VILOCON - AN ULTRA-LIGHTWEIGHT LOSSLESS VLSI VIDEO CODEC

Shani Rehana, Or Turgeman, Ran Manevich, and Avinoam Kolodny
Electrical Engineering Department, Technion - Israel Institute of Technology
Haifa, Israel

ABSTRACT

A novel lossless image and video codec termed ViLoCoN is presented. ViLoCoN stands for **V**ideo **L**ossless **Co**mpression for **N**oCs (i.e. Networks-on-chip) as it is highly applicable to reduce power consumption and bandwidth of on-chip and inter chip interconnect in modern multimedia SoCs.

ViLoCoN's encoder/decoder use fewer gates (71% and 84%) as compared to previous lightweight encoders and decoders, respectively, and provide a higher compression ratio (by ~31%) on average, as compared to the most lightweight predecessor. ViLoCoN's performance is comparable to well known codecs that demand one order of magnitude more hardware resources.

I. INTRODUCTION

Efficient real-time lossless compression of video streams is useful in multimedia systems and SoCs. It can potentially improve system performance by lowering the bandwidth demands between processors and their off-chip memory [1]. It can also reduce traffic volumes in on-chip interconnect in multimedia chips. Real-time compression in the intermediate stages of signal processing systems should incur minimal power and delay overheads in order to be cost effective. Therefore, hardware oriented approaches with lightweight implementation and few pipeline stages are desirable. General-purpose lossless compression schemes (e.g. [2]) are usually software oriented and are too complex for efficient hardware implementation. Many have proposed VLSI oriented lossless compression techniques that are tailor-made for video signals [3-11]. These techniques perform well when there is correlation among adjacent pixels (in images and video streams) but might yield higher bit rate at their output if such correlation does not exist.

In this paper we present a novel video lossless compression scheme termed ViLoCoN. Our approach is based on custom coding of common sequences of differences between several adjacent pixels. To the best of our knowledge, ViLoCoN is the most lightweight lossless video codec that has been described in literature (5.4K NAND gates, ~23% less than the second most lightweight codec [3]). Moreover, it introduces high compression ratio that is higher by ~31% than its most lightweight predecessor [3] and is comparable to codecs that demand about one order of magnitude more gates (including memory hardware requirements) [4-11].

The paper is organized as follows: section II introduces ViLoCoN's algorithm and its design space. Section III presents RTL implementation and hardware synthesis results. Measurements of compression performance are described in section IV. Section V presents the previous work; section VI concludes the paper.

II. VILOCON – ALGORITHM, CONCEPTS AND PARAMETERS

A. Algorithm

The input for ViLoCoN encoder is a sequence of R-bit words. The algorithm assumes spatial correlation between adjacent words and produces an encoded stream that, if such correlation exists, is at a lower bitrate. KxL Black and White (BW) raw video frame is composed of K·L R-bit words; each word represents a gray level of a single pixel (out of 2^R gray levels). Color video frames are represented by three frames, one for each of the color components[1]. In color videos, ViLoCoN encodes each of the color components separately. For a KxL pixels raw video frame, ViLoCoN gets as input a sequence of its pixels arranged in order, line by line, from the upper left to the bottom right pixel (Figure 1). Without loss of generality, we present the algorithm for a single frame (i.e. image); sequences of adjacent frames in a video stream are concatenated. We denote the *i'th* pixel in the sequence by P_i. The sequence of differences (*DS*) is given by:

[1] In 4-2-0 YCbCr representation, for instance, the Y component have KxL pixels; Cb/Cr components have (K/2)x(L/2) pixels.

