Western Norway
University of
Applied Sciences

BACHELOR THESIS:

H.264 VIDEO COMPRESSING ON FPGA

Ngoc Khiem Doan
Department of Electrical Engineering

31. May. 2019

Document Control

Report title H.264 Video Compressing on FPGA	*Date/Version* 31. May. 2019/0.16
	Report number: BO19E-11
Author(s): Ngoc Khiem Doan	*Course:* HEEL16
	Number of pages including appendixes 27
Supervisor at Western Norway University of Applied Sciences Eivind Vågslid Skjæveland	*Security classification:* Open
Comments: We, the authors, allow publishing of the report.	

Contracting entity: Archer BTC	*Contracting entity's reference:* -
Contact(s) at contracting entity, including contact information: *Tarjei Rommetveit* *Tarjei.rommetveit@archerwell.com*	

Revision	**Date**	**Status**	**Performed by**
0.11	02.05.2019	First issue	Ngoc Khiem Doan
0.12	25.05.2019	Revised issue	Ngoc Khiem Doan
0.13	30.05.2019	Corrected index and numbering	Ngoc Khiem Doan

Preface

Many years ago, there was a stream of refugees from Vietnam to every country all over the world. Norway, a far north, cold and peace kingdom opened its arms to my people, including my aunt's little family. After arriving here, my uncle and aunt have worked very hard for their living, and in some way to repay this beloved country for helping them when they were most desperate. Four years ago, they used most of the money they had saved up helping me and my cousins take our bachelor's degree here. Just before we started our first school days, my uncle said to us: "Study good, do your best and help me repay Norwegians for what they have done to us.". So now I am here, finishing my bachelor's degree and fulfilling my uncle's dream.

Firstly, I would like to express my honor to be able to work in this project. As an engineering student, I understand how a good system is designed. It's a long progress of designing, testing and upgrading. I hope my little work could be a part of big improvements, which is the goal of engineering: Always improve and never settle.

I would like to thank my tutors: Eivind and Øyvind. I know that it takes a lot of patience to work with me – a foreign student who can barely speak Norwegian. Thank you very much for your help, your advices and all.

To mom and dad. Thank you for everything. To have a son who takes two bachelor's degrees in a row is surely not something you had ever thought about, but I have finished anyway.

To Phuong, my beloved soulmate, my best friend, my supporter. Thank you for standing me all the time, your support whenever I failed, your love and your delicious dishes.

To uncle Cuong, aunt Tuyet and my cousins, Huy Anh and Huy Em. Thank you for having me here. Thank you for treating me like your son and brother. I myself always think of you as my second family here in Norway.

Khiem Doan

Summary

At the beginning of this project, the goals were set quite ambitious. The currently model of SPACE system from Archer BTC had already been a very high performance one. But realizing the amount of received data and the speed information being transferred are not match to each other, Archer decided to study on a new data compressing algorithm that can utilize so much data from the sensor that possible.

The project is divided into three main stages:

1. Finding good algorithm.
2. Realizing the algorithm into real system.
3. Testing the new system under both normal and extreme conditions.

This project was supposed to be done by a group of 3-4 people, but since there was only one person that is interested in, the progression became much slower than expected.

To find a good algorithm for data compressing, we have considered many alternatives from very simple to top complicated. These algorithms highly focus on image and video processing since the data from SPACE sensors will lately be read under visual form. This will be discussed further under Chapter 4 – Realization of selected solution.

After that we discussed and chose a good hardware solution for the chosen algorithm. That hardware solution must justify many constrains that were set: physical size, hardware performance and memory size for the solution. It is quite clear that the algorithm needs to be “programmed” on a DSP, microcontroller or FPGA, which takes much effort and time in the project’s schedule.

The finished system is then simulated on simulating application, and thereafter tested under real life’s conditions. Any effect from change in conditions (pressure, humidity, temperature…) on system is recorded so we can determine if the hardware can bear extreme conditions where SPACE usually works in.

1 Index

Document Control 2
Preface 3
Summary 4
2 Table of figures 6
1 Introduction 7
1.1 About Archer BTC and SPACE system 7
1.1.1 Archer BTC 7
1.1.2 SPACE system 7
1.2 Problem description 8
1.3 Solution discussion 8
2 Specification of requirements 10
2.1.1 Data transferring 10
2.1.2 Extreme conditions 10
2.1.3 Size of the package 10
3 Problem analysis 11
3.1 Description of possible solutions 11
3.1.1 Edge detection 11
3.1.2 Video processing 12
3.1.3 Issues regarding tools and HW/SW components 12
3.2 Problem analysis conclusion 13
4 Realization of selected solution 14
4.1 Theory 14
4.1.1 Video processing fundamentals 14
4.1.2 H.264 encoding algorithm 16
4.2 Program implementation 16
4.2.1 Hardh264 IP 16
4.2.2 Adapting to Smartfusion2 on Libero 17
4.2.3 Video exporting 19
5 Testing 22
6 Discussion 24
7 Conclusion 25

2 Table of figures

Figure 1. SPACE Panorama 7
Figure 2. SPACE Focus 7
Figure 3. Space Vernier 8
Figure 4. Looking into a sample picture from dataset we get from SPACE 8
Figure 5. Result from Edge Detection method 11
Figure 6. Smartfusion2 System-on-chip Starter Kit 13
Figure 7. Video processing scheme 14
Figure 8. Principal components in H.264 encoder 17
Figure 9. HardH264 IP's top level 18
Figure 10. Comparing between YUV and RGB color space 18
Figure 11. From Raw data matrices to visualized data 19
Figure 12. Some parameters that can be adjusted 19
Figure 13. Screen cut from akiyo.yuv 22
Figure 14. Screen cut from stefan.yuv 22
Figure 15. Comparing size between compressed videos 22
Figure 16. Image quality of compressed video under different QP (in order 15 - 28 - 35 - 45 - 51) 23
Figure 17. Compressed videos' size comparison 23

