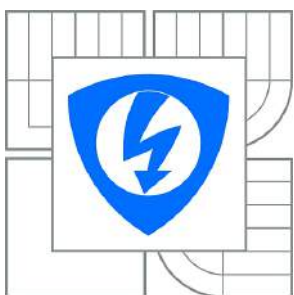# VYSOKÉ UÈENÍ TECHNICKÉ V BRNÌ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAÈNÍCH TECHNOLOGIÍ
ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF MICROELECTRONICS

DESIGN OF DIGITAL IP BLOCK FOR DISCRETE COSINE TRANSFORM

NÁVRH DIGITÁLNÍHO IP BLOKU PRO DISKRÉTNÍ KOSINOVU TRANSFORMACI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE   Bc. FILIP VEŠKRNA
AUTHOR

VEDOUCÍ PRÁCE   doc. Ing. LUKÁŠ FUJCIK, Ph.D.
SUPERVISOR

BRNO 2015

VYSOKÉ UÈENÍ
TECHNICKÉ V BRNÌ

Fakulta elektrotechniky
a komunikaèních technologií

Ústav mikroelektroniky

Diplomová práce

magisterský navazující studijní obor
Mikroelektronika

Student:	Bc. Filip Veškrna	*ID:*	134434
Roèník:	2	*Akademický rok:*	2014/2015

NÁZEV TÉMATU:

Návrh digitálního IP bloku pro diskrétní kosinovu transformaci

POKYNY PRO VYPRACOVÁNÍ:

Navrhnìte na úrovni RTL v jazyce Verilog2001 digitální blok 2D diskrétní kosinovy transformace (DCT II) 8x8 pro HW akceleraci MJPEG komprese video dat z digitální IP kamery s rozlišením 640x480 a snímkovací frekvencí 30 fps. V teoretické èásti semestrální práce rozeberte problematiku algoritmu DCT II, princip využití této transformace v JPEG/MJPEG kompresi a výpoèetnì efektivní algoritmy implementace.

Pro vybranou implementaci DCT algoritmu vytvoøte model v jazyku C a ovìøte jeho správnost proti referenènímu modelu programu Matlab/Octave. Správnou funkcionalitu navrženého DCT II 2D bloku na úrovni RTL ovìøte simulací. V simulaci použijte porovnání s výsledky døíve vytvoøeného C modelu stejného DCT bloku. Syntézou napø. do technologie TSMC 65nm ovìøte celkovou velikost, maximální rychlost a spotøebu bloku. Výsledky ze syntézy (velikost, spotøeba) mohou být porovnány s nabízeným øešením z knihovny DesignWare (Synopsys).

DOPORUÈENÁ LITERATURA:

Podle pokynù vedoucího práce

Termín zadání:	10.2.2015	*Termín odevzdání:*	28.5.2015

Vedoucí práce: doc. Ing. Lukáš Fujcik, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Vladislav Musil, CSc.
Pøedseda oborové rady

UPOZORNÌNÍ:

ABSTRACT

This diploma thesis deals with design of IP block for discrete cosine transform. Theoretical part summarizes algorithms for computation of discrete cosine transform and their hardware usability is discussed. Chosen algorithm for hardware implementation is modeled in C language. Algorithm is described at RTL level, verified and synthesized to TSMC 65 nm technology. Hardware implementation is then evaluated with respect of throughput, area, speed and power consumption.

KEYWORDS

DCT VLSI C model IP-block Verilog-2001

ABSTRAKT

Tato diplomová práce se zabývá návrhem IP bloku pro diskrétní kosinovou transformaci. V teoretické části jsou shrnuty algoritmy pro výpočet diskrétní kosinové transformace a diskutována jejich použitelnost v hardwaru. Zvolený algoritmus pro hardwarovou implementaci je modelován v jazyce C. Poté je popsán na RTL úrovni, verifikován a je provedena syntéza v technologii TSMC 65 nm. Hardwarová implementace je poté zhodnocena s ohledem na datovou propustnost, plochu, rychlost and spotřebu.

KLÍČOVÁ SLOVA

DCT VLSI C model IP-block Verilog-2001

DECLARATION

I declare that I have written my master's thesis on the theme of "Design of digital IP block for discrete cosine transform" independently, under the guidance of the master's thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the master's thesis I furthermore declare that, as regards the creation of this master's thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Sb., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No 40/2009 Sb., Section 2, Head VI, Part 4.

Brno   . . . .
                                                                (author's signature)

ACKNOWLEDGEMENT

Brno   . . . .

(author's signature)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Constantly growing need to transfer more and more information in today's communication requires efficient encoding of the communication channel while preserving the speed requirements and power efficiency at receiver/decoder side. One of methods for reducing data amount is discrete cosine transform (DCT) which is related to widely used fourier transform. Since DCT has been derived in 1974 by N. Ahmed, T. Natarajan and K. R. Rao, DCT and similar transforms are used in numerous applications in engineering and science. It is common compression method for audio formats such MP3, AAC, etc. and both still and moving pictures formats such JPEG, MPEG or H.26x.

This diploma thesis is focused on design of IP block for calculation of DCT. It is divided to two parts: theoretical and practical.

Theoretical part is focused on definition of DCT and its properties, explanation of DCT techniques and describes why DCT is better for compression than other discrete transforms. Its usage in JPEG and MJPEG compression formats is covered by this chapter. Different types of DCT are described. Furthermore there are described some algorithms for both 1-D and 2-D DCT and their hardware requirements are discussed.

Practical part maps design process of IP block in Verilog 2001 language. After some hardware prerequisites are discussed, design methodology is described. Complete workflow of design starts with model in language C description and its verification against reference. Detailed RTL design using Verilog 2001 language based on C model is then depicted. Thereafter verification methodology of created RTL is discussed and then synthesis is done in TSMC 65 nm technology.

1 THEORETICAL PART

Theoretical part is in the first place focused on Discrete Cosine Transform (DCT) definition and summary of methods for computation DCT. Then usage of DCT in JPEG and MJPEG compression is summarized. Thereafter some perspective algorithms and their requirements are described followed by summary of human visual perception. At the final of this chapter Q number representation is described.

1.1 Fourier transform

For the beginning, it is useful to define and briefly describe Fourier transform. It is used to express a time domain function to frequency domain.

1.1.1 Definition of Fourier transform

Fourier transform is defined in equation 1.1, where $j = \sqrt{-1}$, $\omega = 2\pi f$ is radian frequency and f is the frequency in Hertz. [1]

$$X(\omega) \equiv F[x(t)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{1.1}$$

Reversely, function $x(t)$ could be obtained by the inverse Fourier transform as described in equation 1.2. [1]

$$x(t) \equiv F^{-1}[X(\omega)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} X(\omega)e^{-j\omega t}dt \tag{1.2}$$

Functions in Equations 1.1 and 1.2 describe forward and inverse Fourier transform. If $x(t)$ is defined only for $t > 0$ a function $y(t)$ can be constructed as given by:

$$y(t) = x(t) \qquad t \geq 0,$$

$$x(-t) \qquad t \leq 0.$$

Then

$$F[y(t)] = \left(\frac{1}{2\pi}\right)^{1/2} \left\{ \int_{0}^{\infty} x(t)e^{-j\omega t}dt + \int_{-\infty}^{0} x(-t)e^{-j\omega t}dt \right\}$$

$$= \left(\frac{1}{2\pi}\right)^{1/2} \int_{0}^{\infty} x(t)[e^{-j\omega t} + e^{j\omega t}]dt$$

$$= \left(\frac{1}{2\pi}\right)^{1/2} \int_{0}^{\infty} x(t)\cos(\omega t)dt \tag{1.3}$$

Given by equation 1.3, Fourier cosine transform (FCT) can be defined as shown in Equation 1.4.

$$X_c(\omega) \equiv F_c[x(t)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)dt \qquad (1.4)$$

Fourier inversion as defined in equation 1.3 now can be applied to equation 1.4, because $X_c(\omega)$ is an even function of ω. [1] Resulting equation is shown in 1.5.

$$y(t) = x(t) \equiv F_c^{-1}[X_c(\omega)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty X_c(\omega)\cos(\omega t)d\omega \qquad (t \geq 0) \qquad (1.5)$$

1.2 Discrete cosine transform

In previous section Fourier cosine transform was defined. In this section DCT will be presented and described. The DCT introduces no loss to the source image. It transforms image to a frequency domain where coefficients can be encoded more effectively.

1.2.1 Definition of discrete cosine transform

There are four types of discrete cosine transform as classified by [2]. Each one is defined by following equation:

DCT-I

$$[C_{N+1}^I]_{mn} = \left(\frac{2}{N}\right)^{1/2}\left[k_m k_n \cos\left(\frac{mn\pi}{N}\right)\right] \qquad (1.6)$$

DCT-II

$$[C_N^{II}]_{mn} = \left(\frac{2}{N}\right)^{1/2}\left[k_m \cos\left(\frac{m(n+\frac{1}{2})\pi}{N}\right)\right] \qquad (1.7)$$

DCT-III

$$[C_N^{III}]_{mn} = \left(\frac{2}{N}\right)^{1/2}\left[k_n \cos\left(\frac{(m+\frac{1}{2})n\pi}{N}\right)\right] \qquad (1.8)$$

DCT-IV

$$[C_N^{IV}]_{mn} = \left(\frac{2}{N}\right)^{1/2}\left[k_n \cos\left\{\frac{(m+\frac{1}{2})(n+\frac{1}{2})\pi}{N}\right\}\right] \qquad (1.9)$$

As shown in Figure 1.1 discrete cosine transform has almost equal mean-square error as Karhunen-Loeve transform (KLT). KLT provides a benchmark against other discrete transforms may be judged. It is ideal (in the decorrelation sense) but impractical tool. It can be also used for solving partial differential equations using spectral methods. [1] DCT has some advantages over Fourier transform:

