European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 08, Issue 01, 2021

Intelligent and Efficient Video Embedded Memory Software

[1]SriramojiVenkat Prasad Chary, [2]P. Munaswamy
[12]Department of electronics and communication engineering, Institute of Aeronautical Engineering, Hyderabad-500043, India
[1]sriramojichary007@gmail.com, [2]sidduvamsi@gmail.com

ABSTRACT

Volatile memories like double data rate random access memory or non-volatile memories like embedded flash memory are key components in most of the embedded systems, especially in today's mobile video processing systems. There is a requirement to use the on-chip and off-chip embedded memories very efficiently to meet the end user requirements. Increasingly dominating power consumption and shortening battery life of mobile devices is a key factor which is to be considered while designing and developing the embedded systems and embedded software solutions. Traditional hardware-level power optimization techniques usually come with significant implementation overhead to solve the memory failure problems during low-voltage operations. This paper presents advanced mobile video memory optimization techniques which indirectly reduces power consumption and increases the overall system performance. The viewing contexts significantly influence the quality of the viewer experience, and in the context with higher quality, mobile users have higher tolerance to the video degradation. Accordingly, the memory failures can be introduced adaptively to achieve power savings without influencing the viewer experience. To meet the silicon area constraint in mobile devices, a simple but an efficient hardware implementation scheme can be developed based on these optimization techniques to minimize area overhead. Input mp4 video will be processed by various techniques like bicubic, bilinear, nearest neighbor and H264 compression to generate output mp4 videos with various memory sizes. The simulation results will also be provided.

Keywords: Embedded systems;H265; Nearest Neighbor method;RAM.

1. Introduction

Many types of memory devices are available for use in modern computer systems. As an embedded software engineer, we must be aware of the differences between them and understand how to use each type effectively. We will approach these devices from the software developer's perspective. Keep in mind that the development of these devices took several decades and that their underlying hardware differs significantly. The names of the memory types frequently reflect the historical nature of the development process and are often more confusing than insightful. The RAM family includes two important memory devices: static RAM (SRAM) and dynamic RAM (DRAM). The primary difference between them is the lifetime of the data they store. SRAM retains its contents as long as electrical power is applied to the chip. If the power is turned off or lost temporarily, its contents will be lost forever. DRAM, on the other hand, has an extremely short data lifetime-typically about four milliseconds. This is true even when power is applied constantly.

In short, SRAM has all the properties of the memory you think of when you hear the word RAM. Compared to that, DRAM seems kind of useless. By itself, it is. However, a simple piece of hardware called a DRAM controller can be used to make DRAM behave more like SRAM. The job of the DRAM controller is to periodically refresh the data stored in the DRAM. By refreshing the data before it expires, the contents of memory can be kept alive for as long as they are needed. So, DRAM is as useful as SRAM after all.When deciding which type of RAM to use, a system designer must consider access time and cost. SRAM devices offer extremely fast access times (approximately four times faster than DRAM) but are much more expensive to produce. Generally, SRAM is used only where access speed is extremely important. A lower cost-per-byte makes DRAM attractive whenever large amounts of RAM are required. Many embedded systems include both types: a small block of SRAM (a few kilobytes) along a critical data path and a much larger block of DRAM for everything else.

Memories in the ROM family are distinguished by the methods used to write new data to them (usually called programming), and the number of times they can be rewritten. This classification reflects the evolution of ROM devices from hardwired to programmable to erasable-and-programmable. A common feature of all these devices is their ability to retain data and programs forever, even during a power failure.The very first ROMs were hardwired devices that contained a pre-programmed set of data or instructions. The contents of the ROM had to be specified before chip production, so the actual data could be used to arrange the transistors inside the chip. Hardwired memories are still used, though they are now called “masked ROMs” to distinguish them from other types of ROM. The primary advantage of a masked ROM is its low production cost. Unfortunately, the cost is low only when large quantities of the same ROM are required.One step up from the masked ROM is the PROM (programmable ROM), which is purchased in an unprogrammed state. If you were to look at the contents of an unprogrammed PROM, you would see that the data is made up entirely of 1's. The process of writing your data to the PROM involves a special piece of equipment called a device programmer. The device programmer writes data to the device one word at a time by applying an electrical charge to the input pins of the chip. Once a PROM has been programmed in this way, its contents can never be changed. If the code or data stored in the PROM must be changed, the current device must be discarded. As a result, PROMs are also known as one-time programmable (OTP) devices.

An EPROM (erasable-and-programmable ROM) is programmed in the same manner as a PROM. However, EPROMs can be erased and reprogrammed repeatedly. To erase an EPROM, you simply expose the device to a strong source of ultraviolet light. (A window in the top of the device allows the light to reach the silicon.) By doing this, you essentially reset the entire chip to its initial-unprogrammed-state. Though more expensive than PROMs, their ability to be reprogrammed makes EPROMs an essential part of the software development and testing process.As memory technology has matured in recent years, the line between RAM and ROM has blurred. Now, several types of memory combine features of both. These devices do not belong to either group and can be collectively referred to as hybrid memory devices. Hybrid memories can be read and written as desired, like RAM, but maintain their contents without electrical power, just like ROM. Two of the hybrid devices, EEPROM and flash, are descendants of ROM devices. These are typically used to store code. The third hybrid, NVRAM, is a modified version of SRAM. NVRAM usually holds persistent data.EEPROMs are electrically-erasable-and-programmable. Internally, they are similar to EPROMs, but the erase operation is accomplished electrically, rather than by exposure to ultraviolet light. Any byte within an EEPROM may be erased and rewritten. Once written, the new data will remain in the device forever-or at least until it is electrically erased. The primary tradeoff for this improved functionality is higher cost, though write cycles are also significantly longer than writes to a RAM. So you wouldn't want to use an EEPROM for your main system memory.Flash memory combines the best features of the memory devices described thus far. Flash memory devices are high density, low cost, nonvolatile, fast (to read, but not to write), and electrically reprogrammable. These advantages are overwhelming and, as a direct result, the use of flash memory has increased dramatically in embedded systems. From a software viewpoint, flash and EEPROM technologies are very similar. The major difference is that flash devices can only be erased one sector at a time, not byte-by-byte. Typical sector sizes are in the range 256 bytes to 16KB. Despite this disadvantage, flash is much more popular than EEPROM and is rapidly displacing many of the ROM devices as well.

2. Block Diagram of the proposed work and its description

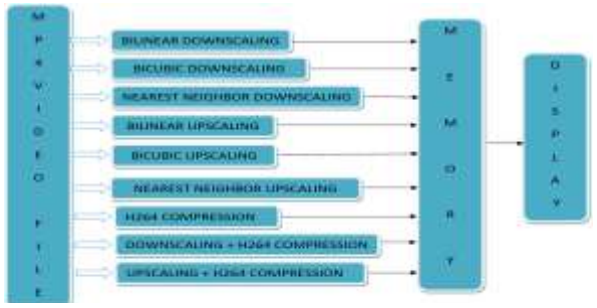
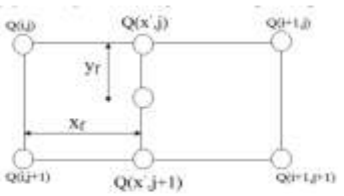


Fig. 1. Block Diagram of the proposed.

We will be taking a sample MP4 video file as input for testing the various techniques proposed by us. In this process we will develop the algorithm for the various techniques and show how the videos are processed before sending them to the memory. All the processed videos which are in the memory, will be read back and displayed to check the video quality. Image scaling has been widely applied in the fields of digital imaging devices such as digital cameras, digital video recorders, digital photo frame, high-definition television, mobile phone, tablet PC, etc. An obvious application of image scaling is to scale down the high-quality pictures or video frames to fit the mini size liquid crystal display panel of the mobile phone or tablet PC. As the graphic and video applications of mobile handset devices grow up, the demand and significance of image scaling are more and more outstanding. In many applications, from consumer electronics to medical imaging it is desirable to improve the restructured image quality and processing performance of hardware implementation. The image scaling algorithm is of two types: they are polynomial based and non-polynomial based algorithms.

