel Sole, Rajan L. Joshi, and Marta Karczewicz. "Entropy coding coefficients using a joint context model." U.S. Patent No. 8,913,666. 16 Dec. 2014.

[21] Marpe, Detlev, et al. "Entropy encoding and decoding scheme." U.S. Patent No. 9,252,806. 2 Feb. 2016.

[22] Kumar, BS Sunil, A. S. Manjunath, and S. Christopher. "Improved entropy encoding for high efficient video coding standard." Alexandria Engineering Journal (2016).