- it has only real results (no complex)
- it has strong energy compaction property – most of the information of the signal is concentrated in a few low-frequency components
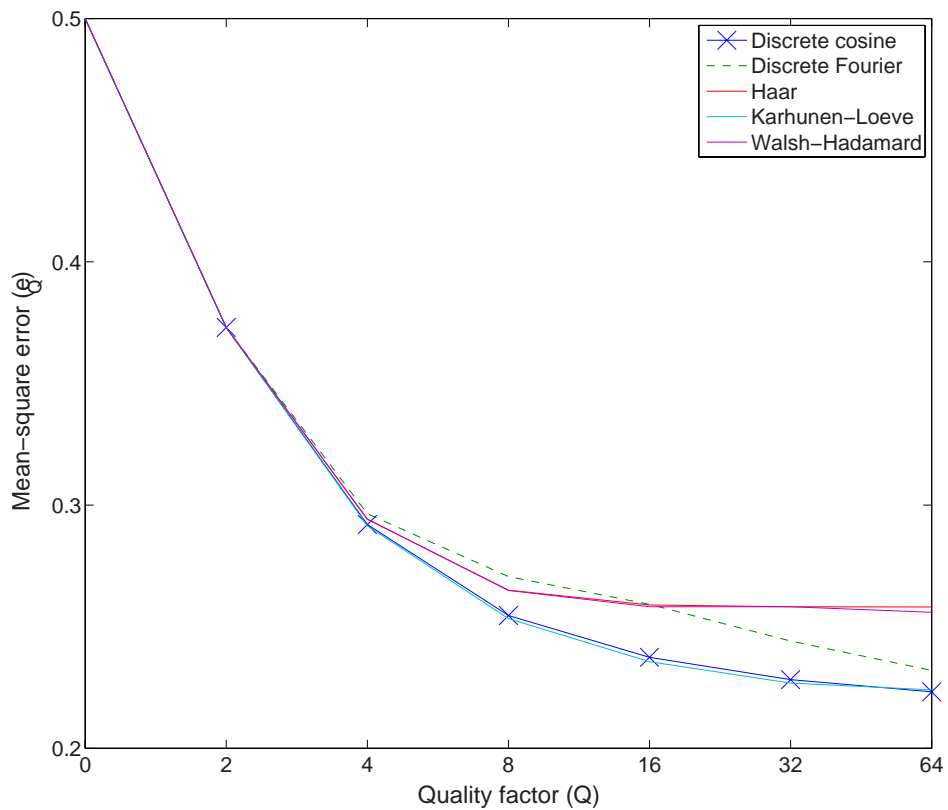
Fig. 1.1: Mean-square error performance of various transforms for scalar Wiener filtering; $\rho = 0.9$ [3]

DCT-II is one of the best tools in digital signal processing due to following properties [1]:

- it has best variance distribution compared to other non-KLT transforms
- it has only real coefficients

Generally, DCT-II is called DCT. From this point expression 'DCT' refers to DCT-II variation.

1.2.2 DCT-II

Equation 1.10 defines the N-point DCT-II as follows: An input data sequence $\{x_n, n = 0, 1, 2, \ldots, N-1\}$ is transformed into the N-point output sequence $\{y_n, n = 0, 1, 2, \ldots, N-1\}$ [4].

$$y(k) = C \cdot a_k \cdot \sum_{n=0}^{N-1} x(n) \cdot \cos\left(\frac{2\pi \cdot (2n+1) \cdot k}{4N}\right) \tag{1.10}$$

where

$$a_o = \cos\left(\frac{\pi}{4}\right)$$

$$a_k = 1, \ldots N - 1$$

According to MATLAB specification (which implementation is used as reference), one dimensional DCT is described in Equation 1.11. [5]

$$y(k) = w(k) \sum_{n=1}^{N} x(n) \cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right), \quad k = 1, 2, \ldots, N, \tag{1.11}$$

where

$$w(x) = \begin{cases} \dfrac{1}{\sqrt{N}}, & k = 1, \\ \sqrt{\dfrac{2}{N}}, & 2 \leq k \leq N, \end{cases}$$

Note that DCT values are scaled by this definition, whereas values at the output of binDCT algorithm aren't scaled. This has to be taken into account when evaluating results of model against MATLAB specification. Figure 1.2 shows 8 samples of input
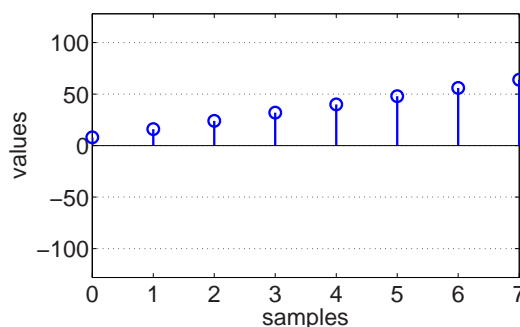
Fig. 1.2: 8-samples signal

signal. Spectra are shown in Figure 1.3a and 1.3b. Input have 8-bit resolution with 2nd complement (values -128 to 127) as well as output DCT and DFT coefficients.

(a) DCT of input signal   (b) DFT of input signal
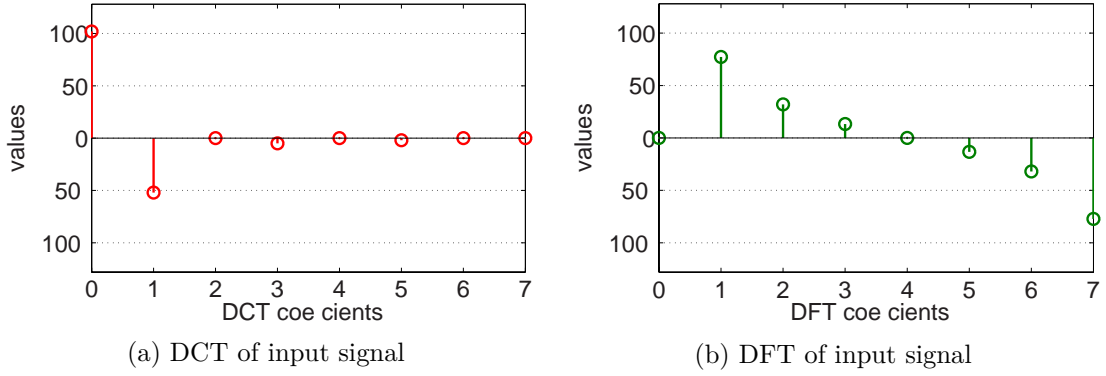
Fig. 1.3: 8-point one-dimensional discrete transforms

Note there is visible important property of DCT when compared with DFT – its coefficients are much more concentrated near DC component[1]

1.2.3   Two dimensional DCT-II

We suppose that $[g]$ is an $(M \times N)$ matrix representing input data that are two-dimensional and $[G]$ are two-dimensional DCT-II coefficients. This is described in Equation 1.12. [1]

$$G_{uv} = \frac{2c(u)c(v)}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} \cos\left[\frac{(2m+1)u\pi}{2M}\right] \cos\left[\frac{(2n+1)u\pi}{2N}\right] \qquad (1.12)$$

where $u = 0, \ldots, M-1, v = 0, \ldots, N-1$, and
if $k = 0$

$$c(k) = \frac{1}{\sqrt{2}}$$

otherwise

$$c(k) = 1$$

Two-dimensional DCT of M-by-N matrix according to MATLAB is defined as described in Equation 1.13 [6]

$$B_{pq} = a_p a_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \qquad (1.13)$$

where

$$a_p = \begin{cases} \dfrac{1}{\sqrt{M}}, & p = 0, \\[2mm] \sqrt{\dfrac{2}{M}}, & 1 \leq p \leq M-1, \end{cases}$$

---

[1]It means coefficient number 0

17

$$a_q = \begin{cases} \dfrac{1}{\sqrt{M}}, & q = 0, \\ \sqrt{\dfrac{2}{M}}, & 1 \leq q \leq N - 1, \end{cases}$$

Its Inverse DCT is defined as described in Equation 1.14. [6]

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} a_p a_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \tag{1.14}$$

where

$$a_p = \begin{cases} \dfrac{1}{\sqrt{M}}, & p = 0, \\ \sqrt{\dfrac{2}{M}}, & 1 \leq p \leq M - 1, \end{cases}$$

$$a_q = \begin{cases} \dfrac{1}{\sqrt{M}}, & q = 0, \\ \sqrt{\dfrac{2}{M}}, & 1 \leq q \leq N - 1, \end{cases}$$
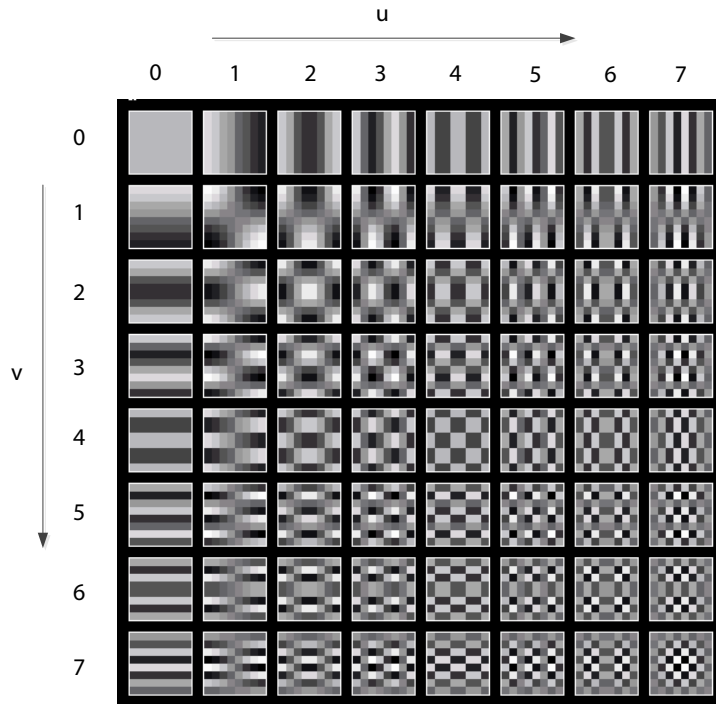
Fig. 1.4: Frequency representation of 8×8 DCT coefficients [7]

Figure 1.4 illustrates frequency representation of DCT coefficients matrix of size 8×8. Element $[0, 0]$ represents DC coefficient. As u is increasing, elements in matrix represents higher vertical frequencies as well as v for horizontal frequencies so element $[7, 7]$ represents highest vertical and horizontal frequency.