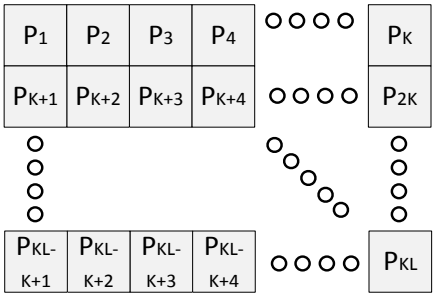

Figure 1 - The order of the pixels in a K by L frame.

$$DS = \left(P_2 - P_1,\ P_3 - P_2,\ \ldots, P_{KxL} - P_{KxL-1}\right) \qquad (1)$$

For any (*i,j*) that satisfy *i<j<KxL*, given P_i and $\{DS_i \ldots DS_{j-1}\}$, the values $P_m \in \{P_{i+1} \ldots P_j\}$ can be calculated as follows:

$$P_m = P_i + \sum_{n=i}^{m-1} DS_n \qquad (2)$$

Moreover, due to the correlation among the gray levels of adjacent pixels, the elements of *DS* are likely to be low. ViLoCoN exploits this property by replacing common sub-sequences of *DS* by shorter codewords. For instance, suppose we have the following two sequences of 5 adjacent pixels:

$PA_{1\text{-}5}$ = *(100, 101, 102, 101, 102)*

$PB_{1\text{-}5}$ = *(100, 200, 150, 50, 200)*

The sequences can also be represented with the value of the first pixel and the adjacent differences of the rest:

PA_1 = *100,* $DSA_{2\text{-}5}$ = *(1, 1, -1, 1)*

PB_1 = *100,* $DSB_{2\text{-}5}$ = *(100, -50, -100, 150)*

Obviously, the sequence $PA_{1\text{-}5}$ is more likely to appear in natural images or video streams than $PB_{1\text{-}5}$ since the differences between adjacent pixels are much lower. If, for example, we have 256 8-bit codewords for the most common differences sequences and (1,1,-1,1) is among them, the sequence $PA_{1\text{-}5}$ can be represented by only 17 bits instead of 40 (8 bits for the first pixel, 1 bit to distinguish between raw pixel values and codewords, and 8 bits for the codeword that represents $DSA_{2\text{-}5}$). We store a limited set of the most common *DS* sub-sequences and their corresponding codewords in a **D**ifferences **C**odes **M**ap (***DCM***). *DS* sub-sequences that have codewords in the *DCM* can be of different lengths (in pixels); to achieve a higher compression rate, ViLoCoN searches the input stream for the longest sub-sequences that are found in *DCM* before the shorter ones. ViLoCoN's encoded stream is composed of words of two types, each type of a constant length. Each word is either a codeword from the *DCM* or an un-coded raw gray-level pixel value. Each word is preceded by a single **D**elimiting **B**it (***DB***) that defines its type (i.e. codeword or raw pixel value). This enables exact reconstruction of the original pixel sequence (i.e. lossless compression). Obviously, if codewords are short and frequent enough, and the overhead of delimiting bits is low, ViLoCoN can achieve lossless bit-rate reduction. The following notation is used throughout the paper:

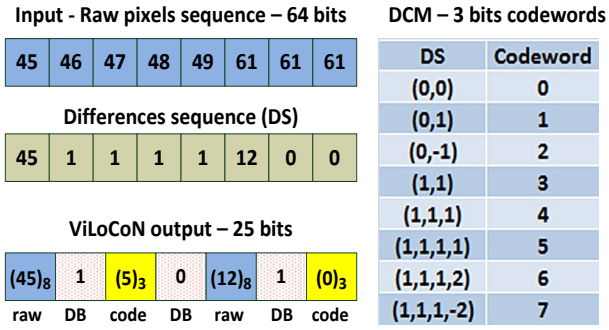

Figure 2 - An example of a sequence of 8 pixels, their differences sequence and ViLoCoN encoded stream using a difference codes map with 3-bit codewords. The differences sub-sequences (1, 1, 1, 1) and (0, 0) are found in the DCM and can be replaced with 3-bit codewords. This way, the 64-bit input sequence is losslessly compressed to 25 bits.

R: Pixel gray-level representation resolution in bits. In this paper, we use *R* = 8 since 256 (2^8) is the most common resolution of gray levels in commercial digital video and imaging.

DCM: **D**ifferences **C**odes **M**ap – A table that matches *C*-bits codewords to the most common differences sequences.

C: The length of the codeword in bits.