1 Introduction

1.1 About Archer BTC and SPACE system

1.1.1 Archer BTC

Archer BTC (Bergen Technology Center) has more than 15 years of experience in developing, manufacturing and supporting ultrasonic logging tools for integrity diagnostics. The company's projects highly focus on SPACE systems – tools that scan the internal surfaces of the well and give engineer 2D and 3D pictures to check if there is a fixing or maintenance must be done.

1.1.2 SPACE system

According to Archer's website, SPACE systems are divided into 3 main models (1):

+ Panorama:

Panorama is used to investigate the condition of the assembly and establish status of the flapper valve. Engineers can detect damages inside the flow tube or confirm flapper position from its results.

Figure 1. SPACE Panorama

+ Focus:

Focus is used to detect collapsed tubing/casing, obstructing fish or parted tubing etc. It has a different viewing methodology from Panorama.

Figure 2. SPACE Focus

+ Vernier:

Vernier looks sideways of the tube, gives another view than Focus and Panorama. Main applications of them is: ID evaluation/Caliper, Pipe thickness evaluation, Corrosion logging and Metal loss evaluation.

Figure 3. Space Vernier

1.2 Problem description

Nowadays, the transferring capacity and long distance between SPACE systems in operation inside oil wells and surface makes it difficult to send raw data to engineers. Therefor ultrasonic pictures are compressed with JPEG before being sent with JPEG-algorithm is realized by using 6 FPGAs. However, many FPGA has been introduced newly that have higher potential, which allows us to simplify the system in physical mean and improve its performance. So other image compressing algorithm or new alternatives could be considered so that we can catch up with technology improvement.

While trying to improve data compressing performance, there will always be two more importance goals for the new system: Working performance in extreme conditions and optimal physical size.

1.3 Solution discussion

The main goal is to transfer as much usable data as possible. Therefor our number one priority is to find out how to compress data better. By figuring out a better way to compress data, and implement it into the existing system, we can spare time and resource to design another data transferring protocol and upgrade bandwidth.

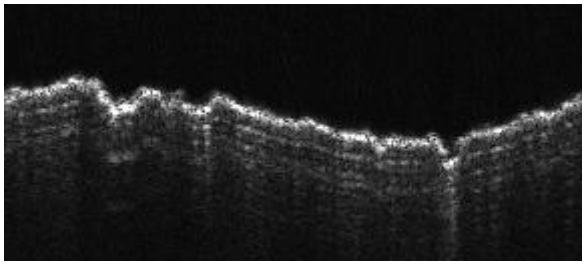

Figure 4. Looking into a sample picture from dataset we get from SPACE

We can see that the main data (light points) takes a very small part of the image, and the rest is duplicated/neutral. That means there is big potential that raw data can be compressed in a very high rate. By reducing each pixel color depth (grayscale depth) and compress duplicated data, we can surely reduce the size of each frame, hence increase transferring speed.

To solve the high temperature problem, we have thought about combining two solution: choosing the components that have highest tolerance with high temperature and use active and passive cooling method. This and the size of new system will be at higher concerning in the later part of project.

2 Specification of requirements

2.1.1 Data transferring

Nowadays, the bandwidth from SPACE systems to surface's node is 219kbps. For JPEG compressed pictures with 288x128 resolution, the bandwidth offers nearly 5 frames per second. However, the ultrasonic sensors on SPACE can record up to 15 frames per second. So, the data compressing technology (JPEG to be specific) is somehow the cause of bottleneck phenomenon for the system. To get a better frame rate, we need to either enlarge the bandwidth or reduce size of each sent frame. We realized that by adding some "tweaks" on the system, with some minor changes in the hardware, we can apply new compressing methods which can make enormous effect om data transferring.

2.1.2 Extreme conditions

To be working down in the oil wells needs some resistance. The temperature here can range from around the freezing point at the surface to 130 Celsius degree deep down. High temperature can reduce circuit speed and degrade transistors inside components. Therefor we need to decide the working temperature range to the system and choose components wisely so that our system can work as stable as possible.

2.1.3 Size of the package

One oil well size can vary from 12,5cm to 90cm wide (3), which leads to another constrains: size. The SPACE systems have already been designed so that they can fit in oil wells from surface to deep down, in another word to be quite small. Since there are many other components inside one SPACE, we need to optimize the size of new data-compressor so that it can fit into SPACE systems.

3 Problem analysis

3.1 Description of possible solutions

Setting aside JPEG image processing, which is now being used on SPACEs, we came up with many solutions: GIF or PNG image processing, Edge detection and video processing. Since GIF and PNG image processing methods do not give a big advantage comparing with JPEG, we decide to consider two best and most possible solutions: Edge detection and video processing.

3.1.1 Edge detection

Realizing that in most cases, the usable data (inside edge of oil wells) is one line across the image, we tried to apply a classical edge detecting method on images: Choose the brightest points as the edge. The result can be seen:

Data compressing rate here is very high, as we need very little data to express position of edge points across the frames. That means the data is now converted:

Original: Brightness of each pixel across the whole frame with resolution 288x128

Converted: Position of each edge point (vertical) across the whole frame (horizontal).