1.3 Applications of DCT

DCT or its derivatives like modified discrete cosine transform (MDCT) which is based on DCT-III are used in various applications in engineering and science. One-dimensional version is part of compression methods for audio like AAC, Vorbis, WMA or MP3. Two-dimensional version is used in compression methods for still pictures like JPEG or moving pictures like MJPEG, MPEG or H.26x.

1.3.1 JPEG

JPEG is a method of lossy compression for digital images developed by Joint Photographic Experts Group and introduced in 1992. It is popular method for compressing images produced by most digital cameras because of its relatively good compression ratio 1:10, which could be easily achieved. [8] As mentioned above, DCT is part of
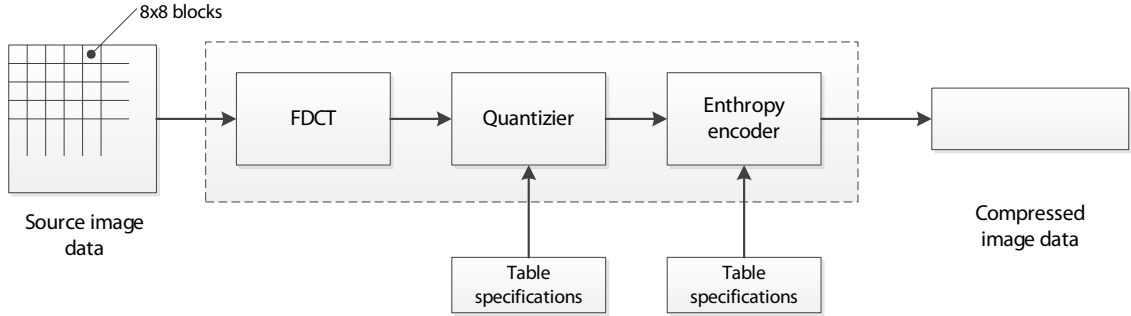
Fig. 1.5: DCT-based encoder processing steps [8]

JPEG compression method. An example of DCT-based encoder is shown in Figure 1.5.

We suppose that source image data are matrix 8×8 of 8-bit signed grayscale values (-128 – 127). At the encoder's input source image data are grouped into 8×8 blocks and routed to Forward DCT (FDCT). DCT and IDCT computation of 8×8 block as defined by JPEG standard is described by Equation 1.15 and 1.16. [9]

$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \qquad (1.15)$$

$$S_{yx} = \frac{1}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} C_u C_v S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \qquad (1.16)$$

where

$$C(u), C(v) = \begin{cases} \dfrac{1}{\sqrt{N}}, & u, v = 0, \\ 1, & 1 \le u, v \le N, \end{cases}$$

This is followed by Quantizier and Enthropy encoder whose parameters are dependent on Table specifications (JPEG quality). There are compressed image data on the output of the encoder chain.
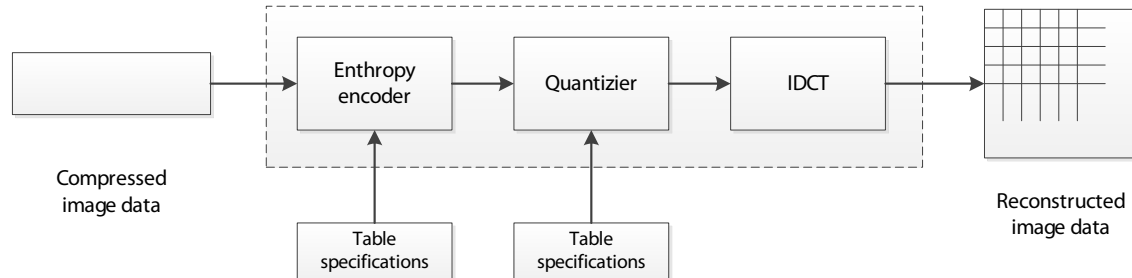
Fig. 1.6: DCT-based decoder processing steps [8]

Figure 1.6 shows a simplified DCT-based decoder. Compressed image data are brought to Enthropy decoder, then to Quantizier (whose parameters are also dependent on JPEG quality). Thereafter quantized data are input to Inverse DCT (IDCT) block and reconstructed as image on the output.

1.3.2   M-JPEG

Motion JPEG (M-JPEG or MJPEG) is a video compression format. Each frame or interlaced field of digital video is compressed separately using JPEG compression algorithm. It means that there is not any motion-prediction method used unlike MPEG2 [10] or MPEG4 codec [11]. MJPEG is widely used in consumer electronics devices like digital cameras, IP cameras, webcams and game consoles. It has some advantages over motion-prediction based formats [12]:

- Resulting quality is independent from the motion in the image – MPEG4 visual quality is often reduced when footage contains lots of movement
- Latency of image processing is minimized
- Simple implementation
- Hardware requirements are low due to its simplicity

Otherwise, there are some disadvantages:

- Single exact format is not specified – Microsoft has its own method to store M-JPEG in AVI files [13] as well as Matroska MJPEG container is being developed [14].
- M-JPEG is not as much efficient as more modern formats such Motion JPEG 2000 or motion prediction based codecs H.264/MPEG-4 AVC.

1.4 Overview of DCT computation methods

Many algorithms for computing DCT-II have been developed. Some of them are optimal to be used in software for general-purpose CPU. What is advantageous for software implementation is not generally optimal for a programmable DSP processor, FPGA or dedicated VLSI chip.

Development of DCT efficient algorithms began after publication by Ahmed et al [3] in 1974 when DCT was first described. Generally, computation methods can be categorized [1]:

- Via DFT
- Using sparse factorization scheme
- Decimation-in-Time (DIT)
- Decimation-in-Frequency (DIF)
- Via other discrete transform
- Through prime factor decomposition
- Recursive computation
- Via planar rotation

1.4.1 DCT via DFT

Since DCT is related to DFT, many algorithms have been developed, for example [15] and [16]. Block diagram of DCT algorithm via DFT computation is shown in Figure 1.7. In first section N-point signal is merged to 2N-point signal. Then signal
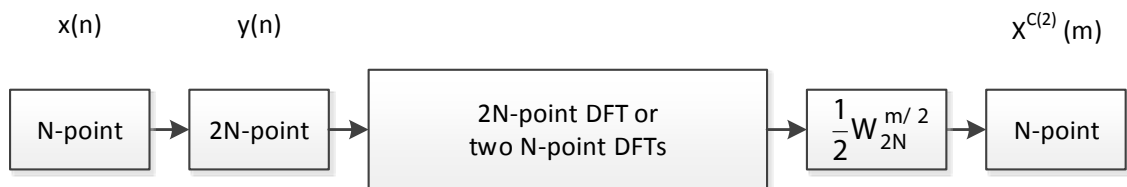
Fig. 1.7: N-point DCT via 2N-point DFT [1]

is transformed via DFT and converted from complex values to real values.

Equation 1.17 shows how number of multiplication needed can be computed. Since this method has reduced number of multiplications needed, this is at the cost of increased topology complexity – it is slow. [1]

$$\left(\frac{N}{2}\right)\log_2 N \tag{1.17}$$

1.4.2 Using sparse factorization scheme

Sparse factorization scheme is another method of calculation DCT. This is done by normalization factor of input matrix and decomposition into sparse matrices [1]. Number of real additions required can be calculated by equation 1.18. Calculation of multipliers needed is shown in equation 1.19. First algorithm using sparse factorization scheme was developed by Chen. [17]

$$\frac{3N}{2}(\log_2 N - 1) + 2 \tag{1.18}$$

$$N \log_2 N - \frac{3N}{2} + 4 \tag{1.19}$$

1.4.3 Decimation-in-Time (DIT) and Decimation-in-Frequency (DIF)

These algorithms consists of input sequence rearrangement. DIT algorithm [18] requires number of real additions as shown in equation (1.20) and number of real multiplications as shown in equation (1.21).

$$\left(\frac{3N}{2} - 1\right) \log_2 N + \frac{N}{4} + 1 \tag{1.20}$$

$$\left(\frac{N}{2}\right) \log_2 N + \frac{N}{4} \tag{1.21}$$

Number of operations needed for calculating DCT components via DIF [19] is shown in equations (1.22) –number of additions and (1.23) – number of multiplications. All operations are real can be calculated by using real arithmetic only thus not complex.

$$\left(\frac{3N}{2}\right) \log_2 N - N + 1 \tag{1.22}$$

$$\left(\frac{N}{2}\right) \log_2 N \tag{1.23}$$

1.4.4 DCT via other discrete transforms

Using other discrete transform, there are 2 common methods to compute DCT:
- DCT via Walsh-Hadamard Transform (WHT) [20]
- DCT via Discrete Hardley Transform (DHT) [1]

1.4.5   DCT via Prime Factor Algorithm

Since PFA was originally developed for computation DFT coefficients and DCT can be directly related to the DFT, so the PFA can be used for DCT computation [21]. Advantage of this method is that it is not restricted to radix-2 but has one disadvantage – it requires very complex index mapping.

1.4.6   DCT via recursive computation

DCT coefficient can be computed by recursive algorithm. This was first proposed by Hou [22]. It is fast, require simple index map but it is slower when implemented without multiplexing. The number of operations needed is shown in Figure 1.24 for additions and in Figure 1.25 for multiplication.

$$N - 1 \tag{1.24}$$

$$\left(N^2 + N - \frac{7}{3} \right) \tag{1.25}$$

1.4.7   DCT realization via planar rotations

This method requires only rotations that replace adders and multipliers, but it is slow. [23].