N: The maximum length of differences sub-sequences that appear in DCM.

DB: **D**elimiting **B**it for distinguishing between raw pixels and codewords.

ViLoCoN's encoding process, an arbitrary *DCM* and the output stream for an arbitrary input sequence are presented in Figure 2.

B. Selection of ViLoCoN Parameters

In this sub-section we describe the considerations and the process for selection of ViLoCoN design parameters: the length of the codeword (*C*), the

Figure 3 - Six of the benchmark images - an heterogeneous set of pictures that represents a wide variety of possible inputs.

coded differences sub-sequences (i.e. the contents of *DCM*) and the maximum length of difference sub-sequence that can have a corresponding codeword in *DCM (N)*. Limiting *N* to very low values (e.g. 2), on one hand, might limit ViLoCoN's compression ratio. On the other hand, using large values of *N* might incur high computation complexity and implementation costs. Our goal is to choose parameters that yield a sufficient compression ratio with low hardware costs. We explored ViLoCoN's design space using a set of 10 black and white 1920x1080 benchmark images (Figure 3). We found the most common differences sequences and formed DCMs for each combination of *N* and *C* in the ranges $2 \leq N \leq 12$ and $7 \leq C \leq 14$. Average compression ratio results for several *N* and *C* values in our exploration range are presented in Figure 4. Based on the results, we decided to use *N*=4, *C*=10 as the best tradeoff between compression ratio and implementation costs. The common difference sub-sequences were symmetrically concentrated around 0^N (i.e sequence of N zeros). We approximate the optimal DCM by N-dimensional rectangles that contain most of the common differences sub-sequences. Consequently, *DCM* is represented only by the limits of the range of differences sub-sequences length, which results in a very lightweight implementation and a simple

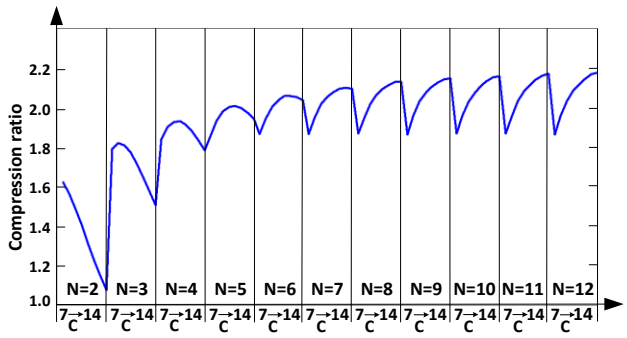

Figure 4 - Compression ratio for N and C combinations. N is fixed and C rises from 7 to 14 in each of the vertical segments.

Table 1: The ranges of difference sub-sequences that have codewords in DCM for N = 4 and C = 10.

	Seq. of 2	***Seq. of 3***	***Seq. of 4***
Upper limit	(10,10)	(3,3,3)	(2,2,2,2)
Lower limit	(-9,-10)	(-3,-3,-3)	(-1,-1,-1,-1)
# of sequences	420	343	256

combinatorial codeword lookup logic. The *DCM* that we finally use for ViLoCoN in this paper is presented in Table 1. Differences sub-sequences between the lower and the upper levels in DCM are arranged in order such that, for instance, the sub-sequence (-9,-10) is coded as "0", (-9,-9) as "1", (10,10) as "419", (-3,-3,-3) as "420", and (2,2,2,2) as 420+343+256-1 = "1018".

III. IMPLEMENTATION

A. Encoder

ViLoCoN encoder is composed of four modules: difference sequence generator, *DCM* lookup and codeword calculation module, output controller and main control module. The input of the encoder is a sequence of 8-bit pixels, one pixel per cycle. For maximum length *DS* sub-sequence of 4 (*N* = 4), the encoder might produce a codeword only once every 4 clock cycles (if *DS* is composed of sub-sequences of 4 that have codewords in *DCM*). The output of the encoder, on each cycle, can be: idle, raw pixel (1 *DB* + 8 raw bits = 9 bits), or codeword (1 *DB* + 10 bits codeword = 11 bits). Therefore the output bus is 11 bits wide. We have designed ViLoCoN encoder using Verilog and have synthesized and produced it's layout in 65nm technology. Our design could sustain up to ~1GHz (a throughput of 1 GB/s) and requires ~1.8K NAND gates. A block diagram of the encoder is presented in Figure 5.

