COMPSYS 701 – Advanced Digital Systems Design (Semester 1, 2019)

YCbCr and DCT implementations on FPGA

Jonathan King Jkin677

1. Introduction to the selected topic

This report covers the topics of YCbCr Image Transformation and Discrete Cosine Transform operations used in image compression. If gives an overview of each method's purpose then goes into a bit more detail about existing implementations, and methods for increasing performance speed.

1.1 YCbCr Transformation

The YCbCr transformation represent an image using three layers Y, Cb and Cr, instead of the red green and blue layers that most cameras will give as output. This is done by using the equations shown in Fig 1 to map the RGB values to the new YCbCr values. Understanding what the YCbCr layers represent helps explain why this transformation is performed. The Y layer represents what is called the Luminance of the image. The luminance of an image can be thought of as the black and white representation of colour intensities throughout the image and often just looks like a greyscale version of the image when viewed in isolation. The remaining two layers, Cb and Cr are called the chrominance layers, and they store the information about actual colours in the picture. We need to two layers for chrominance so that the number of layers is at least as many as RGB, so that we can reconstruct the RGB layers to display when we uncompressed an image.

$$Y = 0.257R + 0.504G + 0.098B + 16$$

$$Cb = 0.439R + 0.368G - 0.071B + 128$$

$$Cr = -0.148R - 0.291G + 0.439B + 128$$

Figure 1 Equations for YCbCr transformations

The reason separating the chrominance and luminance of the image is important is that the human eye is not as good at detecting inaccuracies in the chrominance layers as it is at detecting them in the luminance layer. This means we can use a technique called Chroma Subsampling to cut the resolution of the chroma layers down by up to a half, without any perceived difference in the reconstructed

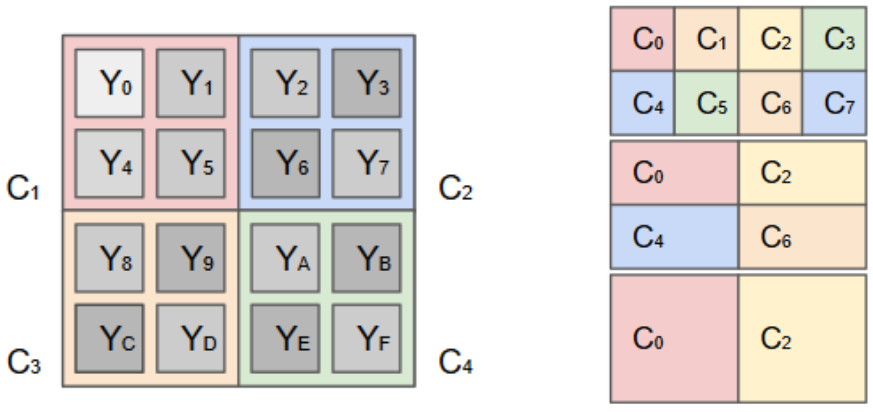

Figure 2a. YCrCb results, 2b. Chroma Subsampling 4:2:2, and 4:2:0

image. The method for reconstruction involves treating the chroma and luminance values as if they were positioned like in Fig 2.

This is the origin of the term 4:2:2 format for image encoding. Chroma Subsampling is not in scope for this project, as it is a simple matrix traversal algorithm that cannot be sped up significantly. But in a real project, this step would be what makes YCrCb transformation useful, and so would be added.

1.2 Discrete Cosine Transform

The DCT is a transformation into the frequency domain that is the next step in the image compression process. It is a useful transformation because it packs the image information into as little coefficients as possible. This means that when we finish the transformation, most values in the new layers will have been reduced in magnitude, while a few will have greatly increased. This is useful in the quantization process (covered by a team member), since it allows most of the values to be stored in much smaller data structures. Fig 3 shows an example of the sort of values to expect after a transformation.

Input matrix

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Output Matrix

186	-18	15	-9	23	-9	-14	19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

Figure 3 Example of DCT results

2. Background information and description of the selected technique(s)

Research has been carried out into efficient mapping of both YCbCr and DCT transformations onto FPGAs. There are ways to accelerate the process and reduce the memory required to execute the computations covered in these papers. The next section covers the algorithms selected to implement these techniques.