Despite of the high compressing rate and ease of implementation, this method meets one critical problem: precision. Normally there is no problem to look at the shape of inner oil wells, until some defects/ fails take place. Experiences is that defects could not be read easily from an simple picture, and need more spectating/considering from engineers. Then we need as much visual data as possible. By ignoring so much raw data, we risk the chance to detect sudden defects inside oil wells.

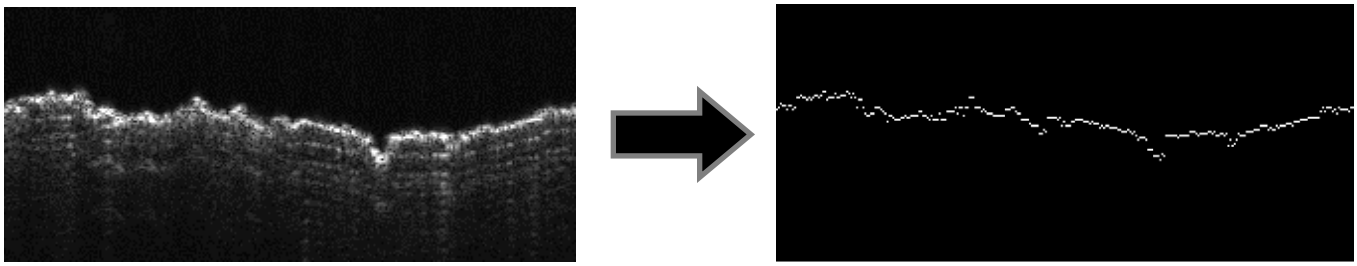

Figure 5. Result from Edge Detection method

3.1.2 Video processing

There is a better method to compress unimportant visual data without losing much raw data: Video compressing. Basically, the goal of one video processing method is to consider, choose and send one frame in a series of picture as main-frame. Afterward it compares other frames with the main-frame to find out differences between them and send information about differences as data for other frames. This method will in theory reduce data load dramatically, which leads to higher "frame rate" as the same bandwidth.

The real problem here is to develop and program a video processing algorithm on FPGA system to compress raw input at one side (encoder), and a protocol to receive and read out videos at the other side (decoder). This leads us to 2 choices:

1. To use an existing encoding method (H.264, Xvid, MPEG-1, etc. This leads us to a standardized method with many documents to read. The downside is that since the method is finished built, there will be a small room to modify/adapt to optimize to the system.
2. To develop a new method based on the existing methods we listed over: This takes much longer time to research, study, develop and test the new algorithm. Obviously, there will be nearly no documentation to read except the basics of data, image and video processing. Advantage is that we get to control over everything (algorithm, data load, framerate, pixels depth etc.)

3.1.3 Issues regarding tools and HW/SW components

This project is driven mainly on simulation and developing board testing, therefor choosing a good hardware based on requirement specifications is the biggest problem. As stated, we will choose an FPGA family to program on. There are two FPGA families we are considering between:

1. Intel MAX 10: single-chip, non-volatile low-cost programmable logic devices (PLDs) which offer these highlights:
   + Internally stored dual configuration flash.
   + User flash memory.
   + Instant on support.
   + Integrated analog-to-digital converters (ADCs).
   + Single-chip Nios II softcore processor support.

   According to Intel's documentation, these are low cost with small form factor packages and much programming possibility offered by up to 50K logic elements. They also have 3 different operating temperature rates depend on which model we choose: Commercial (0 to 85 degree Celsius), Industrial (-40 to 100 degree Celsius) and Automotive (-40 to 125 degree Celsius). (4)

2. Microsemi Smartfusion2: industry's lowest-power, most reliable, and highest-security programmable logic solution. These SoC FPGAs offer up to 3.6x the gate density and up to 2x the performance of previous flash-based FPGA families. They also include:
   + Multiple memory blocks.
   + Multiply-accumulate blocks for digital signal processing.
   + Enhanced 166MHz ARM Cortex-M3 processor.
   + Additional peripherals: Controller area network (CAN), gigabit Ethernet and high-speed USB.

This FPGA family is very powerful with multiple highspeed clock sources and high number of logic elements (up to nearly 150K logic elements). Comparing with Intel MAX 10, Smartfusion2 steps up to military operating temperature range (-55 to 125 degree Celsius). (5)

3.2 Problem analysis conclusion

After consideration, we decided to focus on video processing, with H.264 algorithm. The reason is that it is a good combination between compressing performance, ease of implementation and project's length. Statistically video encoding has much more advance performance in compressing visual data. It will surely be a hard job because of the complexity of algorithm, but since we chose an existing method (H.264 algorithm), there will not be a big problem to find documentations about it.

For hardware solution, we chose Microsemi's Smartfusion2 SoC FPGA family. The main reason is that Smartfusion2s have passed many tests and have quite good high temperature tolerating rate. Archer BTC's seniors is also very familiar with Microsemi's FPGAs and their programming IDE, Libero. Therefor it would be a good move to choose Smartfusion2 for this project.

Figure 6. Smartfusion2 System-on-chip Starter Kit

4 Realization of selected solution

Data compression in general is complex and video compression is one of the most difficult things. Therefor we spent a long period of our project to study and research on video processing methods and the theories behind them. By understanding how it works inside one encoder, we can try our best to optimize the solution.

4.1 Theory

4.1.1 Video processing fundamentals

In this chapter, a wider and more detailed oversight of video processing is studied. Most of the information here is very basis and fundamentally and can be get anywhere. We had found one document from ICDST (6) which is very useful and precise while still giving all information we need.