The encoder has a 5-pixels shift register at its input stage. The 5 pixels are needed to generate the 4 (*N* = 4) adjacent differences from the pixel P_i ($DS_{i+1}...DS_{i+4}$). We use four full-subtractors to calculate the differences. DCM lookup blocks indicate whether differences sub-sequences of 2, 3, and 4 adjacent pixels starting from the pixel P_i have codewords in *DCM*. There is a sub-module for each of the possible lengths (2, 3, and 4). If one of the sub-sequences exists, the corresponding valid signal is set. The identification is performed by checking whether each of the *DS* sub-sequences is

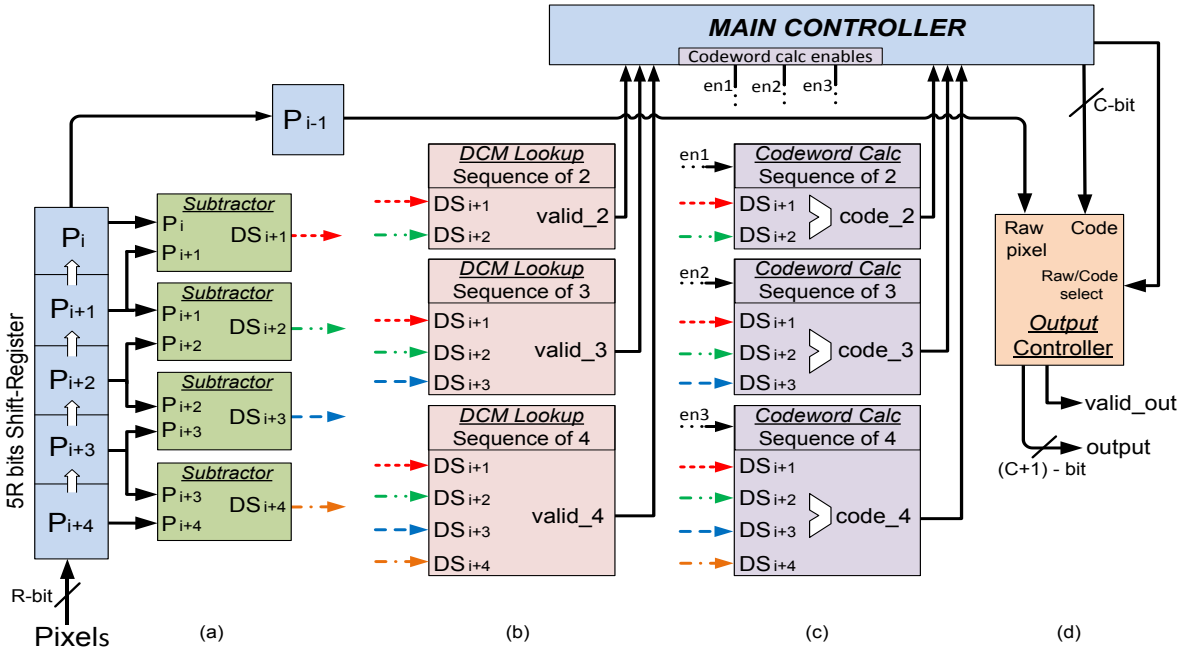

Figure 5 - Architecture of ViLoCoN compression scheme: 5 pixels are turned into ($DS_{i+1}...DS_{i+4}$) using subtractors (a), *DCM* lookup blocks indicate whether differences sub-sequences of 2, 3, and 4 adjacent pixels have codewords in *DCM* (b,c), and the output controller arranges the codeword or raw pixel values into chunks of 11 bits at the output (d).