1.5   Algorithms for DCT

In this section some perspective algorithms for computation of DCT coefficients will be described.

1.5.1   Loeffler's DCT algorithm

In 1989 Loeffler [4] proposed fast algorithm for 8-point DCT computation which requires only 11 multiplications and 29 additions. Number of multiplications has been reduced to the theoretical lower bound. For 16-point DCT these values are 31 multiplications and 81 additions. Table 1.1 shows number of operations needed for 8-point DCT calculation using different algorithms.

Algorithm's structure is shown in Figure 1.8. Symbols used are described in Figure 1.9. Data in stages 1 to 4 are being executed in series and are depended to each other, so it cannot be evaluated in parallel. After stage 1 algorithm is divided in two parts – one for calculation of even coefficient (upper 4 signals) and one for

23

Algorithm	Chen's	Wang's	Lee's	Vetterli's	Suehiro's	Hou's	Loeffler's
mult.	16 (13)	13	12	12	12	12	11
add.	26 (29)	29	29	29	29	29	29
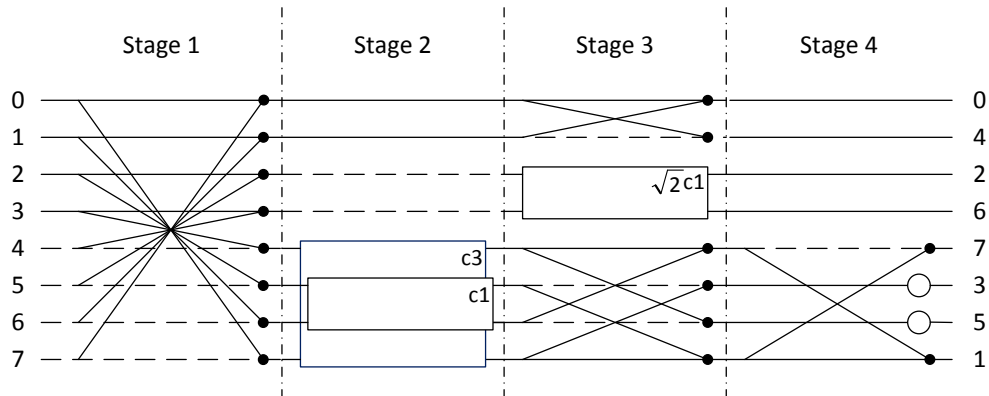
Fig. 1.8: 8-point DCT algorithm with 11 multiplications. Symbols are described in figure 1.9 [4]

calculation of odd coefficients (lower 4 signals). Even part is 4-point DCT, which is again separated in two parts after stage 2. Second building block shown in Figure 1.9 can be calculated using only 3 multiplications and 3 additions.
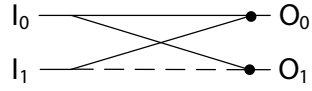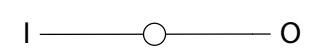
symbol   equations   effort

I_0 ⟶⟶ O_0 $O_0 = I_0 + I_1$

I_1 ⟶⟶ O_1 $O_1 = I_0 - I_1$ 2 add

I_0 ─[kcn]─ O_0 $O_0 = I_0 \cdot k \cdot \cos\dfrac{n\pi}{2N} + I_1 \cdot k \cdot \sin\dfrac{n\pi}{2N}$ 3 mult +

I_1 ─────── O_1 $O_1 = -I_0 \cdot k \cdot \sin\dfrac{n\pi}{2N} + I_1 \cdot k \cdot \cos\dfrac{n\pi}{2N}$ 3 add

I ──◯── O $O = \sqrt{2} \cdot I$ 1 mult

Fig. 1.9: Symbols used to display an algorithm structure [4]

Variations of Loeffler's DCT algorithm

Different variations of stages can be derived from original structure shown in Figure 1.8. Figure 1.10 shows possible variations of stages 2 and 3 of the even part. This is achieved by inverting stages 2 and 3. Different variations of stage 1 with same low

Fig. 1.10: Inversion of stages 2 and 3 of the even part [4]

complexity are shown in Figure 1.11. Figure 1.12 illustrates different combinations
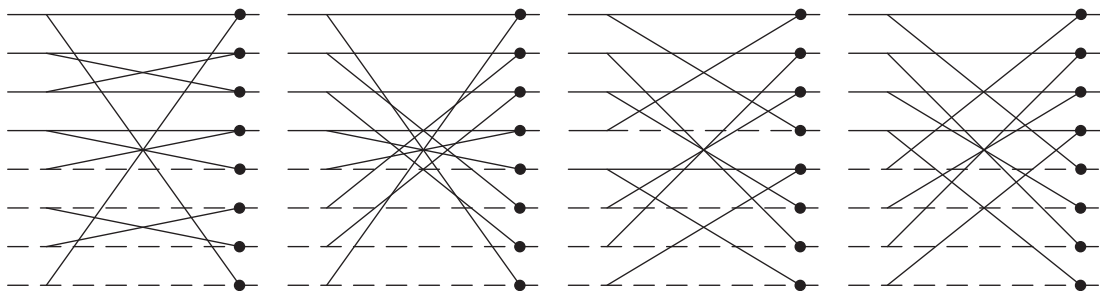
Fig. 1.11: Variations of stage 1 of the algorithm [4]

of rotation angles of stage 2, 3 and 4 of the odd part. Note that this combinations have different coefficients order and sign at the output. Another combinations of stages 2, 3 and 4 for the odd part can be obtained by inverting cross-add module
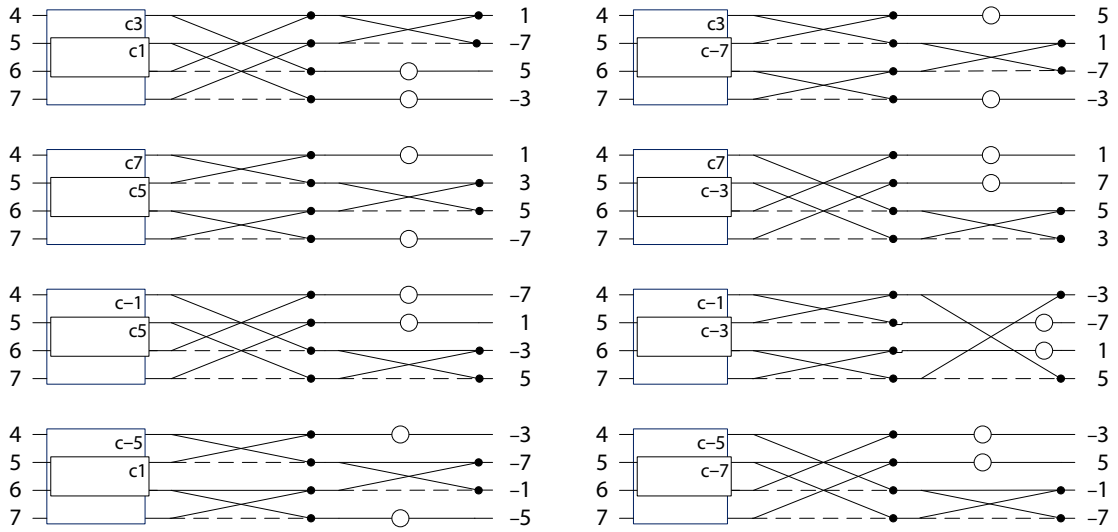
Fig. 1.12: Variations of stages 2, 3 and 4 of the odd part [4]

or combination of them – exchanging the adder and subtractor. This is shown in Figure 1.13.
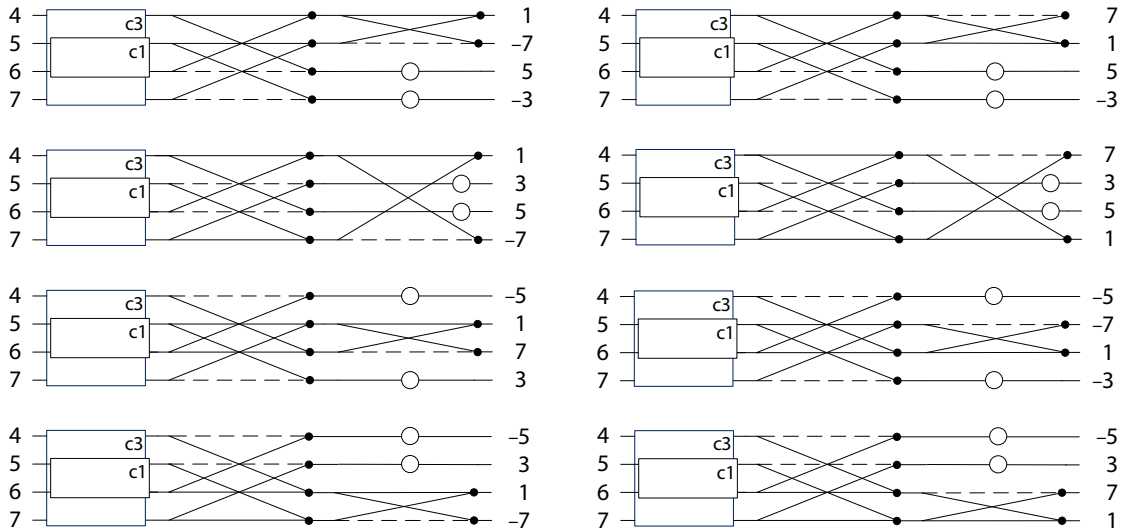
Fig. 1.13: Variations of stage 2, 3 and 4 by inverting add/subtract modules [4]

