Design Process of the Implementation of the Discrete Cosine Transform on a Microprocessor

Callum Colvine   
V00789488  
[ccolvine@uvic.ca](mailto:ccolvine@uvic.ca)

Faculty of Engineering

Undergraduate student in Computer Science

Submission Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and Time \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 1. Introduction

The goal of this project is to implement the Discrete Cosine Transform on a microprocessor. This begins with simple and inefficient implementation on a workstation. After this step, the implementation is compiled for a microprocessor architecture. Next the code is run on the desired microprocessor. Finally, optimizations are made for the code running on the microprocessor.

Performance requirements … <still unknown since I have nothing to compare to>

Initial implementation was done by Callum. This initial implementation was in C on a workstation. The first iteration of the DCT (dct2\_0.c) runs in completion in about 0.00 seconds.

# 2. Theoretical Background

The DCT is incredibly popular for image compression. DCT2 is useful for JPEG image compression. This is because information about an image can be reduced by a certain number of bits during the process. Lower frequency elements get cut out of the result. This allows reduction of this data during the quantization step.

# 3. Design Process

## 3.1 Workstation Implementation

The first iteration of this project was implemented on a workstation PC, and compiled for PC with GCC-4.8. This initial implementation had 2 inner loops in the DCT calculation for each element. With an 8x8 input matrix “x” to an 8x8 output matrix “X”, the program loops 4096 times (4-depth loops that iterate 0-7).

This implementation is basically 2x1-Dimensional DCT’s. This is the slowest method of performing the DCT2 (a 2-Dimentional DCT). There were also no optimizations like pipelining, avoidance of global variables, or cache optimization.

Values for this iteration were used from Prime Core Solution’s website page on JPEG and DCT1. There were examples or 8x8 input and output matrices for a DCT2. These values were also verified by hand for correctness and understanding.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 251 | 118 | -13 | 6 | -2 | 6 | -1 | 0 |
| 279 | -68 | -8 | -7 | -1 | 4 | -4 | -1 |
| -51 | -14 | 34 | -14 | 5 | 0 | -1 | 0 |
| 27 | 5 | -10 | 8 | -7 | 4 | -5 | 1 |
| -22 | -7 | 14 | -9 | 4 | -2 | 1 | 1 |
| -3 | 15 | -18 | 15 | -6 | 2 | -1 | 2 |
| 7 | -9 | 6 | -6 | 4 | 0 | 0 | 2 |
| 3 | 7 | -9 | 3 | 0 | -2 | -1 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 75 | 76 | 75 | 75 | 69 | 66 | 77 | 71 |
| 73 | 74 | 73 | 74 | 63 | 64 | 68 | 69 |
| 69 | 68 | 71 | 72 | 67 | 58 | 48 | 41 |
| 59 | 55 | 56 | 52 | 47 | 40 | 24 | 9 |
| 51 | 50 | 45 | 41 | 33 | 22 | 7 | -5 |
| 43 | 37 | 32 | 24 | 15 | 5 | -6 | -25 |
| 29 | 21 | 9 | -2 | -10 | -21 | -44 | -69 |
| 9 | -4 | -17 | -35 | -52 | -61 | -57 | -35 |

Fig 3.1.1 Input and output matrices from Prime Core Solution.

Apart from the test values from Prime Core Solution’s website, no external code or references were required apart from the Lesson 110 Slides for SENG 4402. The DCT2 algorithm was implemented based on the equation from the slides.

## 3.2 Compiling for Microprocessor

< GCC with mobile option. ARM >

# References

1. "About JPEG." Education - Discrete Cosine Transform. N.p., n.d. Web. 02 Aug. 2017.

2. "Lesson 110: Discrete Cosine Transform." Mihai Sima, n.d. Web. 02 Aug. 2017.