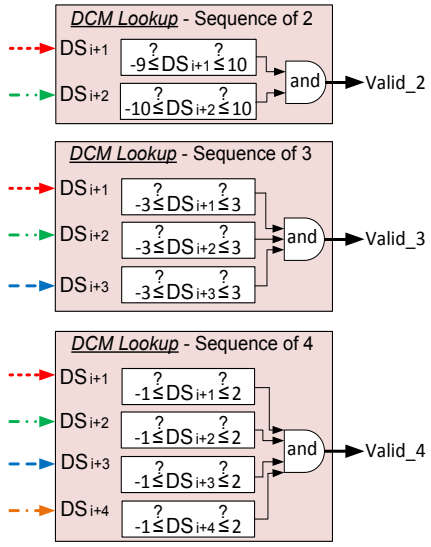

Figure 6 - DCM Lookup Block: indicates whether ($DS_{i+1}...DS_{i+4}$) are between the 2 boundaries for each of the DS sub-sequence lengths (Table 1).

between the 2 boundaries as described in Table 1. Figure 6 presents the inner structure of the *DCM* lookup sub-modules. Since *DCM* is defined only by difference sub-sequences ranges, and sub-sequences in *DCM* are arranged in order, codeword calculation is a simple combinatorial procedure. For $N = 4$, $C = 10$ and the *DCM* defined by Table 1, codewords for each of the *DS* sub-sequence lengths are calculated according to equations 3-5.

$$\text{Codeword}_2 = 21 \cdot \left(DS_{i+1} - (-9)\right) + \left(DS_{i+2} - (-10)\right) \quad (3)$$

$$\text{Codeword}_3 = \begin{aligned} &49 \cdot \left(DS_{i+1} - (-3)\right) + 7 \cdot \left(DS_{i+2} - (-3)\right) \\ &+ \left(DS_{i+3} - (-3)\right) + 420 \end{aligned} \quad (4)$$

$$\text{Codeword}_4 = \begin{aligned} &64 \cdot \left(DS_{i+1} - (-1)\right) + 16 \cdot \left(DS_{i+2} - (-1)\right) \\ &+ 4 \cdot \left(DS_{i+3} - (-1)\right) + \left(DS_{i+4} - (-1)\right) + 763 \end{aligned} \quad (5)$$

ViLoCoN encoder output is composed of 11 data bits and a valid_output signal. At every clock cycle, the valid_output signal indicates whether there are 11 new valid bits on the data bus. As described earlier, the output of the encoder on each clock cycle can be 0, 9 or 11 bits. The output controller arranges the stream into chunks of 11 bits and provides a simple parallel-bus output interface.

The main controller is responsible for managing and synchronizing the entire mechanism. It gets the indication whether there are *DS* sub-sequences that have codewords in *DCM*. If more than one among *valid_x* signals is active, it prioritizes the longer sequences. Finally, if a *DS* sub-sequence is

encoded, the controller stalls the encoder until all the pixels that have been already encoded vacate the input shift-register.

B. Decoder

We have implemented ViLoCoN decoder that decodes the output stream of the previously described encoder. The decoder was implemented with similar CAD tools and the same 65nm process. Its area and maximal clock frequency values are 3.6K NAND gates and 1GHz respectively.

IV. EVALUTION

We estimate ViLoCoN's compression ratio using 6 popular 1080p and 720p YouTube video clips. We downloaded the videos from YouTube and calculated their compression ratio using ViLoCoN simulator that we wrote in Matlab. We calculate compression ratios of luma (Y) only and all color components (4:2:0 YCbCr) combined. Table 2 summarizes ViLoCoNs compression ratio for various input types. Details and compression ratios of each of the YouTube clips are found in Table 3. The average luma and YCbCr compression ratios of the videos (both 1080P and 720P) are 2.175 and 2.37, corresponding to data rate reduction of 54% and 58%, respectively.