C code was used as a software benchmarking tool, as it what is used to compile to most HAL layers. The HLS tool Leg Up was selected as a tool for generating vhd code to synthesize to a Cyclone IV as well as to generate timing analysis results (tests were repeated for Cyclone V and were results were identical). The code that was written only tests the DCT transform over 2D matrices.

3. Description of the algorithms

3.1 YCbCr Simplification

The YCbCr process is a simple operation, compromising of only a few multiplications. Regardless, some methods have been found to optimize the technique. Most papers regarding the topic will convert all the coefficients in the YCbCr equation into integers by shifting them, then shift the result back once all the multiplications and additions have been performed. This is shown in figure 1.

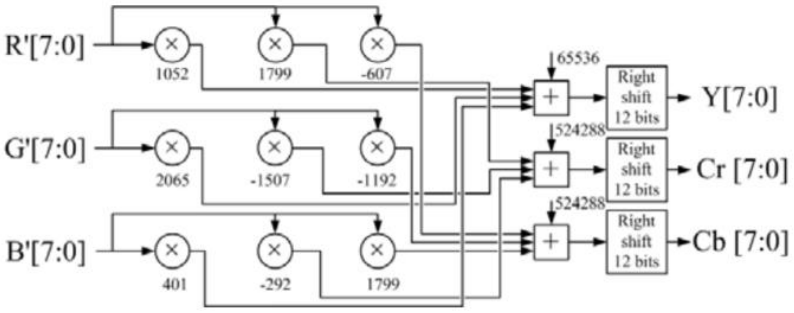

Figure 4 Shifted YCrCb Block Diagram

[1] Finds another method for optimizing this technique using lookup tables to store intermediate results from the multiplications. The paper reports a 3.5x speedup using these methods, and so is an excellent method for improving the speed of this process. As it happens, this method was not necessary to achieve the needed frames per second for this project. But later in the project if this process' efficiency needs to be improved, we could implement this approach.

3.2 DCT Simplification

The Discrete Cosine Transform's 2D formula (shown below) is not a very good target for optimization. It contains trigonometric functions and a great number of sums. To simplify this

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j)$$

process, algorithms have been developed for the 1D DCT that decompose the formula into a much lower number of additions and remove the need for trigonometric functions by using constants to approximate pi and the square root of 2. One such algorithm is called Loeffler's algorithm and is described in Fig 5. [2] Describes how this algorithm can be applied to a 2D array, and how it can be extended to process images.

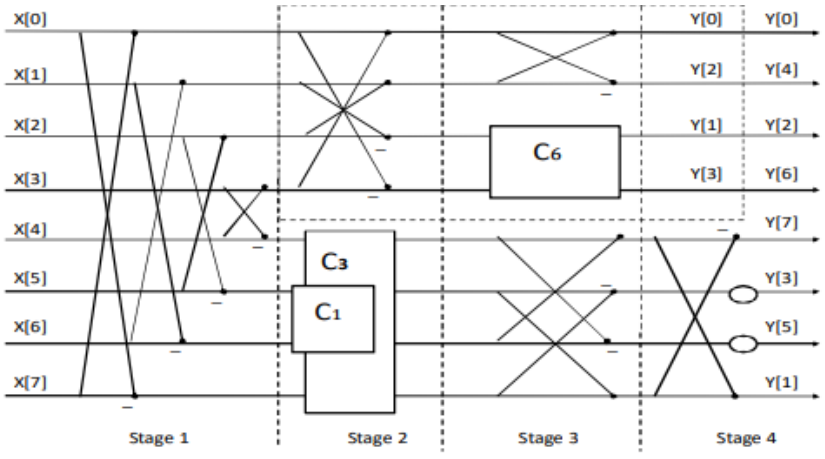

Figure 5 Fast DCT using Loeffler's algorithm [2]

The diagram can be interpreted using the key below.