1.5.2 Multiplierless approx. of DCT with lifting scheme
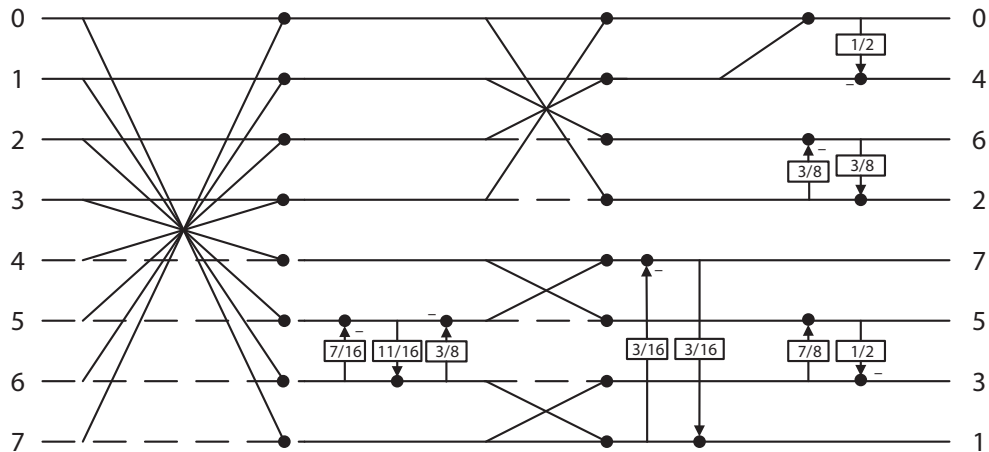
Fig. 1.14: 8-point BinDCT algorithm version A [24]

Another VLSI perspective method how to compute N-point DCT is using fast multiplierless approximation with the lifting scheme called Binary Discrete Cosine Transform (binDCT). This method uses only shift and add operations that can be implemented using simple binary arithmetic. As described in section 1.5.1, DCT can be directly computed using 11 multiplications and 29 additions. Number of operations for three similar binDCT algorithms compared to direct method are shown in Table 1.2. When compared to common DCT computation method, which has coding gain 8.83 dB, binDCT has coding gain ranges 8.77 to 8.82 dB. It means

Tab. 1.2: Number of operations needed for different 8-point binDCT [24]

Algorithm	No. of mul.	No. of add.	No of. shifts	Coding gain
8-point DCT (Loeffler)	11	29	0	8.83 dB
8-point binDCT-A	0	36	19	8.82 dB
8-point binDCT-B	0	31	14	8.77 dB
8-point binDCT-C	0	30	13	8.77 dB

binDCT is only 0.1 to 0.5 dB below the standard DCT. Figures 1.14, 1.15 and 1.16 show structure of three similar version of binDCT algorithm.

Three versions of Inverse binDCT are shown in Figure 1.17, 1.18 and 1.19.
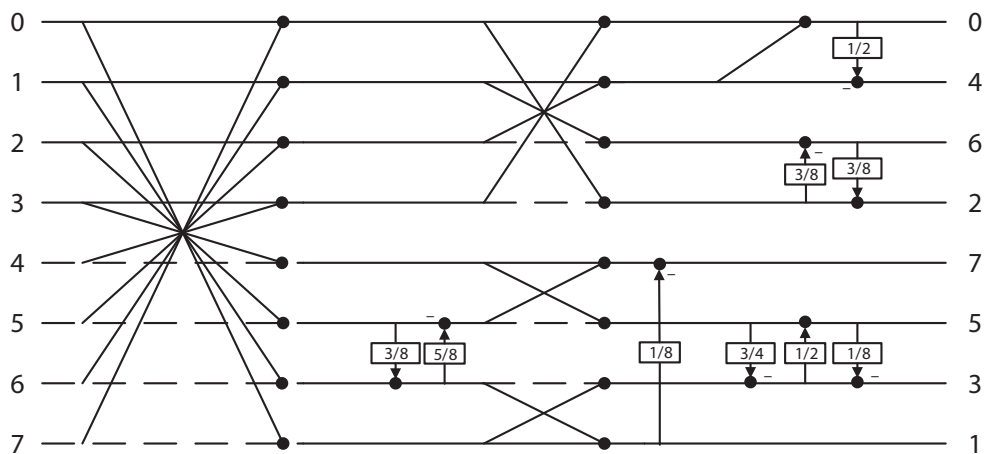
Fig. 1.15: 8-point BinDCT algorithm version B [24]

Note that binDCT is approximation of standard floating-point DCT. Best results are achieved when binDCT algorithms are used on both side encoder and decoder – it means forward binDCT is used as encoder while inverse binDCT is used as decoder. However if combined with standard floating-point DCT, compatibility between these transforms is satisfactory. [26]
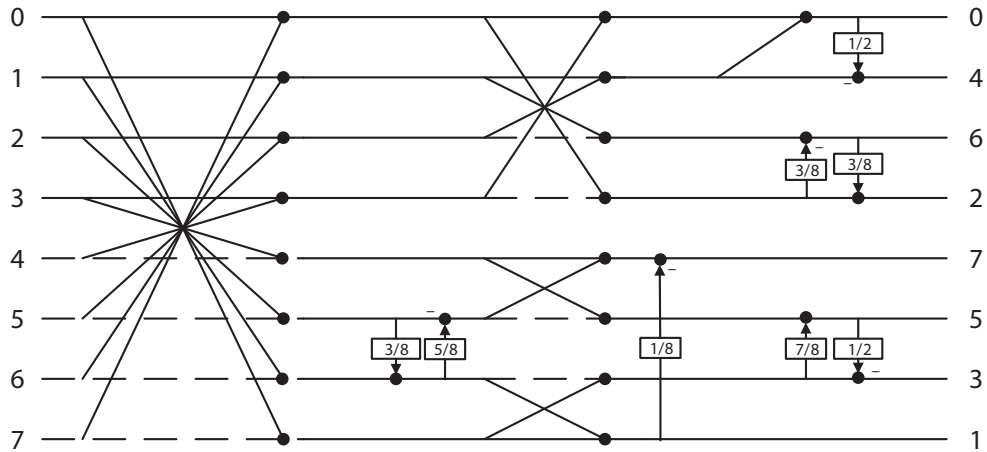
Fig. 1.16: 8-point BinDCT algorithm version C [24]
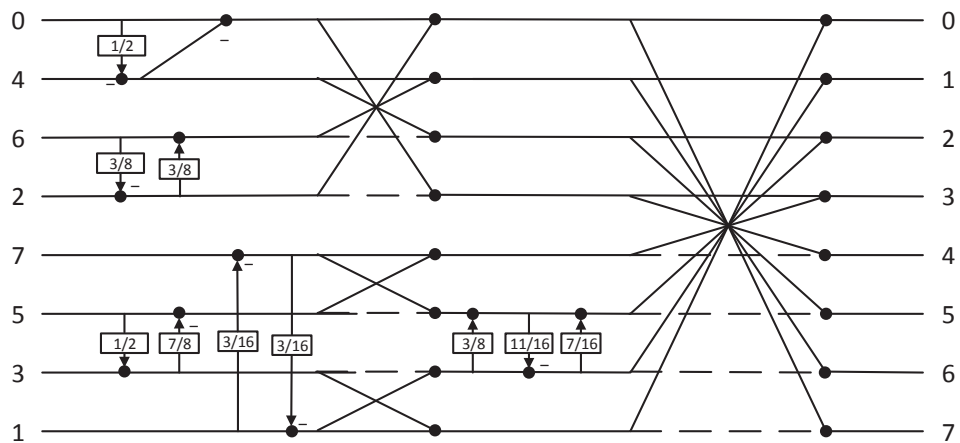
Fig. 1.17: 8-point Inverse BinDCT algorithm version A [24]

1.6 Algorithms for 2-D DCT

Two-dimensional DCT could be obtained with following methods [1]:

- 2-D DCT by reduction to 1-D DCT with row-column transposition
- 2-D DCT by reduction to 1-D DCT using lexicographical reordering
- Block matrix decomposed algorithm
- Computation via two-dimensional DFT
- Computation via two-dimensional WHT

Since algorithms using some other discrete transforms (DFT and WHT) are advantageous to be used if there is some flexibility required (for example DSP processor, not dedicated VLSI block)
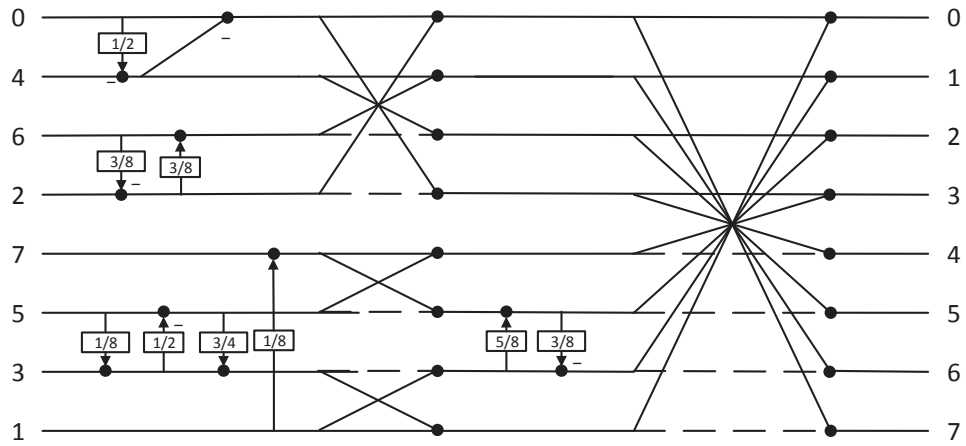
Fig. 1.18: 8-point Inverse BinDCT algorithm version B [24]
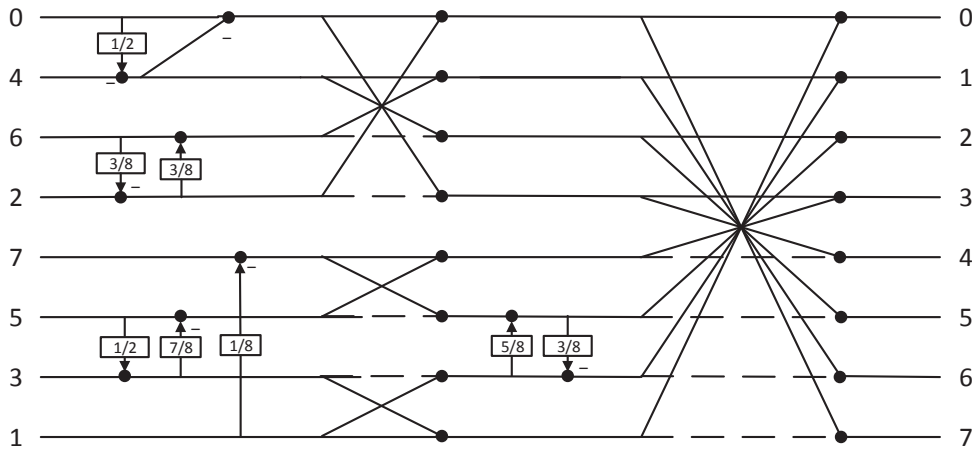
Fig. 1.19: 8-point Inverse BinDCT algorithm version C [24]