V. PREVIOUS WORK

Many VLSI-oriented lossless video compression schemes have recently been proposed. All the codecs that we have encountered during our background research were devised for 8-bits/pixel BW video streams. Color videos in these codecs are usually treated as a set of 3 separate components (e.g YUV, YCbCr, RGB) with 8 bits/component/pixel. The codecs can also be divided into block based [4], [6-7], [10-11] and line-based [3], [5], [8-9]. Block based schemes exploit correlation between pixels on both horizontal and vertical dimensions and encode entire blocks of pixels (e.g. 8x8). These encoders usually demand their input to be ordered in blocks of pixels and provide higher throughputs than 1 pixel/cycle. Line based codecs exploit the correlation between line-wise adjacent pixels. These encoders operate with a serial stream of pixels in their input and typically process 1 pixel/cycle. They replace sets of adjacent pixels by shorter codewords that are usually stored in a look-up table. ViLoCoN belongs to the line-based codecs family. Its implementation demands significantly less hardware than the previous schemes mainly due to the lightweight implementation of the codewords dictionary. The dictionary logic first indicates whether a difference sequence has a corresponding codeword (DCM_Lookup stage in Figure 5) and then produces codewords for the sequences that successfully passed the first stage (Codeword_Calc stage in Figure 5). Instead of storing the mapping between difference sequences and codewords explicitly, the dictionary in ViLoCoN logic uses only the boundaries of difference sequences ranges (Table 1) for both the codeword look-up and codeword calculation stages. The look-up logic checks whether the input sequence is inside one the ranges that have codewords in the dictionary (Figure 6). Codeword calculation logic finds the codeword by calculating the "distance" of the differences sequence form the lower boundary of the range (Eq. 3-5). Both stages require only a few combinatorial comparators, adders and subtractors, what enables the implementation of large dictionary with hardware that is equivalent to less than 1K NAND gates. A comparison between ViLoCoN and previous

Table 2 : Y compression ratio for different types of inputs

Input	*Images*	*Videos*	
	1920x1080	*1920x1080*	*1280x720*
Y Comp. ratio	1.81	2.28	2.07

Table 3 : Compression ratio of YouTube video clips

Video	*URL*	*# of views (16.03.2013)*	*Res.*	*C. ratio (Y)*	*C. ratio (YCbCr)*	*# of frames*
PSY - GANGNAM STYLE	http://www.youtube.com/watch?v=9bZkp7q19f0	1,431,934,513	1080P	2.22	2.38	6,046
GoPro HERO3: Black…	http://www.youtube.com/watch?v=A3PDXmYoF5U	17,697,154		2.25	2.44	9,223
Kobe vs Messi:…	http://www.youtube.com/watch?v=ruav0KvQOOg	103,341,845		2.38	2.54	1,520
Britney Spears – Hold…	http://www.youtube.com/watch?v=-Edv8Onsrgg	85,362,379	720P	2.00	2.20	6,724
What A Wonderful…	http://www.youtube.com/watch?v=auSo1MyWf8g	10,756,894		2.12	2.33	3,000
Formula 1 comes to…	http://www.youtube.com/watch?v=0tEHQJn_hxo	2,598,531		2.08	2.31	11,142

Table 4 : Comparison between ViLoCoN and other encoders

Algorithm	*ViLoCoN*	*DAP + Huffman [3]*	*Sig. Bit Trunc. [4]*	*Dict. Based [5]*	*DDPCM+ Golomb Rice [6]*	*DWT+ SPIHT [7]*	*FELICS + CDP [8]*	*HPE [9]*		*Hwang's [11]*
Parallelism	1	1	14.2	3	16	1	2	1	12.8	1
Compression ratio (Y)	2.175	NA	2.14	NA	1.47	NA	2.35	NA	2.6	NA
Compression ratio (YCbCr)	2.37	1.81	2.52	2.18	1.52	2.00	2.31 RGB	3.09	NA	2.53
Codec Logic (NAND gates)	5.4K	7K	36.1K	23.9K	32.46K	27K	12.97K*	15.7K	28K	35k*
Memory (Bytes)*	0	0	0	5.4K	0	1.28K	1.9K	0.5K	0	3.7k
CEff $_N$(Y)	1	NA	0.145	NA	0.066	NA	<0.143	NA	0.262	NA
CEff$_N$ (YCbCr)	1	0.456	0.166	0.042	0.063	0.083	<0.119	0.347	NA	<0.064