Definition

One video is described as a sequence of images varying by time. In this document, images in a video sequence is sometimes called frames. Number of frames in one second is called frame rate. Normal human eyes consider videos with about 24-25 frame per second (fps) frame rate as comfortable and continuous.

Throughout this study we make an agreement that there is a compromise in frame rate: Higher frame rate gives better resolution for defect-searching activity but takes more bandwidth.

One video processing procedure can be briefly explained using this block diagram:

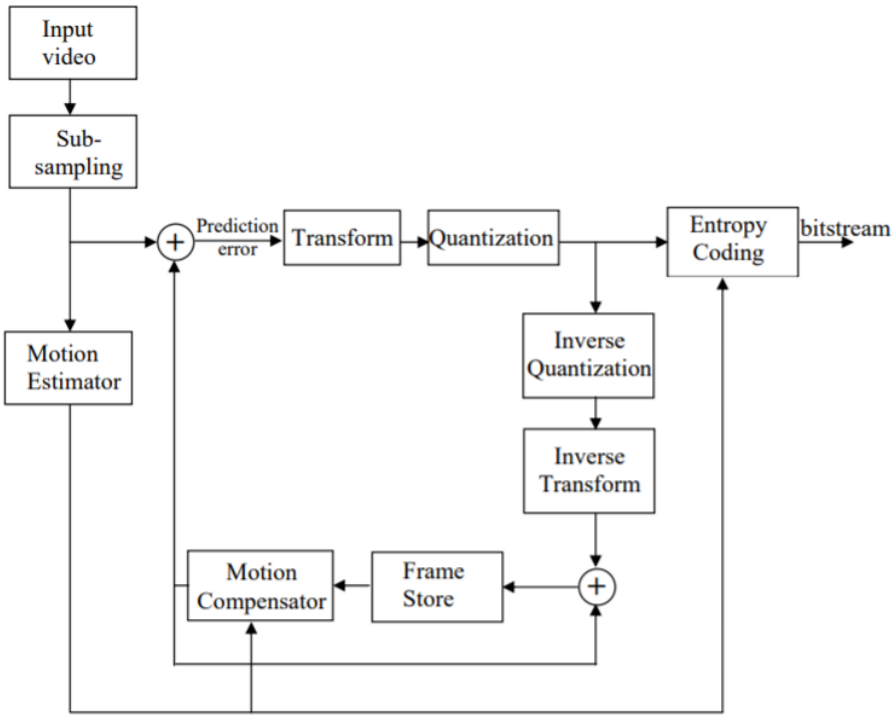

Figure 7. Video processing scheme

(http://dl.icdst.org/pdfs/files/da090a75f2b3c3179de82d428b33ef4d.pdf)

Frame Type

When we talk deeper into video processing methods, there will be three types of video frames: I-frame, P-frame and B-frame.

I-frame: Intra coded frame. These frames contain most of visual data and are used as reference for other types of frame.

P-frame: Predicted frame. These frames contain information from motion compensated prediction from previous I-frame or another P-frame.

B-frame: Bidirectional predictive frame. These frames require lowest number of bits and are encoded using motion compensated prediction from both previous and following frames.

Subsampling

Basically, subsampling reduces the computing work for video processing by reduce the dimension of input. This reduction is applied for often 3 components (Red, Green, Blue colors in RGB or Y – luminance and U,V – chrominance in YUV color space). Depending on the goal, these components are subsampled uneven to give the best processing result.

Example: 4:2:1 YUV means that for every 4 Y samples, there with be 2 U and 1 V samples.

Video compression

Nowadays, with the high amount of video streaming services and increment of videos' resolution over time, it becomes impossible to transmit video sequences without a good video compression. The main ideal of data compression, or video compression in this specific case is to reduce and minimize all redundancies that exist. This is similar to the way we write Sinusoidal functions instead of giving all value over time, it is a waste of time and storing space because a large amount data is duplicated.

The redundancy exists in different levels: Temporal redundancy (in frame level) and spatial redundancy (in pixel level). Removing temporal redundancy is called Interframe coding, while removing spatial redundancy is called Intraframe coding.

Motion estimation and compensation

By estimate the motion of one part of a frame over time, we can predict and reconstruct one frame using a reference frame and the motion parameters. Motion estimation and compensation can also happen in many levels: pixel, small block or large block.

There are two main properties that make a good block matching in motion estimation: Block size and search algorithm.

A large size block gives better computation performance but may contains many motion vectors in one approximation. In contrary a smaller size block requires more computation resource and is more sensible to random noise but offers better approximation.

There are many search algorithms that exist so that man can choose the best one depending on project requirements, or combine between many algorithms together:

+ Full-search

+ Three Step Search

+ New Three Step Search

+ Four Step Search

+ Diamond Search

There are two main classes of block matching algorithms (BMA): Fixed block size and Hierarchical.

Beside these main definitions and steps, one video coding scheme contains of many other blocks: Transform coding, predicting coding, etc.

4.1.2 H.264 encoding algorithm

Nowadays, there are many video compression standards differentiated by the bitstream syntax. Those standards were named H.12x, H.26x by VCEG video coding standards or MPEG-x following naming convention in ISO/IEC MPEG. Over many versions and improvements, H.264 is now one of the most popular video formats with a wide range of applications from low quality and massive internet streaming services to high definition (HD) broadcasting and cinema applications.

H.264 encoding standard sports large number of features that make it much more advanced in comparing with older, outdated video compression standards (7):

+ Multi-picture inter-picture prediction.