1.6.1 2-D DCT with row-column transposition

Due to separability property of two-dimensional DCT-II can be expressed by MN-point DCTs along the rows of $[g]$ (Equation 1.12) followed by NM-point DCTs along the comumns of the matrix obtained after the row transformation. Order of row-column transformation is theoretically immaterial. [1]

If desired to have one shared N-point DCT block[2], block diagram of DCT row-column approach is shown in Figure 1.20. Input data are loaded from $M \times N$ matrix by rows. Every N-point row is transformed by 1-D N-point row DCT block and loaded to transpose buffer. After filing transpose buffer and performing trans-

---

[2]For example if smaller chip area is a goal or designed block's purpose will be computing both one-dimensional and two-dimensional DCTs

position, data are input to 1-D N-point column DCT by columns. Timing control block controls correct data inputs and transpose operation.
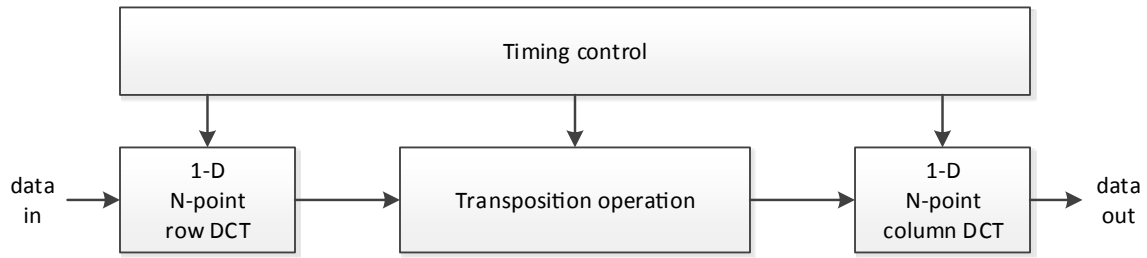
Fig. 1.20: Block diagram of the row-column approach for 2D DCT [1]

1.6.2   2-D DCT using lexicographical reordering

Using lexicographical reordering two-dimensional DCT can be reduced to only one-dimensional DCT. For example two-dimensional DCT of 4×4 matrix can be expressed as 16-point one-dimensional array. This algorithm require 24 real multiplications and 68 real additions. This method is fast and low latency can be achieved but due to its non-recursion calculation it has many parallel operations hence it is large in terms of area. [1]

1.6.3   Other methods for 2-D DCT

There are other methods for computing 2-D DCT coefficients for example using some other discrete transform like Discrete Fourier Transform (DFT) [27] or Walsh-Hadamard Transform (WHT).

Another method for 2-D DCT computation is block matrix decomposed algorithm. Unfortunately this method is more easily applied to Inverse DCT-II or the forward DCT-III.

1.7   Human visual perception

Most of methods for reducing bitrate of video or size of an image (or increasing compression ratio, if you want) are based on human visual perception model and its imperfections. This area is extremely complex and many of its properties are still not well understood. Many of these properties are not taken into account in MJPEG compression so they are not described in this section. [28] However, MJPEG compression benefits from following human visual perception imperfections:

- Human eye has low-pass filter characteristics (there are limited number of rods) so human visual perception is less sensitive to higher frequencies.
- Much greater sensitivity to luminance than color resolution. There two types of photoreceptors in the human retina, rods and cones. The rods are much more numerous than cones (120 milions against 6–7 milions) and they are more sensitive than cones. Rods provides sensitivity to luminance, whereas cones provides sensitivity to color. [29] Rods and cones density on human retina are shown on Figure 1.21.
- Since cones are concentrated in the center of the fovea, rods are responsible for peripheral vision.
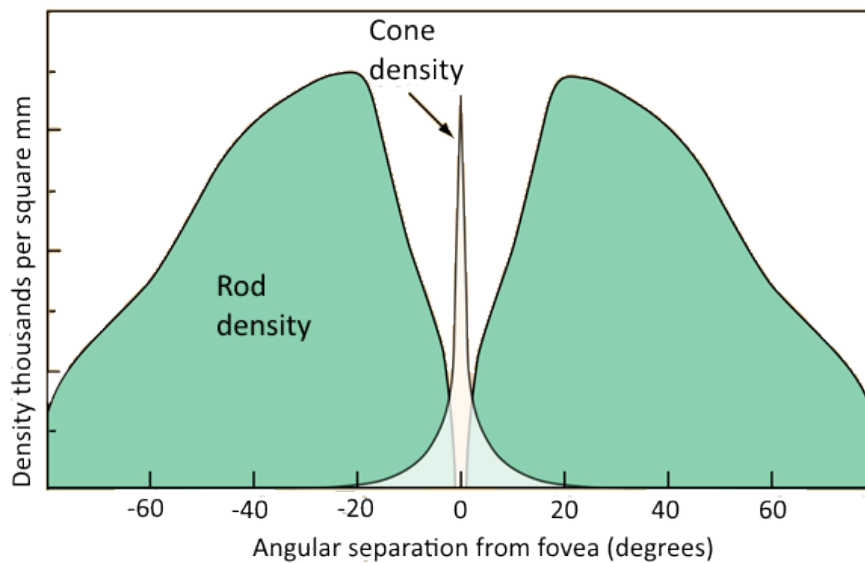
Fig. 1.21: Rods and cones density on human retina [29]

1.7.1 Color models

RGB color model

The color model named RGB is an additive color model. Its name is derived from three colors included - Red, Green and Blue. These colors are added together in various levels to reproduce hues of colors. This model is based on existence of three types of cones in the human eye (also known as *trichromacy*), each sensitive to particular wavelength of light. Meaning of additive color mixing is shown on Figure 1.22.
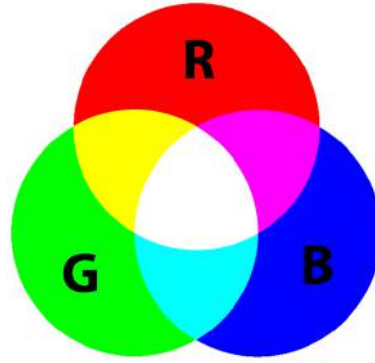
Fig. 1.22: Additive color mixing

YCbCr color model

In the YCbCr model color is composed from 3 components: Y – luminance (or brightness of the pixel)and two chroma components Cb – blue chrominance, Cr – red chrominance, whose can be converted from RGB space by formulas in Equation 1.26. [30]

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

$$C_b = 128 - 0.1687 \cdot R - 0.3313 \cdot G + 0.5 \cdot B \tag{1.26}$$

$$C_r = 128 + 0.5 \cdot R - 0.4187 \cdot G + 0.0813 \cdot B$$

And YCbCr color model can be converted to RGB vice versa by formulas described in Equation 1.27. [30]

$$R = Y + 1.402(C_r - 128)$$

$$G = Y - 0.34414(C_b - 128) - 0.71414(C_r - 128) \tag{1.27}$$

$$B = Y + 1.772(C_b - 128)$$

1.7.2 Chroma subsampling

Subsampling of chroma matrices is one of basic lossy technique of encoding images. This is done by implementing less resolution for chrominance matrix (or chroma information) than for luminance matrix. As described in section 1.7, human visual perception is much more sensitive to luminance than chrominance therefore chrominance information could have less resolution without significant degradation

of visual image quality. [28] It is used in both analog and digital video encoding schemes.
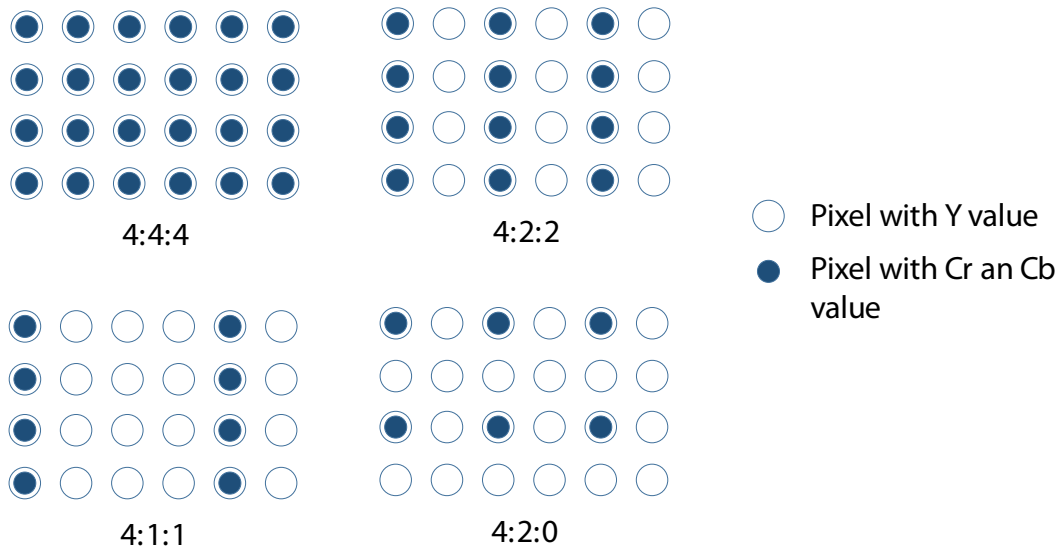
Fig. 1.23: Chroma subsampling schemes

In standard RGB color model without conversion to YCbCr model no chroma subsampling can be applied – RGB has to be converted to YCbCr by formulas in Equation 1.26. Then subsampling is applied only to chrominance information. Scheme for subsampling is usually expressed using three part ratio `J:a:b` (when alpha channel is not present as one component):

Where:

- `J` is horizontal sampling reference
- `a` is number of chrominance samples
- `b` is number of changes of chrominance samples between first and second row of `J` pixels

Most used subsampling schemes are shown in Figure 1.23. Subsampling schemes have different types of sitting depending on compression standard used. This is not subject of this master's thesis. MJPEG standard supports both 4:2:2 and 4:2:0 schemes.