The simplest polynomial based algorithm is the nearest neighbor algorithm. The nearest neighbor algorithm is easy to implement and has less complexity but the images produced are full of blocking and aliasing artifacts. The bilinear interpolation is another polynomial based algorithm which uses linear interpolation model to calculate the value of the unknown pixels. The image produced by the bilinear interpolation method is popular due to its computational efficiency and image quality. The image produces blurring and aliasing effects upon scaling. The bicubic interpolation algorithm which is an extension of the cubic interpolation for interpolating pixels on a 2D regular grid. The images produced by the bicubic image interpolation is of high quality but it has high complexity and high memory requirements which makes it difficult to implement it in a Very large scale Integration(VLSI).

The non-polynomial based algorithms used in image scaling processors are auto regressive model bilateral filter and curvature interpolation. The area pixel mode algorithm called Winscale was proposed by kim. The Winscale algorithm uses a maximum of four pixels and calculates the luminosity of each pixel from grayscale to color image. The Winscale algorithm is more computing resource than the other algorithm. The bilinear algorithm has lower complexity and memory requirement than the other algorithms. The blurring and the aliasing effect caused by the linear interpolation can be smoothed by using a sharpening and clamp filter.Bilinear interpolation is an image restoring algorithm which interpolates the neighboring pixels of an unrestored image to obtain the pixel of a restored image. The linear interpolation is done in one direction and then the same function is repeated in other directions. Bilinear interpolation takes four neighboring pixels to calculate the target pixel. Bilinear interpolation is a popular interpolation technique used in image scaling.



The Q(i,j), Q(i+1,j), Q(i,j+1) and Q(i+1,j+1) are the nearest neighbor pixel of the original image with i = [0,1,2,…M] and j = [0,1,2…N]. Where M is the number of pixels having the width of the original image and N is the number of the pixels corresponding to the length of the image. The temporary pixel created by the vertical and the horizontal direction is as shown.

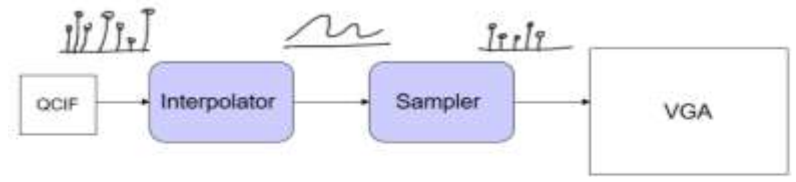
$$p(x', j) = (1 - x_f) \times p_{(i,j)} + x_f \times p_{(i+1,j)} \quad (1)$$

$$p(x', y') = [(1 - x_f) \times p_{(i,j+1)} + x_f \times p_{(i+1,j)}] \times (1 - y_f) + [(1 - x_f) \times p_{(i,j+1)} + x_f \times p_{(i+1,j+1)}] \times y_f$$

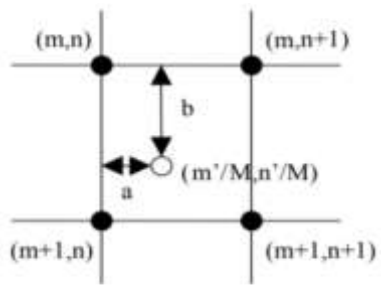
Where xf is the scale parameter in the horizontal direction and yf is the scale parameter in the vertical direction. Bilinear interpolation is a popular implementation in the VLSI chips.In this paper, we propose a novel bicubic method for digital image scaling. Image scaling is a prime technique in image processing. It is used in many important applications such as digital high-definition television, big screen display, copy and print machine, medical imaging, end-user equipment and so on. Bilinear scaling is the cheapest implementation of the Scaler that uses bilinear interpolation to calculate pixels. Bilinear interpolation produces a greater number of interpolation artifacts (such as aliasing, blurring, and edge halos) than more computationally demanding techniques such as bicubic interpolation.

Bicubic scaling is more demanding compared to bilinear scaling, and produces smoother pictures with less artifacts. Compared to bilinear interpolation, which only takes 2 x 2 pixels into account, bicubic interpolation considers a 4 x 4 pixel area.Bilinear method uses the Bilinear scaling which uses bilinear interpolation to calculate pixels. Bicubic method uses the Bicubic scaling which is a little bit more demanding compared to bilinear scaling, and produces smoother pictures with less artifacts.Bicubic is an extension of cubic interpolation for interpolating data points on a two dimensional grid. It can be accomplished using either Lagrange Polynomials or Cubic Convolution algorithm and it is chosen over Nearest Neighbour and Bilinear interpolation as its interpolated surface is smoother than both of them and has fewer interpolation artifacts.

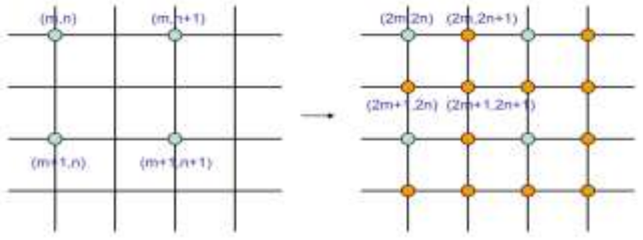
$$b_{-1} = p(t_x, a_{(-1,-1)}, a_{(0,-1)}, a_{(1,-1)}, a_{(2,-1)})$$
$$b_1 = p(t_x, a_{(-1,1)}, a_{(0,1)}, a_{(1,1)}, a_{(2,1)})$$
$$b_2 = p(t_x, a_{(-1,2)}, a_{(0,2)}, a_{(1,2)}, a_{(2,2)})$$
$$p(x,y) = p(t_y, b_{-1}, b_0, b_1, b_2)$$



2.1. Nearest Neighbor method



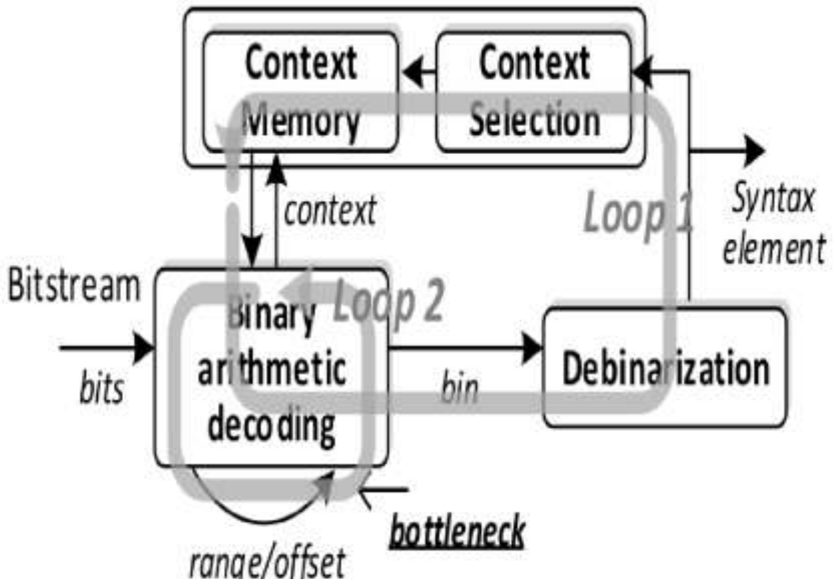
O[m’,n’] (the resized image) takes the value of the sample nearest to (m’/M,n’/M) in I[m,n] (the original image): O[m' n'] = I[(int) (m + 0 5) (int) (n + 0 5)] m = m'/M n = n' /M



Nearest Neighbor:
O[2m,2n]=I[m,n]
O[2m,2n+1]= I[m,n]
O[2m+1,2n]= I[m,n]
O[2m+1,2n+1]= I[m,n]

Next generation video codecs are expected to support the emerging 8K ultra-high-definition (UHD) format that involves up to 10–12 b/pixel, 7680 × 4320 pixels/frame and 120 frames/s, which corresponds to a bit rate by at least an order of magnitude higher than today’s mainstream 1080p HD. Context adaptive binary arithmetic coding (CABAC) is the entropy coding tool in the latest video coding standards: H.265/High Efficiency Video Coding (HEVC) and H.264/Advanced Video Coding (AVC). While CABAC delivers remarkably better coding efficiency than its predecessors, its algorithm suffers from critical data dependencies. In hardware implementation, this results in difficulties in leveraging parallelism and pipelining. The data dependencies, along with the requirement to process ultra-high bit rate in real time, make CABAC decoding a major bottleneck in the video decoder.

