An Efficient Lossless Embedded Compression Engine Using Compacted-FELICS Algorithm

Yu-Yu Lee, Yu-Hsuan Lee, and Tsung-Han Tsai
DSP/VLSI Lab Department of Electronics Engineering, National Central University
No.300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan, R.O.C.
Email : {kimkafa, yxlee, han}@dsp.ee.ncu.edu.tw

ABSTRACT

The memory bandwidth and capacity have become a critical design issue in display media chip for high-end display applications. In this paper, the lossless embedded compression engine using compacted-FELICS algorithm, which primarily consists of adjusted binary code and Golomb-Rice code, is proposed to handle this scenario. The encoding capability of its VLSI architecture can achieve Full-HD 1080p@60Hz. The prototype chip is implemented by TSMC 0.18-um with Artisan cell library, and its core size is 0.98mm x 0.97mm.

I. INTRODUCTION

Owing to the rapid evolution of liquid crystal display (LCD) panel technology, the high definition (HD) display becomes a mainstream in modern life, and the high-speed digital transmission interface, High Definition Multimedia Interface (HDMI) [1], is developed to joint this trend. Even though the HDMI can support the capacity of HD series, it also induces significant requirement on memory bandwidth and processing speed for related display media chips. The processing speed can be properly handled by the great innovation of semiconductor technology; however, the memory bandwidth is still a design bottleneck. For multi-chip system, the access power of DRAM I/O contributes about 60% of total power dissipation in video phone system [2]. In order to reduce the I/O power consumption of frame buffer, the frame buffer has been realized as on-chip DRAM [2]-[4] or SRAM [5]. The yield rate of chip fabrication is seriously limited by the capacity of embedded storage. Once the embedded storage is increased for high-end display specifications, the yield rate issue could be raised and impacts the system cost.

In order to overcome this drawback, the embedded compression (EC) engine is utilized to reduce the requirement of memory bandwidth as well as the I/O power of frame buffer. Since the EC engine is integrated into display media chip, as shown in Fig. 1, the processing speed should be fully compatible with other function modules. Therefore, the throughput of EC engine should be as high as possible especially for high resolution applications. To keep the hardware cost acceptable, the gate count of EC engine should be more compacted. Moreover, the coding efficiency should be also maintained to provide reasonable compression ratio (CR). In summary, the EC engine contains following features: high throughput, compacted gate count and reasonable coding efficiency.

Many sophisticated lossless compression methods apply more complicated algorithm to improve the coding efficiency, such as JPEG-LS [6] and CALIC [7]. However, their complexity and data dependency seriously limits the potentiality to be EC engine. Among these sophisticated methods, fast, efficient, lossless image compression system (FELICS) algorithm [8] could be a suitable algorithm for EC engine and provide competitive coding efficiency [6]; however, the data dependency still exists and limits its processing capability on high-throughput applications. In this paper, the compacted-FELICS algorithm is proposed to be the EC engine, and its encoding capability of VLSI architecture can achieve Full-HD 1080p@60Hz. The rest of this paper is organized as follows. The compacted-FELICS algorithm is described in Section II, and the proposed VLSI architecture is demonstrated in Section III. In Section IV, experimental results and discussion are presented. Finally, conclusions are given in Section V.
II. The Proposed compacted-FELICS Algorithm

A. The Coding Flow of compacted-FELICS Algorithm

As show in Fig. 2, the prediction template indicates the relationship between current and reference pixels. Each current pixel is assigned to one of three sections, including In range, Above range and Below range. For N1 and N2, the larger one is regarded as H, and L is for the other one. If the intensity of current pixel is between N1 and N2, In range is assigned to it. When being greater than H, the current pixel is assigned to Above range. The Below range is for the current pixel less than L. With the description mentioned above, the compacted-FELICS coding flow is described as following steps

1) The first two pixels at first row are directly packed into bitstream without any encoding procedure.
2) According to the prediction template in Fig. 2, find the two reference pixels, N1 and N2.
3) Assign L = min(N1, N2), H = max(N1, N2), and delta = H-L.
4) Apply adjusted binary code for P-L in In range, Golomb-Rice code for L-P-1 in Below range, and P-H-1 in Above range.

Except first two pixels at first row, the others directly start from step 2 to 4. The entire coding flow can be reversely performed as decoding flow. In original FLEICS, both adjusted binary code and Golomb-Rice code present serious data dependency and considerable computation requirement. In following subsections, more efficient adjusted binary code and Golomb-Rice code are provided.