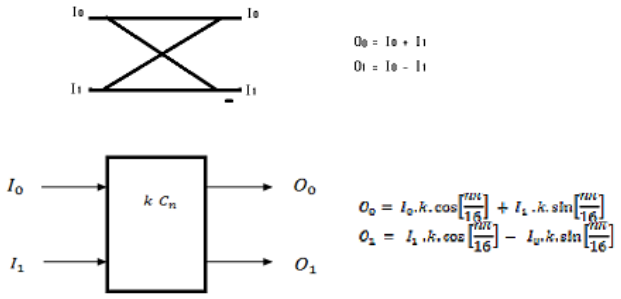

Figure 6 Butterfly diagram key

4. Implementation and the designed testbed

Both algorithms were tested using the LegUp HLS tool. Test benches were written to run the YCbCr algorithm over an array of size 352x288, which will fit the QCIF size outlined in the project brief. The DCT algorithm divides the picture into chunks of 8x8 blocks, so the test performs an 8x8 dct transform a total of 44x36 times, which equals to a full QCIF image

The software benchmarks also include a one dimensional 8-point idct function. There is no 2D idct test function, since the algorithm for performing idct is simply the reverse of dct, and so should have the same timing conditions. This can be seen in fig 6. In a finished product this would have to be implemented, but for planning the system this is not necessary.

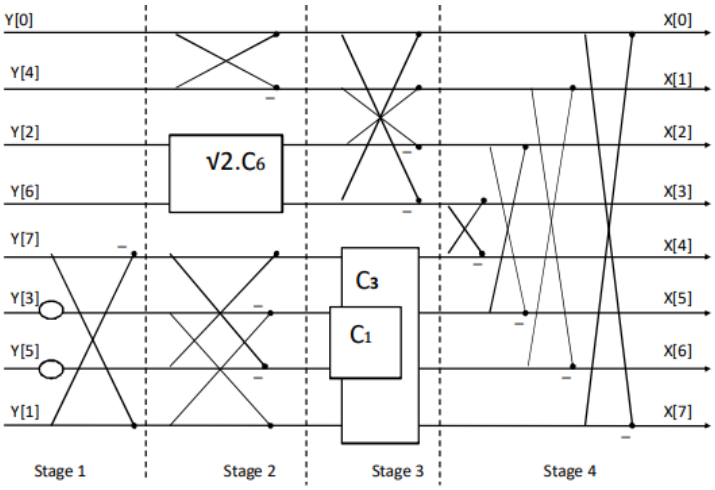

Figure 7 Fast iDCT using Loeffler's algorithm [2]

5. Results and discussion

The timing results from the software benchmarks and the Leg Up simulations are as follows.

Function	Tests	Total Time	Av Case	Possible FPS
YCrCb	400	8969 ms	22.4225 ms	44
DCT	400	375 ms	0.9375 ms	1067
Combined			23.1625 ms	43.17
YCrCb	1	7130 ns	7130 ns	140252
DCT	1	32330 ns	32330 ns	30931
			39460 ns	25342

Figure 8 Table of Simulation results

As can be seen from the results, the time required to complete the software benchmarks is considerable higher in both cases, YCbCr especially. The code for the benchmarks, as well as the test results can be found at https://github.com/JonathanKing01/701in the Software Benchmarks folder.