+ Spatial prediction.

+ Lossless macroblock coding.

+ Flexible interlaced-scan video coding features.

+ New transform design.

+ Quantization design.

+ In-loop deblocking filter.

Etc. (8)

4.2 Program implementation

4.2.1 Hardh264 IP

There is a finished project named Hardh264, which is a hardware H264 video encoder written in VHDL. We decided to adapt this IP (Intellectual Property) to our Smartfusion2 FPGA, since this IP was first meant to be used on Xilinx tools and FPGAs. A diagram of the principle components is given by designer (9):

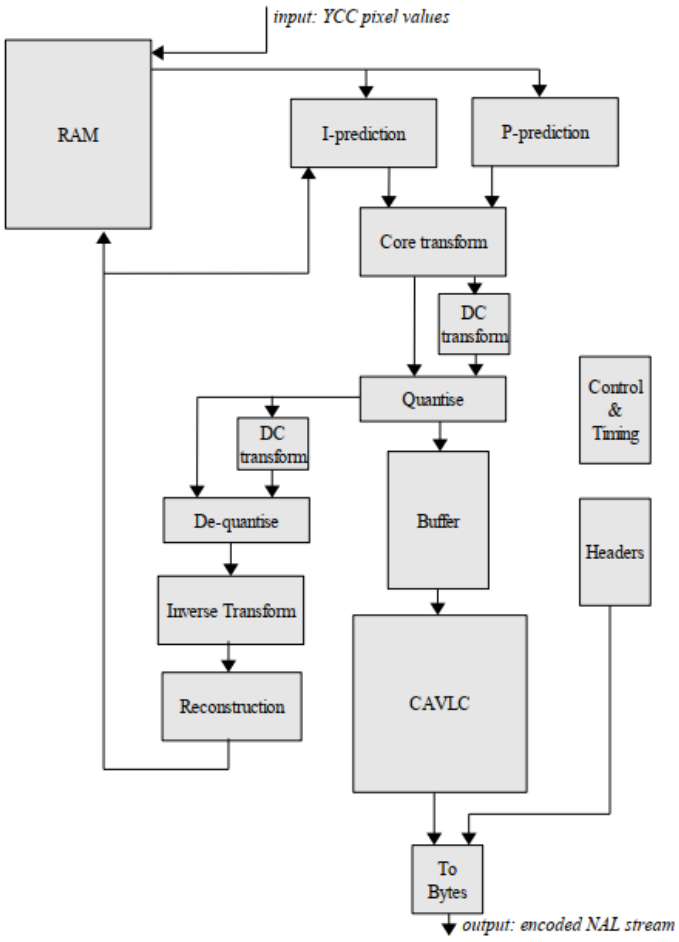

Figure 8. Principal components in H.264 encoder

(Cre: Andy Henson)

This IP is designed to have small and low power components which is very suitable for our project's specifications (low resolution video frames with limited physical size). The IP's flexibility also let us replace components inside the encoder to different applications or customize compressing rate and frames' resolution if necessary.

4.2.2 Adapting to Smartfusion2 on Libero

Despite of being designed for Xilinx FPGAs, we found it not so difficult to adapt HardH264 to our Smartfusion2 on Libero thanks to the skeleton top level which is provided. This skeleton top combined HardH264 components by designer's default specification and made it simple for us to implement it on Microsemi Libero SoC Design Software:

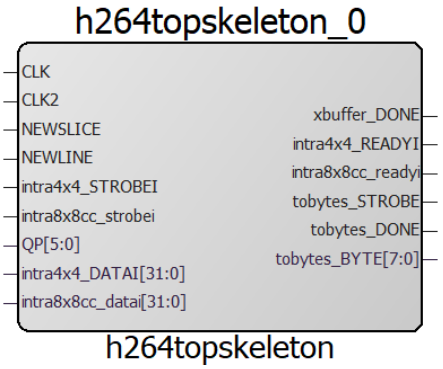

Figure 9. HardH264 IP's top level

When it comes to simulation on ModelSIM, it appeared to be some minor errors because of console code lines. These code lines help us as programmer/designer checking which state our encoder has been executed inside a process loop and can be deleted without affecting encoder's usability. After deleting all code lines that caused errors during compiling, we can get the encoder run as desired.

HardH264 takes files in YUV format as input. YUV is a convention and can be understood as kind of files that contain a series of images in YUV color space. So YUV files are taken as "raw" videos.

YUV (Y- luminance and U, V – chrominance) color space is more effective than RGB (R – Red, G – Green, B – Blue) in term of coding efficiency and bandwidth reduction. Since our raw images are in grayscale, it's very comfortable to use YUV color space since we just need Y – value to draw our images (10):

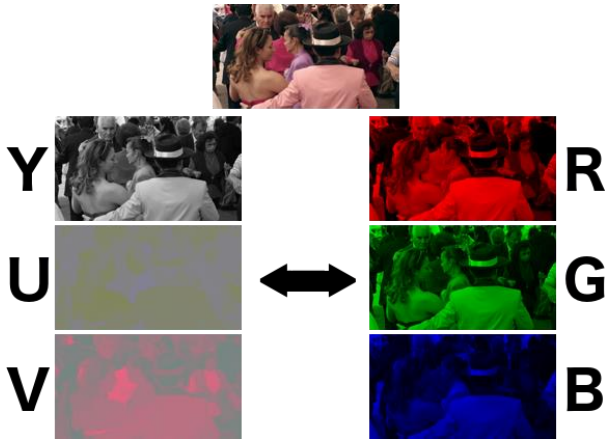

Figure 10. Comparing between YUV and RGB color space

To be able to get to this image, we have written a small MatLAB program that allows us to convert grayscale value 128x288x(frame number) matrices into YUV files. Those files are now used as input for HardH264 simulation:

	1	2	3	4
1	14	14	9	8
2	10	13	14	10
3	17	14	12	12
4	9	17	13	13
5	13	12	15	11
6	10	9	13	15
7	12	10	14	19
8	15	10	7	20
9	16	13	12	13
10	18	15	12	17
11	9	12	13	13
12	12	8	9	11
13	11	12	12	11
14	13	13	15	7
15	12	7	13	5
16	12	13	14	11

```
load('raw.mat');
yuvoutput=fopen('test.yuv','w');


Ui=127*ones([1 18432]);
Vi=127*ones([1 18432]);

for i = 1:1230
    raw=transpose(ultrasound_scan(:,:,i));
    Yi=transpose(raw(:));
    fwrite(yuvoutput,Yi,'uint8');
    fwrite(yuvoutput,Ui,'uint8');
end

fclose(yuvoutput);
yuv2avi('test.yuv',[288 128],'test.avi',30);
```

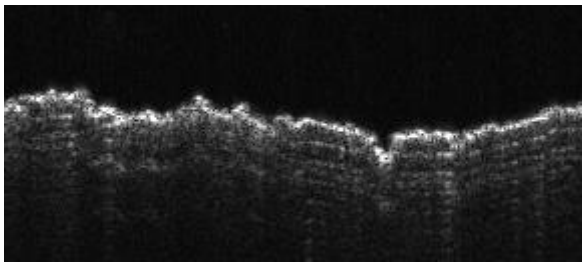

Figure 11. From Raw data matrices to visualized data

As stated, HardH264 allows us to adjust quality of videos to reduce amount of data transferred. This actually is changing quantization parameter (QP). QP is an index used to derive a matrix, so the higher QP is (range from 0 to 51), the lower quality our videos become (and higher compression rate as result). Besides, there are other customizable parameters which help us to control the compressing process: frame width, frame height, number of frames to process, etc.

```
constant IMGWIDTH : integer := 288; --sample stuff is 352x288
constant IMGHEIGHT : integer := 128;
constant IMGSKIP : integer := 0;        --amount to skip between lines (usually 0)
constant MAXFRAMES : integer := 5;  --number of frames to process
constant INITQP : integer := 28;    --0..51
constant MAXQP : integer := INITQP;
constant IWBITS : integer := 9;     --bits required for IMGWIDTH
```

Figure 12. Some parameters that can be adjusted

4.2.3 Video exporting

There is one important operation at the end of video compressing process: Exporting compressed videos. In this case, HardH264 compresses and then converts *.yuv files into *.264 files. *.264 file type is a common and easy to read type which is supported by many applications and decoders, so it makes the process of reading received data become much simpler.

Video exporting happens at the end of hierarchy: On the receiver side. That means after receiving compressed data from SPACE systems, the surface system needs to rewrite it into readable files, in this case is in *.264 data type. The only problem we got when trying to adapt HardH264 into our system so

far is the difference in video resolution: the sample is 352x288 while our images is 288x128. A header at the beginning of the *.264 files decides many specifications of the video frames it's carrying, which we need to pay attention on to get the desired resolution.

There are two important parameters in the header that decide the video and its frames properties:

+ SPS (Sequence Parameter Set): Data that is applied for all the frames in one sequence of pictures.

+ PPS (Picture Parameter Set): Data that is the characteristics for one frame (width, height, cropping flag, etc.)

As above, we can see that it's SPS that decide resolution of all frames. Therefor we will go deeper into SPS's specification. H.264's header information like PPS and SPS is complicated to understand, and there is many concepts of parameter set that can be used, and we found this concept is applicable for our project (11):

Parameter	**Type**
Forbidden_zero_bit	U(1)
Nal_ref_idc	U(2)
Nal_unit_type	U(5)
Profile_idc	U(8)
Constraint_set0_flag	U(1)
Constraint_set1_flag	U(1)
Constraint_set2_flag	U(1)
Constraint_set3_flag	U(1)
Reserved_zero_4bits	U(4)
Level_idc	U(8)
Seq_parameter_set_id	Ue(v)
Log2_max_frame_num_minus4	Ue(v)
Pic_order_cnt_type	Ue(v)
Log2_max_pic_order_cnt_lsb_minus4	Ue(v)
Num_ref_frames	Ue(v)
Gaps_in_frame_num_value_allowed_flag	U(1)
Pic_width_in_mbs_minus_1	Ue(v)
Pic_heigh_in_mbs_minus_1	Ue(v)
Frame_mbs_only_flag	U(1)
Direct_8x8_inference_flag	U(1)
Frame_cropping_flag	U(1)
Vui_parameters_present_flag	U(1)
Rbsp_stop_one_bit	U(1)

Concentrating on changing resolution of frames, we pay attention on pic_width_in_mbs_minus_1 and pic_heigh_in_mbs_minus_1 in datatype ue(v):

ue(v) means that the value is unsigned exponential-golomb coded of variable number of bits.

Mbs means that the resolution is calculated in "macroblock" unit, one macroblock width or height is 16 pixels. For 288x128 resolution, **pic_width_in_mbs_minus_1 = 17** and **pic_height_in_mbs_minus_1 = 7**. By applying those values into our PPS, we can make our video decoder read video files in the desired resolution.