Figure shows the key components of CABAC decoding including binary arithmetic decoding (AD), context memory (CM), context selection, and debinarization. In the corresponding data flow, two loops lie outside and inside AD, respectively. This also makes CABAC decoding more challenging than encoding that contains only one critical loop.

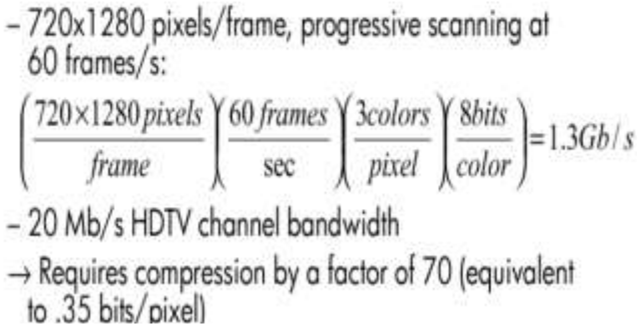


The first loop in CABAC decoding involves data dependencies between consecutive syntax elements (SEs). Many previous works have been conducted to alleviate its influence, by techniques including speculative/predictive execution of the pipeline, exhaustive coverage of branches, and reducing the frequency of SE switching. Moreover, in HEVC, the SE loop has also been alleviated at the algorithm level through reduction of context-coded bins. In comparison, the loop inside AD is much more critical by involving data dependencies at the binary symbol (bin) level.

To address this loop, Zhao et al., Yu and He, and Choi and Choi applied multi symbol AD designs to process more than 1 bin per clock cycle (BPCC). These multi-BPCC designs, however, suffered from an increase of critical path delay with increased parallelism. As a result, improvement in overall performance, in terms of the product of BPCC and maximum clock frequency, is limited. Moreover, a multi-BPCC CABAC decoder has to read/write multiple context models and may have to process multiple SEs in each clock cycle, which increases difficulty in implementation. Chen and Sze proposed a deeply pipelined 1-BPCC CABAC decoder. Despite being optimized with a subinterval reordering technique, AD is still the bottleneck of the whole CABAC decoder. Sze's architecture achieves above 3 Gbin/s throughput with the same AD architecture of and data parallelism at the SE level, but latter is not fully compatible to the final HEVC standard. For future video coding frameworks, the proposed AD design has the potential to be combined with SE-level parallelism for faster CABAC decoding.

We further explore the performance limit of AD, with a variable-clock-cycle-path (VCCP) design that exploits the differences in critical path delay and in probability of occurrence between various types of bins. The resulting AD throughput reaches 1010 Mbins/s in 90-nm CMOS, with a BPCC of 0.96 and a maximum clock frequency of 1053 MHz, which outperforms prior art by at least 19.1%.Binary AD transforms an input stream of bits into a sequence of bins, including regular bins and bypass bins. A regular bin can be either a more probable symbol

(MPS) or a less probable symbol (LPS). A terminate bin that signals the completion of an entire slice can be regarded as a special regular bin. Raw video contains an immense amount of data. Communication and storage capabilities are limited and expensive. Example HDTV video signal is shown below:



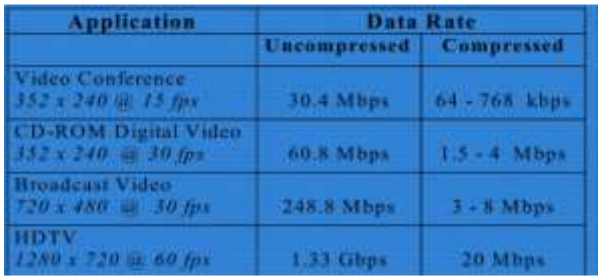
– 720x1280 pixels/frame, progressive scanning at 60 frames/s:

$$\left(\frac{720\times1280\,pixels}{frame}\right)\left(\frac{60\,frames}{\sec}\right)\left(\frac{3colors}{pixel}\right)\left(\frac{8bits}{color}\right)=1.3Gb/s$$

– 20 Mb/s HDTV channel bandwidth

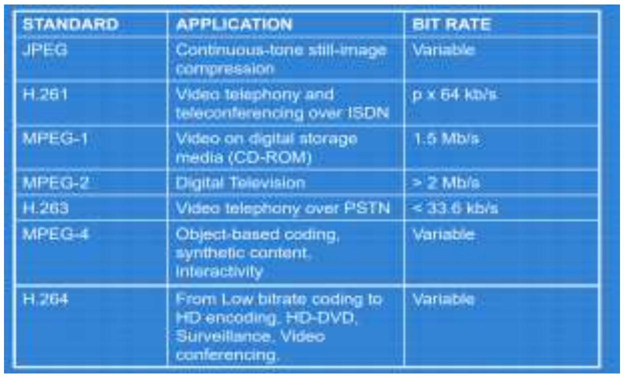
→ Requires compression by a factor of 70 (equivalent to .35 bits/pixel)

Bandwidth Reduction is as shown below, we can see the Application and data rate for both the uncompressed data and compressed data:



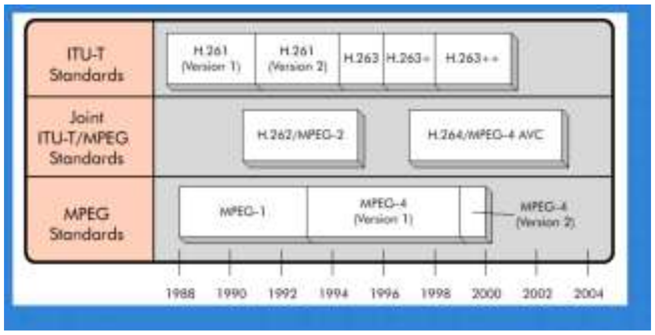
Application	Data Rate	
	Uncompressed	Compressed
Video Conference *352 x 240 @ 15 fps*	30.4 Mbps	64 - 768 kbps
CD-ROM Digital Video *352 x 240 @ 30 fps*	60.8 Mbps	1.5 - 4 Mbps
Broadcast Video *720 x 480 @ 30 fps*	248.8 Mbps	3 - 8 Mbps
HDTV *1280 x 720 @ 60 fps*	1.33 Gbps	20 Mbps

Video Compression standards are shown below, we can see the standard, application and the bit rate:

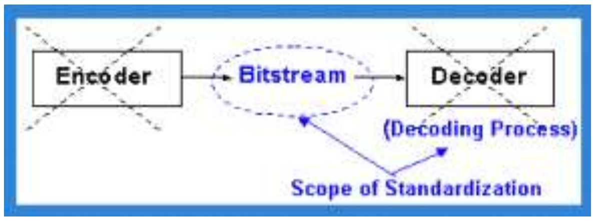


STANDARD	APPLICATION	BIT RATE
JPEG	Continuous-tone still-image compression	Variable
H.261	Video telephony and teleconferencing over ISDN	p x 64 kb/s
MPEG-1	Video on digital storage media (CD-ROM)	1.5 Mb/s
MPEG-2	Digital Television	> 2 Mb/s
H.263	Video telephony over PSTN	< 33.6 kb/s
MPEG-4	Object-based coding, synthetic content, interactivity	Variable
H.264	From Low bitrate coding to HD encoding, HD-DVD, Surveillance, Video conferencing.	Variable

Motivation of compression is Ensuring interoperability, Enabling communication between devices made by different manufacturers, Promoting a technology or industry, and Reducing costs.