* In [8] and [11] the area is known only for the encoder. ** Each memory bit is counted as 2 NAND gates.

encoders with regard to popular performance and implementation metrics is presented in Table 4. We have also defined codec efficiency (*CEff*) as:

$$CEff = \frac{Compression\ Ratio - 1}{Codec\ Logic\ [K\ NAND\ gates]} \quad (6)$$

Normalized *CEff* is defined as:

$$CEff_N = \frac{CEff}{CEff_{ViLOCoN}} \quad (7)$$

In codecs that use memory for the calculation of *CEff*, we counted each memory bit as 2 gates.

ViLoCoN codec demands 23% less NAND gates than the previous most lightweight codec (encoder + decoder) [3]. The compression ratio of ViLoCoN outperforms [3] by ~31% and is comparable to codecs that occupy more hardware by about 1 order of magnitude (including memory hardware requirements) [4-11].

VI. CONCLUSIONS

This paper presents, to the best of our knowledge, the most lightweight and efficient lossless VLSI video codec termed ViLoCoN. We described ViLoCoN's algorithm and implementation in details and analyzed its performance on popular YouTube HD video clips. ViLoCoN codec requires ~23% less than the second most lightweight codec [3]. It introduces a compression ratio of 2.175 for luma, and 2.37 for 4:2:0 YCbCr on average - higher by 31% compared to its most lightweight predecessor [3].

VII. ACKNOWLEDGMENT

The authors would like to thank the Technion EE VLSI laboratory staff and Mr. Goel Samuel especially for their valuable time and support.

REFERENCES

[1] Roy, Sumit, Raj Kumar, and Milos Prvulovic. "Improving system performance with compressed memory." Parallel and Distributed Processing Symposium., Proceedings 15th International. IEEE, 2001.

[2] Ziv, Jacob, and Abraham Lempel. "Compression of individual sequences via variable-rate coding." Information Theory, IEEE Transactions on 24.5 (1978): 530-536.

[3] Song, Tian, and Takashi Shimamoto. "Reference frame data compression method for H. 264/AVC." IEICE Electronics Express 4.3 (2007): 121-126.

[4] Kim, Jaemoon, and Chong-Min Kyung. "A lossless embedded compression using significant bit truncation for HD video coding." Circuits and Systems for Video Technology, IEEE Transactions on 20.6 (2010): 848-860.

[5] Yang, Hui-Ting, et al. "An effective dictionary-based display frame compressor."Embedded Systems for Real-Time Multimedia, 2009. ESTIMedia 2009. IEEE/ACM/IFIP 7th Workshop on. IEEE, 2009.

[6] Kim, Hong-Sik, et al. "A Lossless Color Image Compression Architecture Using a Parallel Golomb-Rice Hardware CODEC." Circuits and Systems for Video Technology, IEEE Transactions on 21.11 (2011): 1581-1587.

[7] Cheng, Chih-Chi, Po-Chih Tseng, and Liang-Gee Chen. "Multimode embedded compression codec engine for power-aware video coding system." Circuits and Systems for Video Technology, IEEE Transactions on 19.2 (2009): 141-150.

[8] Tsai, Tsung-Han, Yu-Hsuan Lee, and Yu-Yu Lee. "Design and analysis of high-throughput lossless image compression engine using VLSI-oriented FELICS algorithm." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 18.1 (2010): 39-52.

[9] Li, Yifu, Wenxiang Wang, and Guangfei Zhang. "Hybrid Pixel Encoding: An Effective Display Frame Compression Algorithm for HD Video Decoder."Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on. IEEE, 2012.

[10] Kim, Jaemoon, Jungsoo Kim, and Chong-Min Kyung. "A lossless embedded compression algorithm for high definition video coding." Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on. IEEE, 2009.
[11] Hwang, Yin-Tsung, Chen-Cheng Lin, and Ming-Wei Lyu. "Design and implementation of a low complexity lossless video codec." Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on. IEEE, 2010.