Unsigned exponential-golomb coded

To encode a nonnegative integer using exponential-golomb code, first we write value of that number plus 1 in binary. Then we count the bits written, subtract one to get the result n. Then we write n zeros at the beginning of our binary number.

Example: We want to write the integer number 7 into exp-golomb coded form:

7 + 1 = 8.

8 (dec) ➔ 1000 (bin)

1000 (bin) ➔ 0001000 (exp-golomb)

By using this method, we can help the system control the number of bits used for one value by reading the number of zeros before the first "1" bit. This is very important and helpful since PPS and SPS parameters is placed serial.

In the sample for resolution 352x288, the author used the hex sequence **0x67420028da058259**. After having read this sequence into integer values, and placed them into the table, we changed value of pic_width_in_mbs_minus_1 and pic_height_in_mbs_minus_1 and then rewrite the whole table into hex sequence. The new SPS hex sequence is now **0x67420028da048464**.

5 Testing

Firstly, we decided to test HardH264 with some normal, common open source videos at resolution 352x288 to see how much variation of motion in videos can affect compressing rate.

First video is Akiyo.yuv (12) which shows one reporter sitting in front of a screen with minimal movement:

Figure 13. Screen cut from akiyo.yuv

The second is Stefan.yuv (13) which shows one tennis player in his game, contains of course many movements:

Figure 14. Screen cut from stefan.yuv

We used ModelSIM to compress the first 10 frames of each video, then compared the output files' size (test_rec.yuv is raw data, and test.264 is compressed file):

Figure 15. Comparing size between compressed videos

According to output data, we get an oversight of how number of motions would affect compressing rate. Output file from Stefan.yuv (lots of motion) is more than two times larger than one from akiyo.yuv (nearly zero motion).

We also did 10-frames-tests on real raw data under different quantization parameter (the higher QP is, the worse quality images become):

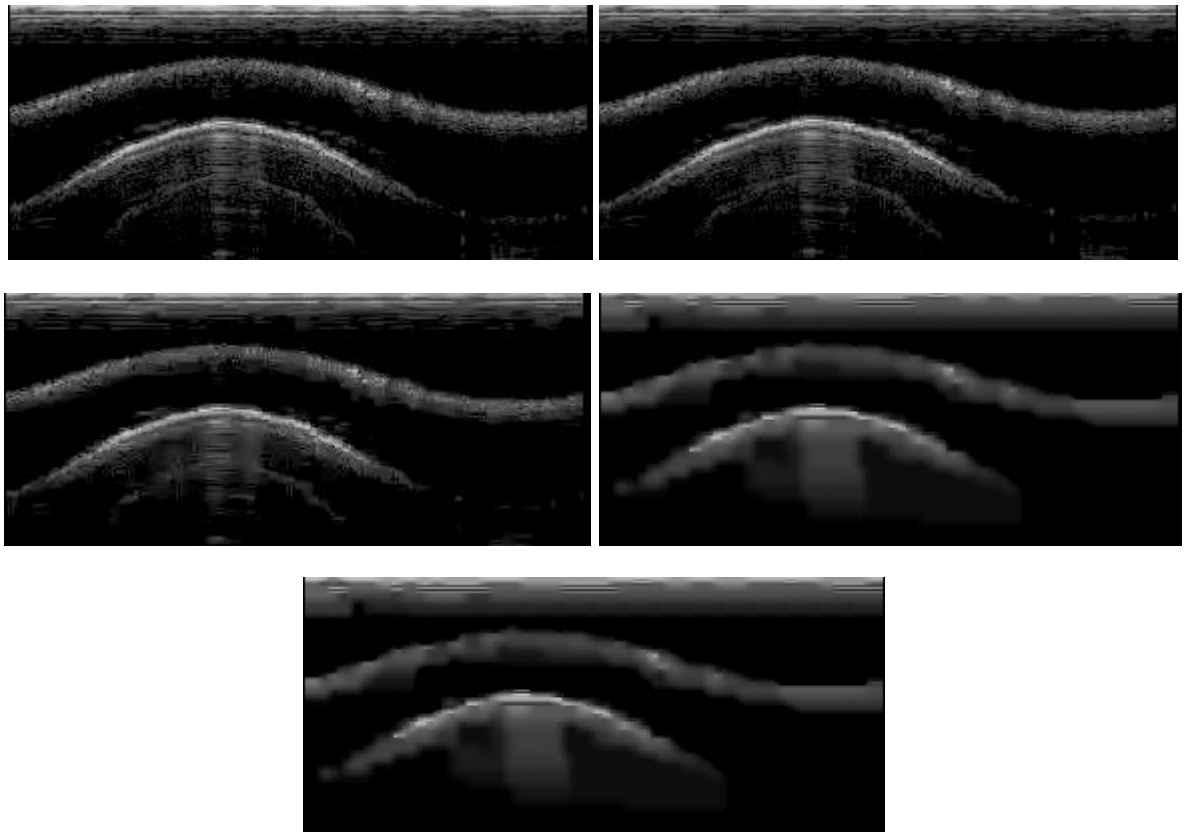

Figure 16. Image quality of compressed video under different QP (in order 15 - 28 - 35 - 45 - 51)

Along with image quality, the file size is also dramatically affected by QP (the last two digits indicate value of QP):

File name	Quantization parameter	File size	Frame per second
Test_10frames_15.264	15	116KB	1,89
Test_10frames_28.264	28	47KB	4,66
Test_10frames_35.264	35	22KB	9,95
Test_10frames_45.264	45	12KB	18,25
Test_10frames_51.264	51	12KB	18,25