Some more problems we need to consider are What Do the Standards Specify?A video compression system consists of the following: An encoder, Compressed bit -streams, A decoder, What parts of the system do the standards specify?Not the encoder, not the decoder:



Just the bit -stream syntax and the decoding process, for example it tells to use IDCT, but not how to implement the IDCT.

Enables improved encoding and decoding strategies to be employed in a standard -compatible manner.

Achieving Compression: Reduce redundancy and irrelevancy.

Sources of redundancy: *Temporal – Adjacent frames highly correlated.*
Spatial – Nearby pixels are often correlated with each other.
Color space – RGB components are correlated among themselves.
Irrelevancy – Perceptually unimportant information.
Basic Video Compression Architecture:
Exploiting the redundancies
Temporal – MC -prediction and MC -interpolation
Spatial – Block DCT
Color – Color space conversion Scalar quantization of DCT coefficients
Run -length and Huffman coding of the non -zero quantized DCT coefficients

The increasing demand to incorporate video data into telecommunications services, the corporate environment, the entertainment industry, and even at home has made digital video technology a necessity. A problem, however, is that still image and digital video data rates are very large, typically in the range of 150Mbits/sec. Data rates of this magnitude would consume a lot of the bandwidth, storage and computing resources in the typical personal computer. For this reason, Video Compression standards have been developed to eliminate picture redundancy, allowing video information to be transmitted and stored in a compact and efficient manner.The above figure shows how an input video signal is converted to an output bitstream after several processing techniques like color space conversion i.e. RGB to YUV, DCT, quantization, Huffman coding,etc.

3. Software Analysis

Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language. As similar IDEs, it offers to the programmer a simple and unified tool to edit, compile, link, and debug programs. It also provides support for the management of the files of a program in "papers" containing all the elements required to produce a final executable program. Dev-C++ uses the Mingw port of GCC (GNU Compiler Collection) as a compiler. It can create native Win32 executables, either console or GUI, as well as DLLs and static libraries. Dev-C++ can also be used in combination with Cygwin or any other GCC based compilerDev-C++ is a Free Software distributed under the terms of the GNU General Public License (GPL).

Dev-C++ can be installed on any Windows machine with Windows XP and Windows 7. This tutorial uses Dev-C++ 4.9.9.2 on Windows 7 (configuration in the computer labs as of course 2012-2013).

3.1. Application Development:

The application development process encompasses the following steps:

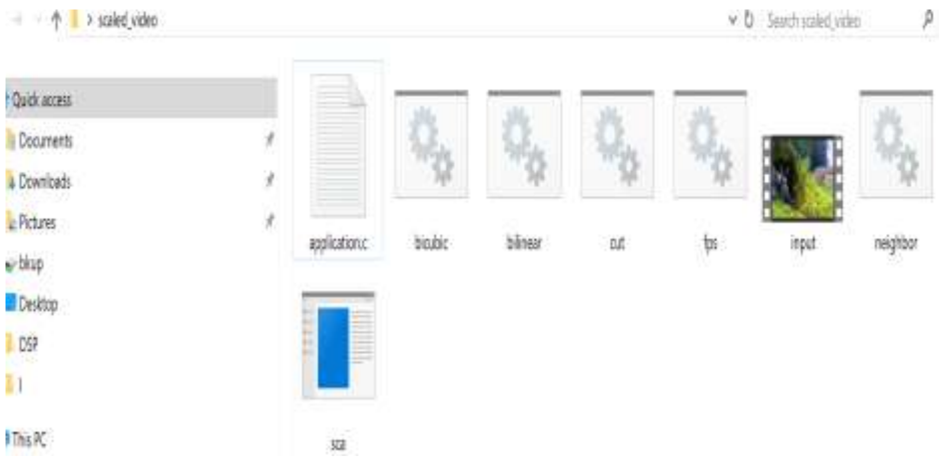
1. Create a paper: The type of application and the programming language to be used are specified.
2. Write source code: Write the program in C and save the source code file.
3. Compile and link the code: The source code is compiled and linked to generate a running program. Other files of the paper may be created.
4. Fix compilation errors: If the syntax of the program is not correct, the compilation fails and the compiler generates a message related to the error/s. The programmer must correct the errors.
5. Run the program: Run the program to validate the functioning.
6. Fix execution errors: If the actions performed by the program are not as expected, it is necessary to correct the source code. It may be also convenient to use the debugger to find complex errors.

3.2. Paper Creation:

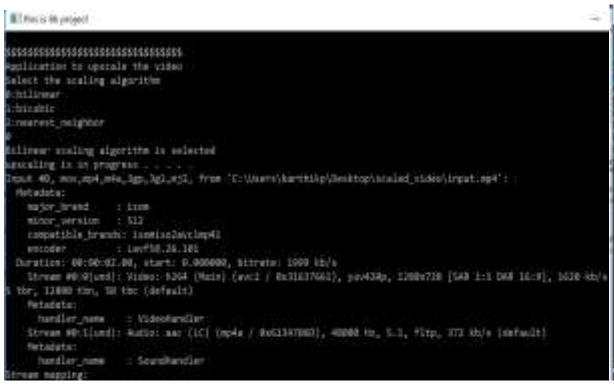
A paper is a center for managing your different source files and options inside Dev-C++. It helps you navigate through your code, and easily set different parameters, like the type of program you are doing (GUI, console, DLL ...).A paper groups several files with a common purpose. In our programs, papers will include a file with metadata about the paper (.dev), a file with C source code (.c), a file with object code (.o), and a file with linking instructions (makefile.win). Only the ".c" file must be explicitly created –the remaining files are automatically created by Dev-C++. When creating a paper, Dev-C++ asks the user where the files must be stored. It is convenient to use different folders for each paper, since by default, the source code is named main.c and previous files can be overwritten. Paper type determines the type of application that will be developed. User interaction is performed by typing information on the keyboard (input) and printing characters on the screen (output) with proper read and write instructions.

4. Results

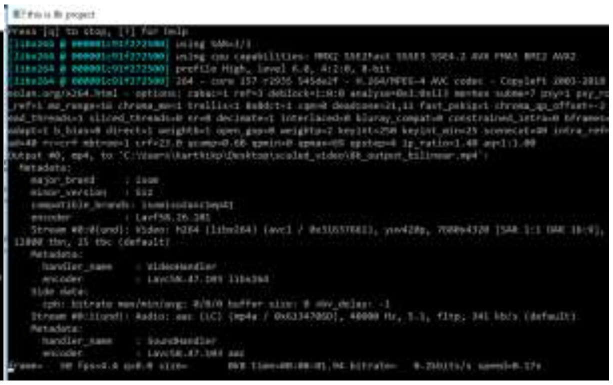
Before running the application the directory window is like this:



User can select any scaler algorithm, for example select 0

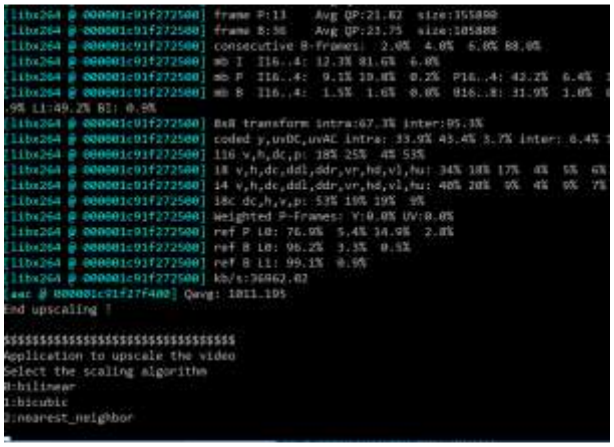


```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Application to upscale the video
Select the scaling algorithm
0:bilinear
1:bicubic
2:nearest_neighbor
0
Bilinear scaling algorithm is selected
upscaling is in progress . . . .
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'C:\Users\karthikp\Desktop\scaled_video\input.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.26.101
  Duration: 00:00:02.00, start: 0.000000, bitrate: 1999 kb/s
    Stream #0:0(und): Video: h264 (Main) (avc1 / 0x31637661), yuv420p, 1280x720 [SAR 1:1 DAR 16:9], 1620 kb/s
s tbr, 12800 tbn, 50 tbc (default)
    Metadata:
      handler_name    : VideoHandler
    Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, 5.1, fltp, 372 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
Stream mapping:
```



```
Press [q] to stop, [?] for help
[libx264 @ 000001c91f272500] using SAR=1/1
[libx264 @ 000001c91f272500] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
[libx264 @ 000001c91f272500] profile High, level 6.0, 4:2:0, 8-bit
[libx264 @ 000001c91f272500] 264 - core 157 r2935 545de2f - H.264/MPEG-4 AVC codec - Copyleft 2003-2018
videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd
_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2
ead_threads=1 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes
adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=250 keyint_min=25 scenecut=40 intra_refr
ad=40 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'C:\Users\karthikp\Desktop\scaled_video\8k_output_bilinear.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.26.101
    Stream #0:0(und): Video: h264 (libx264) (avc1 / 0x31637661), yuv420p, 7680x4320 [SAR 1:1 DAR 16:9],
 12800 tbn, 25 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc58.47.103 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
    Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, 5.1, fltp, 341 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
      encoder         : Lavc58.47.103 aac
frame=   30 fps=4.4 q=0.0 size=       0kB time=00:00:01.96 bitrate=   0.2kbits/s speed=0.17x
```

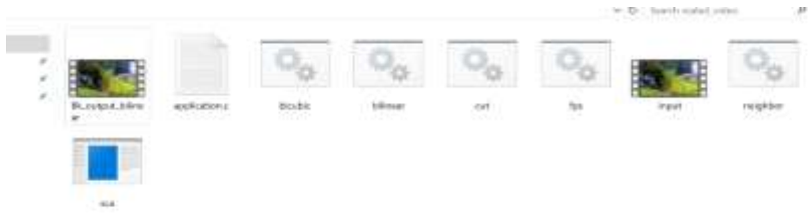
After the scaling is done, the menu repeats:



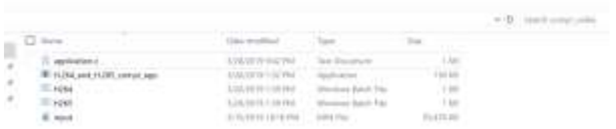
```
[libx264 @ 000001c91f272500] frame P:13    Avg QP:21.02  size:355080
[libx264 @ 000001c91f272500] frame B:36    Avg QP:23.75  size:105888
[libx264 @ 000001c91f272500] consecutive B-frames:  2.0%  4.0%  6.0% 88.0%
[libx264 @ 000001c91f272500] mb I  I16..4: 12.3% 81.6%  6.0%
[libx264 @ 000001c91f272500] mb P  I16..4:  9.1% 10.8%  0.2%  P16..4: 42.2%  6.4%
[libx264 @ 000001c91f272500] mb B  I16..4:  1.5%  1.6%  0.0%  B16..8: 31.9%  1.0%
.9% L1:49.2% BI: 0.9%
[libx264 @ 000001c91f272500] 8x8 transform intra:67.3% inter:95.3%
[libx264 @ 000001c91f272500] coded y,uvDC,uvAC intra: 33.9% 43.4% 3.7% inter: 6.4%
[libx264 @ 000001c91f272500] i16 v,h,dc,p: 18% 25%  4% 53%
[libx264 @ 000001c91f272500] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 34% 18% 17%  4%  5%  6%
[libx264 @ 000001c91f272500] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 40% 20%  9%  4%  9%  7%
[libx264 @ 000001c91f272500] i8c dc,h,v,p: 53% 19% 19%  9%
[libx264 @ 000001c91f272500] Weighted P-Frames: Y:0.0% UV:0.0%
[libx264 @ 000001c91f272500] ref P L0: 76.9%  5.4% 14.9%  2.8%
[libx264 @ 000001c91f272500] ref B L0: 96.2%  3.3%  0.5%
[libx264 @ 000001c91f272500] ref B L1: 99.1%  0.9%
[libx264 @ 000001c91f272500] kb/s:36962.02
[aac @ 000001c91f27f400] Qavg: 1011.195
End upscaling !

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Application to upscale the video
Select the scaling algorithm
0:bilinear
1:bicubic
2:nearest_neighbor
```

After the scaling is done you can see a new video is generated in the same directory:



The results directory looks like this:



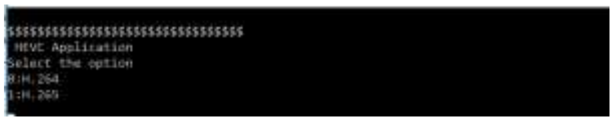
```
Name                        Date modified   Type                Size
application.c               [illegible]     Text Document       1 KB
H.264_and_H.265_compr_app   [illegible]     Application         [illegible]
H264                        [illegible]     Windows Batch File  1 KB
H265                        [illegible]     Windows Batch File  1 KB
input                       [illegible]     MP4 File            [illegible]
```

In the directory we have some files like application.c, H.264_and_H.265_compr_app, H264, H265, and input files.

H.264_and_H.265_compr_app is an application which has the compression algorithms like H264 and H265.

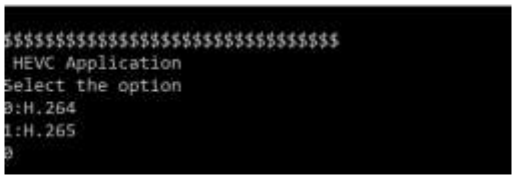
The directory has an input video which is to be compressed and 2 batch files named H264 and H265 which are used by the application.c to interact with the application specific interfaces or functions.

Open the application by double clicking the app exe file, after that you can see:



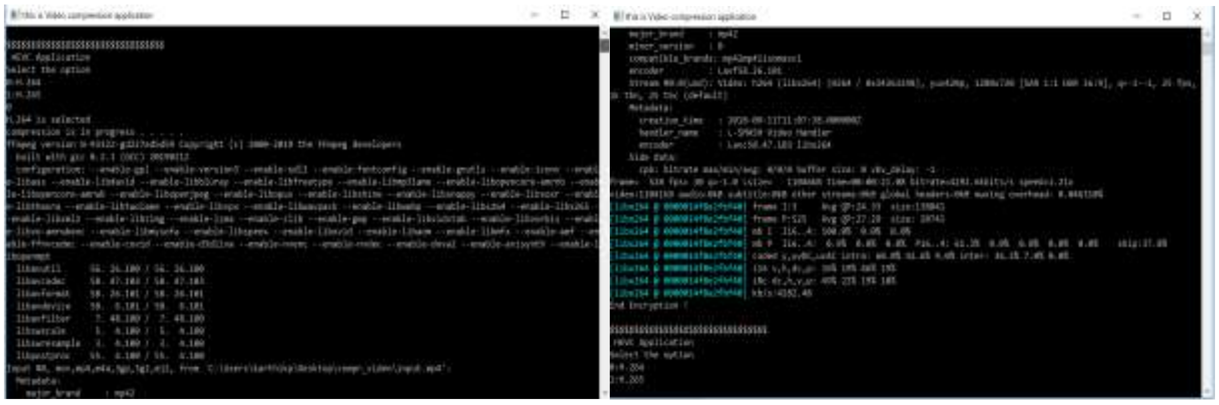
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 HEVC Application
Select the option
0:H.264
1:H.265
```

After that enter 0:



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 HEVC Application
Select the option
0:H.264
1:H.265
0
```