B. Adjusted binary code for compacted-FELICS algorithm

In In range, the intensity of current pixel is between H and L. The P-L is in the range of [0, delta], where

delta denotes H-L. If delta+1 is exactly equal to power of two, the P-L is encoded with $log_2(delta+1)$ bit. Otherwise, the required number of bit is represented as following equation

$$\left\lfloor log_2(delta + 1) \right\rfloor \leq \#\text{ bit \_ sample} \leq \left\lceil log_2(delta + 1) \right\rceil$$  (1)

where $\#\text{ bit \_ sample}$ stands for the required number of bit to represent the sample of P-L. Based on (1), the sample of P-L could be assigned to $[log_2(delta + 1)]$ or $[log_2(delta + 1)]$ bit. The coding flow of adjusted binary code is illustrated in Fig. 3(a), and an example of delta = 4 is also presented in Fig. 3(b). According to (1), the required number of bits is 2 or 3 for delta = 4. Then, the parameter computation of range and threshold is calculated. If the P-L is less than threshold, the $[log_2(delta + 1)]$ bit is directly assigned to P-L. Otherwise, the P-L is added to threshold, and encoded with $[log_2(delta + 1)]$ bit. The main reason is that since smaller P-L should be frequently occurred than larger one, more efficient codeword should be allocated to it. In Fig. 3(b), the parameter of range and threshold is 5 and 3, respectively. Based on the principle mentioned above, the P-L less than threshold is allocated to 2 bit, and 3 bit for the P-L which is greater than or equal to threshold. The generated codeword in Fig. 3(b) is consistent with the principle mentioned above.

C. Golomb-Rice code for compacted-FELICS algorithm

For both Above range and Below range, Golomb-Rice code is adopted as the coding tool. The codeword of sample $x$, P-H-1 or L-P-1, is partitioned into unary and binary part. The number
III. The Proposed VLSI Architecture for Compacted-FELCIS Algorithm

The proposed VLSI architecture mainly consists of prediction template module, intensity processing module, adjusted binary code, Golomb-Rice code and bitstream generator. Because data dependency has been completely removed by our proposed method, the pipeline stage can be inserted among primary blocks to further improve processing speed. The block diagram of proposed architecture with two-level parallelism and four-stage pipelining is illustrated in Fig. 4. The prediction template module adaptively adjusts the data path from case1 to case4 in prediction template, and prepares relative reference pixels for following prediction. According to the prediction template, the intensity processing module identifies which coding mode should be selected and generates the corresponding residual, such as P-L for adjusted binary code and P-H-1 or L-P-1 for Golomb-Rice code. Since the current pixel is encoded with adjusted binary code or Golomb-Rice code, both corresponding modules are interconnected in parallel for individual data flow. In addition, the line buffer is applied to store relative reference pixels for each current pixel, and the capacity of it is exactly identical to the width of display resolution. Since both adjusted binary code and Golomb-Rice code belong to variable length code (VLC), the bitstream generator is responsible to pack them into fixed bit length for transmission on dedicated bus width. The control unit precisely schedules each primary block to accomplish complete encoding flow.

As illustrated in Fig. 5, two pixels are classified into a group, and each group is regarded as basic processing unit in proposed VLSI architecture. Because the data dependency problem is removed by our proposed method, the pipelining data scheduling presents compact and regular data flow without data hazard and bubble cycle. According to the prediction template in Fig. 2, the reference pixel is reused within the same group, and boundary one is reused with adjacent groups. Therefore, the prediction template module can be concurrently shared among each current pixel of the same group and is represented as share core as well as control unit. Although the prediction template module is shared with each pixel within the same group, the coding mode of each current pixel could not be completely identical. Hence, a parallel core is individually assigned to each current pixel within the same group, and it contains following primary blocks: intensity processing, adjusted binary code and Golomb-Rice code. The output codeword of each parallel core is concatenated into the same bitstream by the other share core, bitstream generator. Since the prediction template module concurrently accesses individual reference pixels for each current pixel within the same group, the line buffer is partitioned into two segments with ping-pong operation.

![Fig. 4. The proposed VLSI architecture with two-level parallelism and four-stage pipelining.](image-url)

![Fig. 5. The data scheduling for two-level parallelism and four-stage pipelining in proposed VLSI architecture.](image-url)
IV. Experiment Result and Discussion

Our design is implemented by TSMC 0.18-um with Artisan cell library. The performance evaluation with other works is shown in Table I. Among these works, our proposed architecture presents higher operation frequency and more compacted gate count. The main reason can be categorized into two points. First, the processing speed is dominated by algorithm complexity. If the embedded compression algorithm consists of many multiplication operations such as transformation, then the operation speed is seriously limited. Therefore, the algorithm complexity should be kept as low as possible. Second, the embedded compression algorithm should be capable of parallel processing. Most lossless compression algorithms are with higher data dependency, and it seriously limits the performance of hardware parallelism.

The hardware performance of [10] and [11] are seriously limited by the reason mentioned above. Although article [9] modifies JPEG-LS to be capable of parallel processing, its gate count is significantly increased and unacceptable. Based on proposed compacted-FELICS algorithm, the encoding capability of our proposed architecture can achieve Full-HD1080p@60Hz. The layout view and specification of prototype chip is illustrated in Fig. 6 and Table II, respectively.

V. Conclusions

In this paper, the embedded compression approach using compacted-FELICS algorithm is proposed to remove the data pendency and improve processing speed. Experiment results show that the proposed VLSI architecture demonstrates superior performance in comparison with other works. The prototype chip is implemented by TSMC 0.18-um with Artisan cell library, and the encoding capability can achieve Full-HD1080p@60Hz with two-level parallelism and four-stage pipelining.

REFERENCES