1.8 Number representation

1.8.1 Q number format

Q is fixed point number format. Number of fractional and integer bits is specified. For example Q12 number has 12 fractional bits only no integer bits, Q8.0 has no

fractional bit and 8 integer bits, Q13.7 has 7 fractional bits and 13 integer bits. Number of bits is without sign bit. [31]

Numbers are stored as regular binary numbers (signed integers), but according to its fractional part it allows to perform rational number calculations.

For a given Q$m.n$ format applies:

- it is using $m + n + 1$ bits
- its resolution is 2^{-n}
- its range is $[-(2^m), 2^m - 2^{-n}]$

So for example format Q13.7 has:

- 21 bits
- resolution 0.0078125
- range $[-8192, 8191.9921875]$

2 PRACTICAL PART

Practical part is focused on defining some necessary parameters of DCT block. Also, DCT algorithm choice is discussed and its model in language C is described and compared with reference definition. After verification of model, RTL description of chosen algorithm is depicted from bottom up. There is verification and synthesis section at the end of the practical part.

2.0.1 Hardware prerequisites

Designed hardware has to be able to process data provided by digital IP camera. Output from camera has resolution 640×480 pixels and frame rate is 30 fps. We expect that raw data from camera sensor have been already de-interpolated from Bayer-mask values to RGB values and then converted to YCbCr space. Chrominance matrices of data have same resolution as luminance in worst case, so chrominance subsampling is not considered.

Data throughput of designed hardware depending on video resolution and number of fps can be calculated by Equation 2.1. For video with 640×480 pixels and frame rate 30 fps value 27.65 is obtained. Table 2.1 summaries hardware requirements for DCT calculation of various resolutions used in consumer media and TV broadcasting.

$$N = \frac{3 \cdot (x \cdot y) \cdot n_{\text{fps}} \cdot d}{10^6} \tag{2.1}$$

where

- x is number of horizontal pixels
- y is number of vertical lines
- n_{fps} is number of frames per second
- d is number of bits per pixel

2.0.2 DCT algorithm choice

As described in the section 1.4 above, some techniques and alg. For DCT are suitable for software, hardware implementation or both. Since main requirement of hardware is to be fast, small and has low power consumption, this is better fullfilled with fixed point arithmetics rather than floating point arithmetics. [32]

Loeffler's algorithm for computation of DCT is effective in terms of number of multiplications and additions needed. Its floating point implementation is not effective in term of area and speed. Similar results can be achieved with implementation

Tab. 2.1: Hardware requirements for various resolutions

Format	x dim.	y dim.	Chroma subs.	fps	Mbit/s
QVGA	320	240	4:4:4	30	55.30
VGA	640	480	4:2:0	30	110.59
VGA	640	480	4:4:4	30	221.18
720p	1280	720	4:2:0	25	276
1080p	1920	1080	4:2:0	25	622
2K	2048	1080	4:2:0	25	664
4K	4096	2160	4:2:0	25	2654
8K	7680	4320	4:2:0	25	9953
8K	7680	4320	4:2:2	25	13271

of one of binary DCT algorithms whose requirements for hardware resources are compared in Table 2.2.

Tab. 2.2: Comparision of 8-point binDCT algorithms [24]

Algorithm	No. of add.	No of. shifts	Coding gain
binDCT-A	36	19	8.82 dB
binDCT-B	31	14	8.77 dB
binDCT-C	30	13	8.77 dB

Since implementation of binary DCT version C has lowest number of adders and shifts needed with preservation of coding gain which is similar to standard DCT (described in section 1.5.2), it has been chosen to be implemented in this master's thesis.

2.0.3 Design methodology

Choosing effective methodology is important part of design on which future modifications will be depend on. Process diagram of design is shown in Figure 2.1. It has 3 levels of design – **System design**, **RTL design** and **Synthesis**, which is part of Physical implementation.

Firstly, conceptual model written in C/C++ language is made. It is verified against specification. When any bugs are found or model's output and expected behavior does not met specification, concept can be easily and flexibly changed by

System design   RTL design

Write conceptual C/C++ model → Partition design

Verify against specification → Create reference stimulus → Write each block in HDL

Write testbenches → Reverify

Constraints → Synthesize

To implementation
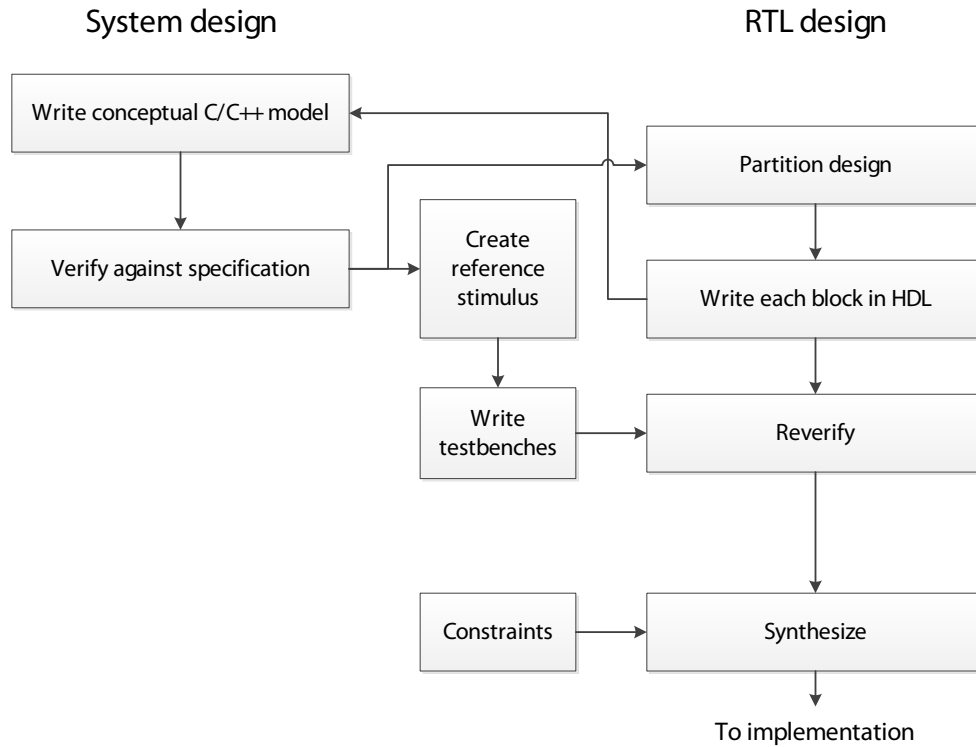
Fig. 2.1: Design process diagram [33]

modifying higher-level hardware description. Once C model outputs are correct and specifications are fulfilled, reference stimulus for RTL design is created.

Then existing conceptual model is designed and described by HDL Verilog language. When any change needs to be done to concept, re-writing of conceptual model is necessary. After re-verification against reference stimulus created previously by model is done, next step of design can be advanced – Synthesis.

Synthesis is process when HDL description in Verilog is synthesized to netlist of available cells and gates in specific process technology. These cells are provided by vendor in library. Synthesizer is software tool for doing this and its algorithm is trying to met provided constraints.

2.1 Hardware modeling

One of advantages of creating executable model is that no external tool needs to be launched when simulating. It is used only for compiling so once model is made, it is packed in one executable file. Basically, modeling by executable program written in C can be done using standard C/C++ libraries or SystemC library. SystemC libraries are developed by Accellera Initiative as Open Source Licence. [34] Their

comparison is shown in Table 2.3. SystemC is C++ based library which adds support for describing more hardware-like behavior and time based simulation as well as custom precision data types. However, these features complicate development of models and could be counterproductive for small design.

Tab. 2.3: Comparision of C/C++ and SystemC modeling methodology

	C/C++ language	SystemC library
Synthesizability	complicated	easier
Suitable for design size	small	medium & large
Time based	no	yes
Custom data types	no	yes

Even SystemC has many advantages over standard C/C++ libraries, its higher complexity could slower design process in final when designing smaller system. For this reason, modeling using standard C/C++ libraries has been chosen as this step in design workflow.

2.1.1 Precision modeling

As was discussed in section 2.0.2, target RTL implementation is going to use fixed point arithmetics. This could be easily modeled by double-precision floating point format specified by the IEEE 754 which is part of C programing language. [35]

2.1.2 One-dimensional bin DCT modeling

Principle of modeling one-dimensional binDCT algorithm is shown in Figure 2.2. Input one-dimensional array consisting of 8 samples in double precision is calculated according to Figure 1.16 in section 1.5.2. Intermediate calculation is done in double precision resolution. Because all multiplying is two-based and can be done by simple bit-shift, division by two is modeled in double precision. There is *round half to even* implemented on the output of the chain.