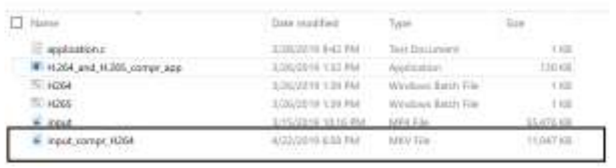
Press enter:



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 HEVC Application
Select the option
0:H.264
1:H.265
0
H.264 is selected
compression is in progress . . . . .
ffmpeg version [illegible] Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc [illegible]
  configuration: --enable-gpl --enable-version3 --enable-sdl2 --enable-fontconfig --enable-gnutls [illegible]
[illegible]
  libavutil      56. [illegible]
  libavcodec     58. [illegible]
  libavformat    58. [illegible]
  libavdevice    58. [illegible]
  libavfilter     7. [illegible]
  libswscale      5. [illegible]
  libswresample   3. [illegible]
  libpostproc    55. [illegible]
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'C:\Users\[illegible]\input.mp4':
  Metadata:
    major_brand     : mp42
```

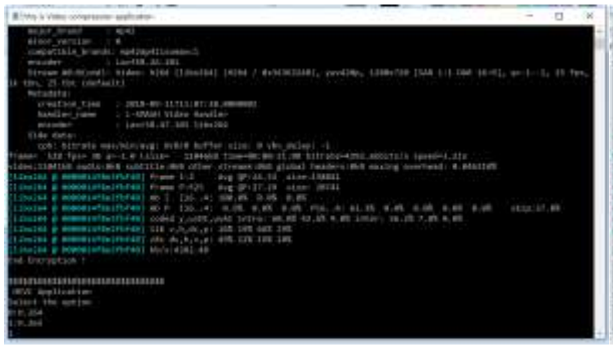
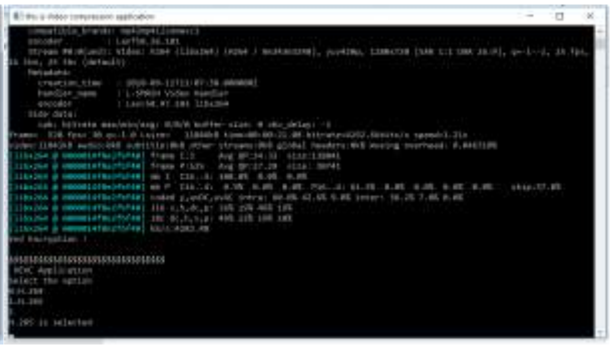
```
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: [illegible]
    encoder         : [illegible]
    Stream #0:0(und): Video: h264 (libx264) [illegible]
    Metadata:
      creation_time   : [illegible]
      handler_name    : [illegible]
      encoder         : [illegible]
    Side data:
      cpb: [illegible]
[illegible]
End Encryption !
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 HEVC Application
Select the option
0:H.264
1:H.265
```

Now see the results directory, a new file with name input_compr_H264 is generated:

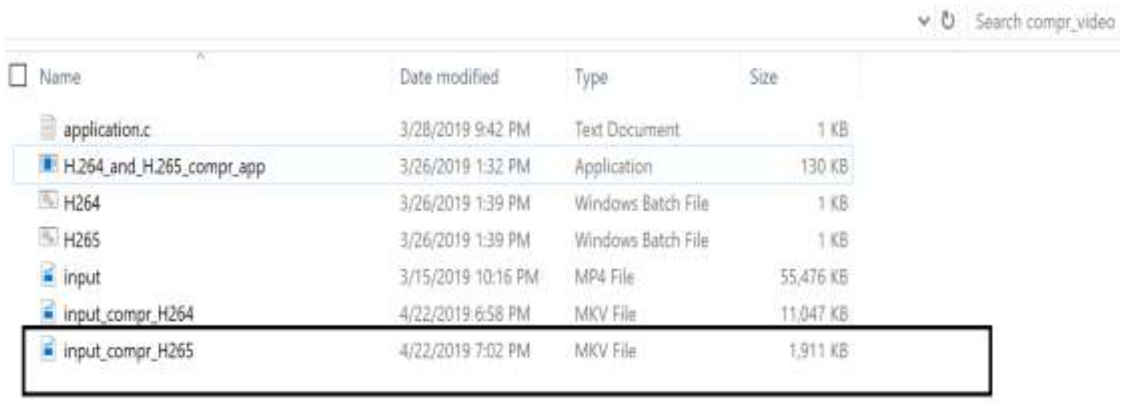