The hardware results also contain non-pipelined results, which are much slower than the shown results. Without pipelining the hardware implementations wouldn't be feasible in this project as they could not obtain the necessary fps required (>=30). The results of hardware synthesis can be seen below.

```
====== 1. Simulation Cycle Latency ======

LegUp-generated testbench finished in 1614 cycles.

====== 2. Timing Result ======

+-----------------------------------------------------+
; Slow 1200mV 85C Model Fmax Summary                  ;
+------------+-----------------+------------+------+
; Fmax       ; Restricted Fmax ; Clock Name ; Note ;
+------------+-----------------+------------+------+
; 192.86 MHz ; 192.86 MHz      ; clk        ;      ;
+------------+-----------------+------------+------+


====== 3. Resource Usage ======

Fitter Status : Successful - Mon May 06 09:17:09 2019
Quartus Prime Version : 18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name : top
Top-level Entity Name : top
Family : Cyclone IV E
Device : EP4CE115F29C8
Timing Models : Final
Total logic elements : 73 / 114,480 ( < 1 % )
    Total combinational functions : 73 / 114,480 ( < 1 % )
    Dedicated logic registers : 57 / 114,480 ( < 1 % )
Total registers : 57
Total pins : 0 / 529 ( 0 % )
Total virtual pins : 36
Total memory bits : 0 / 3,981,312 ( 0 % )
Embedded Multiplier 9-bit elements : 0 / 532 ( 0 % )
Total PLLs : 0 / 4 ( 0 % )
```

```
====== 1. Simulation Cycle Latency ======

N/A. Cycle latency is not available for custom testbench.

====== 2. Timing Result ======

+-----------------------------------------------------+
; Slow 1200mV 85C Model Fmax Summary                  ;
+------------+-----------------+------------+------+
; Fmax       ; Restricted Fmax ; Clock Name ; Note ;
+------------+-----------------+------------+------+
; 229.31 MHz ; 229.31 MHz      ; clk        ;      ;
+------------+-----------------+------------+------+


====== 3. Resource Usage ======

Fitter Status : Successful - Sun May 05 10:57:24 2019
Quartus Prime Version : 18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name : top
Top-level Entity Name : top
Family : Cyclone IV E
Device : EP4CE115F29C8
Timing Models : Final
Total logic elements : 56 / 114,480 ( < 1 % )
    Total combinational functions : 56 / 114,480 ( < 1 % )
    Dedicated logic registers : 40 / 114,480 ( < 1 % )
Total registers : 40
Total pins : 0 / 529 ( 0 % )
Total virtual pins : 36
Total memory bits : 0 / 3,981,312 ( 0 % )
Embedded Multiplier 9-bit elements : 0 / 532 ( 0 % )
Total PLLs : 0 / 4 ( 0 % )
```

Figure 10 Sythesis results

6. Conclusions

The Sobel example tutorial provided for Leg Up by intel shows how to write c code that will generate vhs components with clear input and output arrays. The current c code has not been modified for this, since it is using the main function as its testbench already. To generate components that could be integrated into a larger project this would have to be modified.

Additionally, the results show that the YCbCr process takes considerably longer to perform than the DCT transformation, due to the much larger number of times it is called parsing a 2D array. This would be a drawback to a pipelined architecture, since a pipeline's clock frequency is limited by its slowest member. Since the YCbCr comes before the DCT there is no need for memory registers to store intermediate results. The long-time of the YCrCb may lead to the DCT or other steps in the pipeline needing a memory register to store outputs. An alternative would be to do quantization in the same step, if it is fast enough.

7. References

1. Hongxu Jiang , Hanqing Li , Tingshan Liu , Ping zhang , Jinyuan Lu, 2013, A Fast Method for RGB to YCrCb Conversion Based on FPGA, 3rd International Conference on Computer Science and Network Technology
2. NEHA V. MAHAJAN1, Dr. J. S. CHITODE2, 2014, May. DCT/IDCT Implementation with Loeffler Algorithm. International Journal of Application or Innovation in Engineering & Management (IJAIEM).
3. NAN YU, 2003, A NEW MOTION ESTIMATION ALGORITHM FOR LOW BIT-RATE REAL-TIME VIDEO AND ITS FPGA IMPLEMENTATION, Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical and Electronic Engineering
4. Gary Wang, 2005, Real-Time Multiprocessor Implementation of an Automated Video Surveillance Application, Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical and Electronic Engineering
5. Emil Dumic ; Mario Mustra ; Sonja Grgic ; Goran Gvozden, 2009, Sept, Image quality of 4:2:2 and 4:2:0 chroma subsampling formats, 2009 International Symposium ELMAR,
6. C.-c. Sun ; S.-j. Ruan ; B. Heyne ; J. Goetze, 2007, Dec, Low-power and high-quality Cordic-based Loeffler DCT for signal processing, IET Circuits, Devices & Systems
7. Haifeng Zhang, 2017, Dec, Realization on image 2D-DCT sparse transform based on FPGA, 2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)