Figure 17. Compressed videos' size comparison

6 Discussion

Throughout the testing procedure, the video compressing solution appeared to be very effective in compressing raw data file size and boosting data transferring speed. A small compare has been done between the existing JPEG image processing and simulated results we got from HardH264:

	JPEG	**H.264**
Frame size	5,5KB/frame	1,2 – 11KB/frame
Frames per second	5 fps	2,5 – 22,8 fps
Processing time	(fast enough for 5fps)	4ms/frame

From around 5,5kB per frame in JPEG, we now get averaging size of 1,2kB to 11kB per frame in H.264 videos (file size of H.264 videos depends on selected visual quality and complexity of movements). That leads to the fact that the framerate is boosted from 5fps to max 22,8fps (at the same 219kbps bandwidth).

However, there is still room for much improvement:

1. Reduce number of color spaces: HardH264 is designed for color videos, meanwhile our data is just in grayscale (or Y-value – luminance). If we can optimize the encoder to work with only Y-value, we can save approximately 1/3 unused data in U- and V-values.
2. Vary quantization parameter: As said, we can used QP to control visual quality of the videos. The idea is that at some points, there is less need to take high quality frames, and at some other specific points, we need to take a more carefully look. Finding a way to make QP varies over time as desired, we can make sure that there is no wasted time and resource looking for defects where there is less likely to happen.

7 Conclusion

This project is very interesting and have big affection on how data is processed inside SPACE systems. At beginning we expected that we could do much more than what we have achieved so far. The goal was divided into 3 smaller ones:

1. Designing and simulating the FPGA system on Microsemi Libero and ModelSIM.
2. Realizing the system onto real FPGA (in this case Microsemi Smartfusion2).
3. Testing the system under extreme conditions (high temperature, high pressure, etc.)

Unfortunately, due to the lack of human resource, we could not go as far as planned. The project was run by one student while it was meant for a group of 3-4 students. Therefor we have done around 50% of the expectation:

1. Designing and simulating the FPGA system on Microsemi Libero and ModelSIM.
2. Programming the system and write it onto Smartfusion2 and making sure that there is enough memory space for the system.

A large amount of studying has been done before the HardH264 adaptation could be finished, and we could come to the conclusion that the solution of replacing JPEG image processing by H.264 video encoding is possible. This solution gives us better compressing rate, which means better data transferring, and a flexibility of compromising between video quality and transferred frame rate. Therefor this can be considered as a good improvement from the existing system.

Appendix A. Literature

A.1 VHDL Intellectual Property

H.264 Hardware Encoder in VHDL. © 2008 Zexia Access Ltd. All rights reserved.

http://hardh264.sourceforge.net/

A.2 Reference

(1) Archer [Online]. Archer BTC; 2018. Available on: https://www.archerwell.com/products-services/wireline/cased-hole-logging-services/space/

(2)

(3) J.M.K.C. Donev et al. Energy Education - Oil well [Online]. 2017 [Accessed: May 01, 2019]. Available: https://energyeducation.ca/encyclopedia/Oil_well

(4) Intel Max 10 FPGA Device Overview. 2017 [Accessed: May 03, 2019]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/m10_overview.pdf

(5) Product Brief – Smartfusion2 SoC FPGA. 2018 [Accessed: May 03, 2019]. Available: https://www.microsemi.com/document-portal/doc_download/132721-pb0115-smartfusion2-soc-fpga-product-brief

(6) Introduction to Video Processing. [Accessed: May 05, 2019]. Available: http://dl.icdst.org/pdfs/files/da090a75f2b3c3179de82d428b33ef4d.pdf

(7) Thomas Wiegand, Gary J. Sullivan, Gisle Bjørntegaard and Ajay Luthra. Overview of the H.264/AVC Video Coding Standard. 2003 July [Accessed May 05, 2019]. Available: http://ip.hhi.de/imagecom_G1/assets/pdfs/csvt_overview_0305.pdf

(8) H.264/MPEG-4 AVC – Wikipedia [Online]. [Accessed May 27, 2019]. Available: https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

(9) Andy Henson. H.264 Hardware Encoder in VHDL – notes and usage information. 2008 [Accessed March 01, 2019]. Available: http://hardh264.sourceforge.net/H264-encoder-manual.html

(10) Ronald S. Bultje. Displaying video colors correctly. November 2016 [Accessed May 27, 2019]. Available: https://blogs.gnome.org/rbultje/2016/11/02/displaying-video-colors-correctly/

(11) Ben Mesander. The H.264 Sequence Parameter Set. April 2011 [Accessed April 2019]. Available: https://cardinalpeak.com/blog/the-h-264-sequence-parameter-set/

A.3 Books

"The Digital Signal Processing Handbook, Second Edition". Vijay K. Madisetti, 2010. ISBN: 9781420046083.

A.4 Medias

(12) Akiyo.yuv. http://trace.eas.asu.edu/yuv/akiyo/akiyo_cif.7z

(13) Stefan.yuv http://trace.eas.asu.edu/yuv/stefan/stefan_cif.7z

Appendix B. Abbreviations and dictionary explanations

DSP	Digital Signal Processor
FPGA	Field Programmable Gate Arrays
JPEG	Joint Photographic Experts Group
Fps	Frame per second
SoC	System on Chip
ADC	Analog-to-Digital Converter
RGB	Red-Green-Blue
IP	Intellectual Property
IDE	Integrated Development Environment