```
Name                        Date modified        Type                Size
application.c               3/26/2019 8:42 PM    Text Document       1 KB
H.264_and_H.265_compr_app   3/26/2019 1:32 PM    Application         130 KB
H264                        3/26/2019 1:39 PM    Windows Batch File  1 KB
H265                        3/26/2019 1:39 PM    Windows Batch File  1 KB
input                       3/15/2019 10:16 PM   MP4 File            55,476 KB
input_compr_H264            4/22/2019 6:33 PM    MKV File            11,047 KB
```

Next enter 1 in the app:

In the directory you can see that a new file by name input_compr_H265 is generated.



Name	Date modified	Type	Size
application.c	3/28/2019 9:42 PM	Text Document	1 KB
H.264_and_H.265_compr_app	3/26/2019 1:32 PM	Application	130 KB
H264	3/26/2019 1:39 PM	Windows Batch File	1 KB
H265	3/26/2019 1:39 PM	Windows Batch File	1 KB
input	3/15/2019 10:16 PM	MP4 File	55,476 KB
input_compr_H264	4/22/2019 6:58 PM	MKV File	11,047 KB
input_compr_H265	4/22/2019 7:02 PM	MKV File	1,911 KB

Algorithm	Input	Size
H264	55,476 KB	11,047 KB
H265	55,476 KB	1,911 KB*

Note: You can see that the compressed video has very less size.

Play the 2 compressed videos and the input video:

Input video:

(a) H264 video

(

b) H265 Video

4. Application Areas

Broadcast Displays, Cameras, Switchers, and Video Servers; LED Wall; Multi-Panel Displays; Digital Cinema; Projectors; Medical Endoscope; Video Surveillance; Consumer Displays; Video Conferencing; Machine Vision; Privacy and security: E-mail message, wireless network; Confidentiality: personal records, transaction records; Authentication: digital signatures, login; Intellectual property: copy protection.

For most use cases,Vormetric Transparent Encryption meets an enterprise's security, contractual and compliance requirements. Vormetric Transparent Encryption applies encryption and access policies "transparently" to data at rest, of any file type and in any environment, at the file or volume level without requiring any application development and maintenance efforts. However, for the applications that require field-level encryption fordatabase,big data,PaaS or other applications there is Vormetric Application Encryption. Vormetric Application Encryption is a library to simplify integrating application-level encryption into existing corporate applications. The application encryption library provides a set of documented standard-based APIs used to perform cryptographic and encryption key management operations. The innovative product design enables developers to choose to standard AES encryption or schema maintaining Format Preserving Encryption (FPE). Vormetric Application Encryption removes the complexity and risk of implementing an in-house encryption and key management solution. Vormetric Application Encryption supports Unicode and the library is in the process of being certified for FIPS 140-2.

4.1 Vormetric Application Encryption Key Attributes

A. Column-level encryption -Easily add standards based column-level encryption or schema maintaining Format Preserving encryption (FPE) to existing applications.

B. Protect sensitive data - Stop malicious DBAs, cloud administrators, hackers, and authorities with subpoenas from accessing valuable data.

C. Deploy with confidence - Leverage proven, Vormetric high-performance encryption and key management agents.

D. Maintain SLA - Experience high-performance encryption transactions per second.

E. Centralize control - Reduce complexity and costs associated with application-layer encryption and file system-level encryption.

F. Support heterogeneous environments - Simply extend application-layer encryption across virtual, cloud, big data and traditional environments that run Linux and Windows.

G. Format Preserving Encryption - The encrypted output field size remains identical to the input reducing storage footprint. Allows encryption within a specified character set. For example, encrypting a 16-digit credit card number will result in a 16-digit number.

H. Batch Data Transformation support - Vormetric Batch Data Transformation is a product that provides capabilities to mask, tokenize or encrypt sensitive column information in databases. It is used to quickly encrypt data in existing databases that are in use with applications that will be protecting sensitive data with Vormetric Application Encryption.

5. Conclusion

In this paper we have developed the source code for a system which reads a video and compresses it with various algorithms and validated it. This video compression application can be used as an IP in the FPGA, this IP can also be used as a peripheral in the embedded system. IP can also be a part of any processor-based system on chip or a cortex M3 processor-based system on chip. In some cases, the IP can be imported as a module and an embedded system can be built manually instead of using a SOC (system on chip). IP can handle all the masters and slaves connected to it on the chip or off the chip, some of the chip hardware components can be interfaced to this IP to meet the end customer requirements like video processing systems, cell phones, Image processing applications, Transmitters and Receivers ..etc. In order to explore some involving challenging theoretical and technical issues, we have developed a novel system design and algorithm analysis approach for FPGA platform-based real-time chaos-based video compression systems. Compared with digital simulations, the hardware implementation for real-time chaos-based video compression algorithms is much more difficult to achieve. The new system has been designed and analyzed to resolve various difficulties, and its corresponding hardware experiments have been verified and validated, demonstrating the feasibility of the proposed methodology.

6. References

[1] IDC. (Dec. 2012). “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East.” [Online]. Available: https://www.emc.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf

[2] N. Gong, S. Jiang, A. Challapalli, S. Fernandes, and R. Sridhar, “Ultra-low voltage split-data-aware embedded SRAM for mobile video applications,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 59, no. 12, pp. 883–887, Dec. 2012.

[3] M. E. Sinangil and A. P. Chandrakasan, “Application-specific SRAM design using output prediction to reduce bit-line switching activity and statistically gated sense amplifiers for up to 1.9× lower energy/access,” IEEE J. Solid-State Circuits, vol. 49, no. 1, pp. 107–117, Jan. 2014.

[4] C. Liu and M. D. Fairchild, “Measuring the relationship between perceived image contrast and surround illumination,” in Proc. IS TSID 12th Color Imag. Conf., 2004, pp. 282–288.

[5] Z. Wei and K. N. Ngan, “A temporal just-noticeble distortion profile for video in DCT domain,” in Proc. 15th IEEE Int. Conf. Image Process.,Oct. 2008, pp. 1336–1339.

[6] J. Xue and C. W. Chen, “Mobile video perception: New insights and adaptation strategies,” IEEE J. Sel. Topics Signal Process., vol. 8, no. 3,pp. 390–401, Jun. 2014.

[7] C.-H. Lo and S.-Y. Huang, “P-P-N based 10T SRAM cell for low-leakage and resilient subthreshold operation,” IEEE J. Solid-State Circuits, vol. 46, no. 3, pp. 695–704, Mar. 2011.

[8] K. Kushida et al., “A 0.7 V single-supply SRAM with 0.495 μm2 cell in 65 nm technology utilizing self-write-back sense amplifier and cascaded bit line scheme,” IEEE J. Solid-State Circuits, vol. 44, no. 4, pp. 1192–1198, Apr. 2009.

[9] Manchala Sreeja, Vallabhuni Vijay, “A Unique Approach To Provide Security For Women By Using Smart Device,” European Journal of Molecular & Clinical Medicine, vol. 7, iss. 1, 2020, pp. 3669-3683.

[10] V. Vijay, and Avireni Srinivasulu, “Grounded Resistor and Capacitor based Square Wave Generator using CMOS DCCII,” in proceedings of the 2016 IEEE International Conference on Inventive Computation Technologies (IEEE ICICT-2016), Coimbatore, India, August 26-27, 2016, pp. 79–82.

[11] V. Siva Nagaraju, P. Ashok Babu, Vallabhuni Rajeev Ratna, Ramya Mariserla, “Design and Implementation of Low Power 32-bit Comparator,” Proceedings of the International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS 2020), Palai, India, December 10-11, 2020, pp. 1-8.

[12] B.M.S Rani, Divyasree Mikkili, Rajeev Ratna Vallabhuni, Chandra Shaker Pittala, Vijay Vallabhuni, Suneetha Bobbillapati, H. Bhavani Naga Prasanna, "Retinal Vascular Disease Detection from Retinal Fundus Images Using Machine Learning," Australia patent 2020101450.

[13] Venkateswarlu, S.C., Kumar, N.U., Kumar, N.S., Karthik, A., and Vijay, V., "Implementation of Area optimized Low power Multiplication and Accumulation," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 9, iss. 9, 2019, pp. 2278–3075.

[14] Vallabhuni Rajeev Ratna, M. Saritha, Saipreethi. N, V. Vijay, P. Chandra Shaker, M. Divya, Shaik Sadulla, "High Speed Energy Efficient Multiplier Using 20nm FinFET Technology," Proceedings of the International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS 2020), Palai, India, December 10-11, 2020, pp. 1-8.

[15] V. Vijay, and Avireni Srinivasulu, "Tunable Resistor and Grounded Capacitor Based Square Wave Generator Using CMOS DCCII," International J. Control Theory and Applications, vol. 8, 2015, pp. 1–11.

[16] Rajeev Ratna Vallabhuni, K.C. Koteswaramma, B. Sadgurbabu, and Gowthamireddy A, "Comparative Validation of SRAM Cells Designed using 18nm FinFET for Memory Storing Applications," Proceedings of the 2nd International Conference on IoT, Social, Mobile, Analytics & Cloud in Computational Vision & Bio-Engineering (ISMAC-CVB 2020), 2020, pp. 1-10.

[17] Vallabhuni Vijay, and Avireni Srinivasulu, "A low power waveform generator using DCCII with grounded capacitor," International Journal of Public Sector Performance Management, vol. 5, 2019, pp. 134–145.

[18] Vallabhuni Vijay, C. V. Sai Kumar Reddy, Chandrashaker Pittala, "System and Method to Improve Performance of Amplifiers Using Bias Current," The Patent Office Journal No. 43/2019, India. International classification: C12Q1/6869. Application No. 201941042648 A.

[19] V. Vijay, and Avireni Srinivasulu, "A square wave generator using single CMOS DCCII," in proceedings of the 2013 IEEE International SoC Design Conference (IEEE ISoCC-2013), Busan, South Korea, November 17-19, 2013, pp. 322–325.

[20] Babu, P. Ashok, et al.,"Realization of 8 x 4 Barrel shifter with 4-bit binary to Gray converter using FinFET for Low Power Digital Applications," Journal of Physics: Conference Series. Vol. 1714. No. 1. IOP Publishing, 2021.

[21] Rajeev Ratna Vallabhuni, J. Sravana, M. Saikumar, M. Sai Sriharsha, and D. Roja Rani, "An advanced computing architecture for binary to thermometer decoder using 18nm FinFET," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20-22 August, 2020, pp. 510–515.

[22] Vallabhuni Vijay, V. Siva Nagaraju, M. Sai Greeshma, B. Revanth Reddy, U. Suresh Kumar, and, C. Surekha, "A Simple and Enhanced Low-Light Image Enhancement Process Using Effective Illumination Mapping Approach," Lecture Notes in Computational Vision and Biomechanics, Cham, Switzerland. 2019, pp. 975–984.

[23] Kurra, A.K., and Sadulla, S., "Analysis of physical unclonable functions (PUFS) for secure key generation on smartcard chips," Journal of Advanced Research in Dynamical and Control Systems, 9, 2017, pp. 1735-1745.

[24] Vallabhuni Vijay, “Second Generation Differential Current Conveyor (DCCII) And Its Applications,” Vignan's Foundation for Science, Technology & Research (Deemed to be University), Guntur, 2017.

[25] Shaik, Sadulla, Anil Kumar Kurra, and A. Surendar, “High secure buffer based physical unclonable functions (PUF's) for device authentication,” Telkomnika, 17, no. 1, 2019.

[26] Rajeev Ratna Vallabhuni, G. Yamini, T. Vinitha, and S. Sanath Reddy, “Performance analysis: D-Latch modules designed using 18nm FinFET Technology,” 2020 International Conference on Smart Electronics and Communication (ICOSEC), Tholurpatti, India, 10-12, September 2020, pp. 1171–1176.

[27] Vallabhuni Vijay, and Avireni Srinivasulu, “A Novel Square Wave Generator Using Second Generation Differential Current Conveyor,” Arabian Journal for Science and Engineering, vol. 42, iss. 12, 2017, pp. 4983–4990.

[28] V. Vijay, J. Prathiba, S. Niranjan reddy, Ch. Srivalli, and B. Subbarami reddy, “Performance evaluation of the CMOS Full adders in TDK 90 nm Technology,” International Journal of Systems, Algorithms & Applications, vol. 2, iss. 1, 2012, pp. 711.

[29] Vallabhuni Vijay, P. Chandra Shekar, V. Siva Nagaraju, S. China Venkateswarlu, and Shaik Sadulla, “High Performance 2:1, 4:1 And 8:1 Binary And Ternary Multiplexer Realization Using CNTFET Technology,” Journal of Critical Reviews, vol. 7, iss. 6, 2020, pp. 1159–1163.

[30] Kurra, Anil Kumar, and Usha Rani Nelakuditi, “A secure arbiter physical unclonable functions (PUFs) for device authentication and identification,” Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 7, no. 1, 2019, pp. 117–127.

[31] Ratna, Vallabhuni Rajeev and M, Saritha and N, Saipreethi and V, Vijay and Shaker, P. Chandra and M, Divya and Sadulla, Shaik, High Speed Energy Efficient Multiplier Using 20nm FinFET Technology (January 19, 2021). ICICNIS 2020, Available at SSRN: https://ssrn.com/abstract=3769235 or http://dx.doi.org/10.2139/ssrn.3769235

[32] Vallabhuni Vijay, Pittala Chandra shekar, Shaik Sadulla, Putta Manoja, Rallabhandy Abhinaya, Merugu rachana, Nakka nikhil, “Design and performance evaluation of energy efficient 8-bit ALU at ultra low supply voltages using FinFET with 20nm Technology,” VLSI Architecture for Signal, Speech, and Image Processing, edited by Durgesh Nandan, Basant Kumar Mohanty, Sanjeev Kumar, Rajeev Kumar Arya, CRC press, 2021.

[33] Rajeev Ratna Vallabhuni, Jujavarapu Sravana, Chandra Shaker Pittala, Mikkili Divya, B.M.S.Rani, Vallabhuni Vijay, “Universal Shift Register Designed at Low Supply Voltages in 20nm FinFET Using Multiplexer,” Lecture Notes in Networks and Systems, 2021.

[34] V, Siva Nagaraju and P, Ashok Babu and Ratna, Vallabhuni Rajeev and Mariserla, Ramya, “Design and Implementation of Low Power 32-bit Comparator (January 20, 2021). ICICNIS 2020, Available at SSRN: https://ssrn.com/abstract=3769748 or http://dx.doi.org/10.2139/ssrn.3769748

[35] P. Saritha, J. Vinitha, S. Sravya, V. Vijay, and E. Mahesh, “4-Bit Vedic Multiplier with 18nm FinFET Technology,” 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 1079–1084.

[36] Vijay, V., J. Prathiba, S. Niranjan Reddy, and P. Praveen Kumar, “A REVIEW OF THE 0.09 µm STANDARD FULL ADDERS,” International Journal of VLSI Design & Communication Systems, vol. 3, no. 3, 2012, p. 119.

[37] Kurra, Anil, and Usha Rani Nelakuditi, "Design of a Reliable Current Starved Inverter Based Arbiter Physical Unclonable Functions (PUFs) for Hardware Cryptography," Ingénierie des Systèmes d Inf., 24, no. 4, 2019, pp. 445–454.

[38] P. Chandra Shaker, V. Parameswaran, M. Srikanth, V. Vijay, V. Siva Nagaraju, S.C. Venkateswarlu, Sadulla Shaik, and Vallabhuni Rajeev Ratna, "Realization and Comparative analysis of Thermometer code based 4-Bit Encoder using 18nm FinFET Technology for Analog to Digital Converters," Advanced Intelligent Systems and Computing (AISC), 2020.

[39] Mohammad Khadir, Kancharapu Chaitanya, S. Sushma, V. Preethi, and Vallabhuni Vijay, "Design Of Carry Select Adder Based On A Compact Carry Look Ahead Unit Using 18nm Finfet Technology," Journal of Critical Reviews, vol. 7, iss. 6, 2020, pp. 1164–1171.

[40] Rajeev Ratna Vallabhuni, D.V.L. Sravya, M. Sree Shalini, and G. Uma Maheshwararao, "Design of Comparator using 18nm FinFET Technology for Analog to Digital Converters," 2020 7th International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 23-24 july, 2020, pp. 318–323.

[41] Rajeev Ratna Vallabhuni, S. Lakshmanachari, G. Avanthi, and Vallabhuni Vijay, "Smart Cart Shopping System with an RFID Interface for Human Assistance," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, December 4-5, 2020, pp. 165-169.

[42] K. Nagalakshmi, Avireni Srinivasulu, Cristian Ravariu, V. Vijay, V. V. Krishna, "A novel simple schmitt trigger circuit using CDTA and its application as a square-triangular waveform generator," J. Mod. Technol. Eng, vol. 3, 2018, pp. 205-216.

[43] Vallabhuni Vijay, C. V. Sai Kumar Reddy, Veerastu Sivanagaraju, Chandrashaker Pittala, "System for Minimizing Crosstalk Effects of Shells and Designing Multiwalled Carbon Nanotube Models," The Patent Office Journal No. 43/2019, India. International classification: B82Y10/00. Application No. 201941042460 A.

[44] V. Vijay, and Avireni Srinivasulu, "A DCCII Based Square Wave Generator With Grounded Capacitor," in proceedings of the 2016 IEEE International Conference on Circuits, Power and Computing Technologies (IEEE ICCPCT-2016), Kumaracoil, India, March 18-19, 2016, pp. 1–4.

[45] Vallabhuni Vijay, C. V. Sai Kumar Reddy, Chandrashaker Pittala, Sonagiri China Venkateswarlu "System for Reducing Crosstalk Delays In Electronic Devices Using A CMOS Inverter," The Patent Office Journal No. 43/2019, India. International classification: H03B5/18. Application No. 201941042515 A.

[46] Shaik, Sadulla, and Prathiba Jonnala, "Performance evaluation of different SRAM topologies using 180, 90 and 45 nm technology,"In 2013 International Conference on Renewable Energy and Sustainable Energy (ICRESE), pp. 15-20. IEEE, 2013.

[47] Rajeev Ratna Vallabhuni, A. Karthik, CH. V. Sai Kumar, B. Varun, P. Veerendra, and Srisailam Nayak, "Comparative Analysis of 8-Bit Manchester Carry Chain Adder Using FinFET at 18nm Technology," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, December 4-5, 2020, pp. 1579-1583.

[48] Vallabhuni Vijay, C. V. Sai Kumar Reddy, Chandrashaker Pittala, P ASHOK BABU, "System to Obtain Finite Gain and Noise of an Electrocardiogram Amplifier," The Patent Office Journal No. 43/2019, India. International classification: H03F3/38. Application No. 201941042674 A.

[49] Rajeev Ratna Vallabhuni, P Shruthi, G Kavya, S Siri Chandana, “6Transistor SRAM Cell designed using 18nm FinFET Technology,” 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2020, pp. 1584-1589.