Input 8x1 array → 1-D binDCT (double precision) → Round half to even → Output 8x1 array

Fig. 2.2: One dimensional binDCT model flowchart

2.1.3 Two-dimensional bin DCT modeling

As described in Section 1.6.1 two-dimensional DCT can be done by row-column transposition of one-dimensional DCT. Figure 2.3 shows its principle. Input *integer* values are converted to *double* an then column-by-column brought to input of 1-D binDCT function. Its output is rounded and input to temporary array of size 8×8. Then samples are read out in different order – by rows and on the output of one-
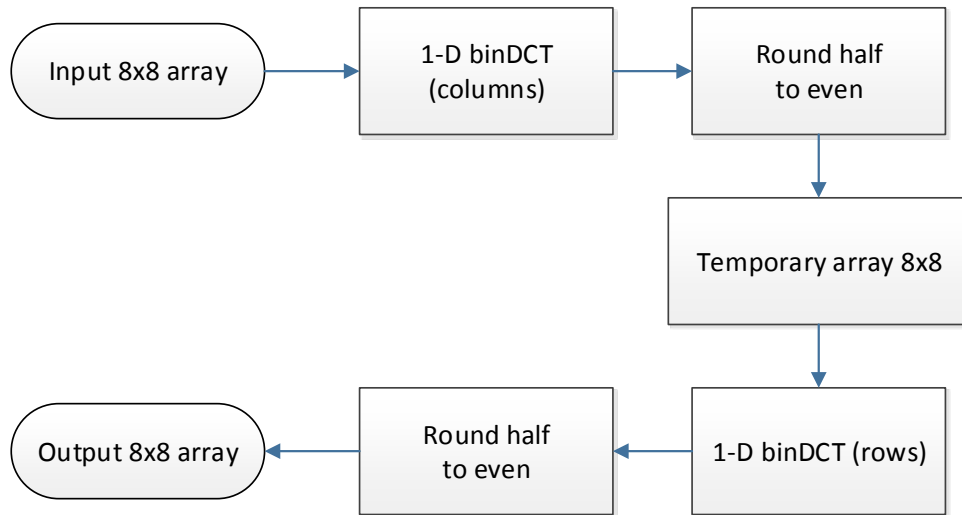
Fig. 2.3: Two dimensional binDCT model flowchart

dimensional binDCT model are rounded again. Output 8×8 array is a result of C model.

Output rounding

Using standard rounding values are always rounded in the same direction. Result can be a bias that grows with every rounding operation. At the output of the one-dimensional binDCT model data are rounded by *half to even* algorithm (often called *banker's rounding*). This is illustrated in Figure 2.4. Half-way values are always rounded toward the nearest even number. Major advantage of this technique is minimization this bias. [36]

2.1.4 Reference data comparison

Since binDCT output coefficients are not compatible with standard DCT as described by MATLAB definition, metric of comparing these two output is shown in Figure 2.5. Input image data which consists of unsigned 8 bits resolution pixels (hence its value is from 0 to 255) are decomposed into 8×8 blocks and then level
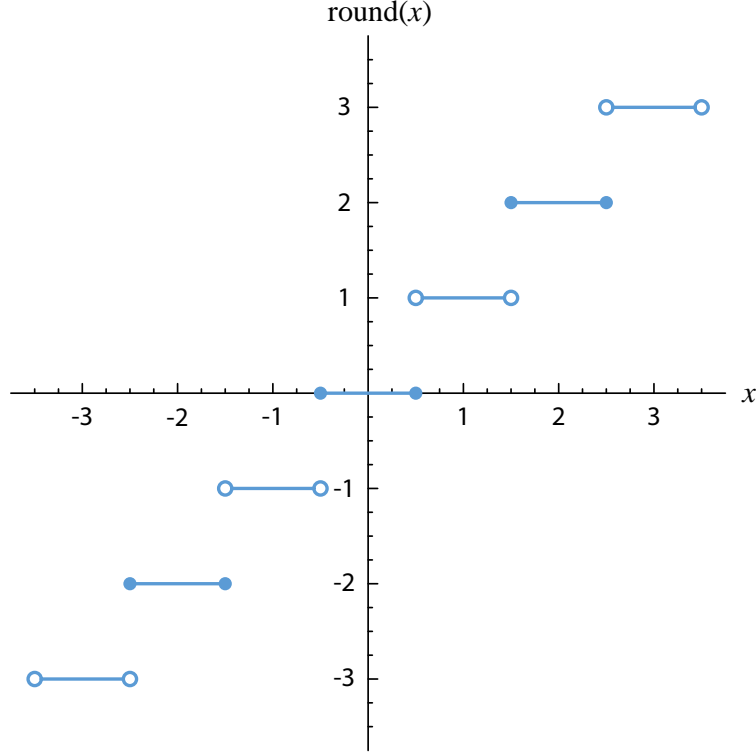
Fig. 2.4: Rounding half to even function

shifted to 8-bit signed values by subtracting 128 as defined in JPEG specification.
[37]

Level shifted 8×8 blocks are input to binDCT model and MATLAB DCT
reference, then rounded and inverse transform is done on both outputs. Regarding
to Inverse binDCT algorithm, its output samples are scaled by constant 4 so besides
inverse level shift, scaling (dividing each value by 4) is done. After process on each
block 8×8 pixels is done, these blocks are composed back to whole image data and
PSNR calculation is done for enumerating of reconstruction error.

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\hat{Y}_{ij} - Y_{ij} \right)^2 \tag{2.2}$$

$$\text{PSNR} = 10 \log \frac{255^2}{\text{MSE}} \, (\text{dB}) \tag{2.3}$$

Firstly, *Mean-Squared Error* – MSE calculation of output data and input image
data are done as described in Equation 2.2. Then *Peak Signal-to-Noise Ratio* is
calculated as described in Equation 2.3.

Note that this methodology only compares performance of various DCT and
IDCT algorithms, no its usage in JPEG Interchange File Format including quanti-
zation (depending on quality set) and coding.

Bin DCT → Rounding → Bin IDCT → Scale & level shift → Rounding

8x8 block composition → Output image data

Input image data → 8x8 block decomposition → Level shift

PSNR comparison

8x8 block composition → Output image data

MATLAB DCT → Rounding → MATLAB IDCT → Level shift → Rounding
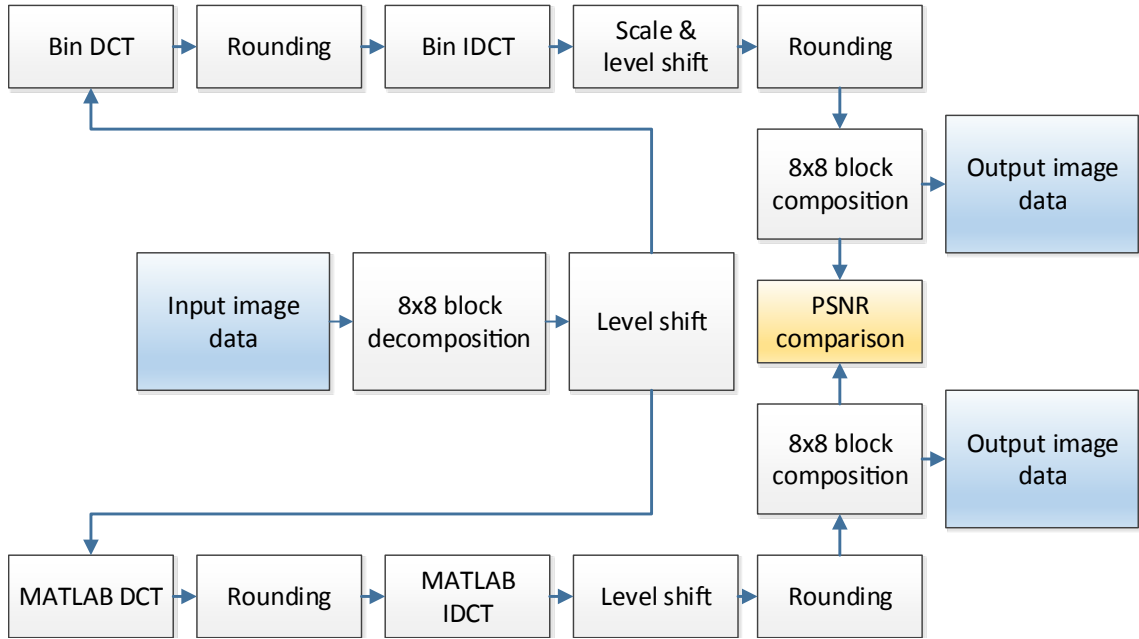
Fig. 2.5: Comparing model with reference MATLAB definition

Result of model and reference comparison is shown in Figure 2.6a for output of Matlab DCT, Figure 2.6b and Figure 2.6c. Because of any changes are not noticeable visually in the output images, Figures 2.7a, 2.7b and 2.7c show error representation between original image and output from DCT algorithms. These results has only illustration information. Pixels with error are represented as white, pixels without error are black. It is obvious that output from BinDCT chain has much less errors than Matlab DCT and binDCT transformed via IDCT algorithm. Values of calculated PSNRs are summarized in Table 2.4. Matlab DCT performs value 58.91 dB, binDCT PSNR is 62.96 dB and binDCT via IDCT performs 33.70 dB when output rounding (after inverse transform) is not considered. However when output rounding is implemented, PSNR value of Matlab DCT chain is slightly improved to 58.93 dB binDCT via IDCT drops to 33.69 dB whereas value of BinDCT chain is dramatically improved to 70.80 dB.

One of reasons of this binDCT advantage over Matlab DCT could be that its coefficients are not scaled like Matlab's. Another reason could be incorrectly derived Inverse BinDCT algorithm which has gain of 4 when compared to input samples. This gain could brought advantage over Matlab definition.

Figure 2.6c shows result when binDCT algorithm is used on the coder side and standard IDCT algorithm is used on the decoder side. Its result is noticeably worse than other two benchmarks, but shows backward compatibility with standard IDCT. Whereas without quantization its results are worse, when quantization is imple-

(a) DCT via IDCT

(b) binDCT via binIDCT

(c) binDCT via IDCT

Fig. 2.6: Test image generated by various DCT algorithms
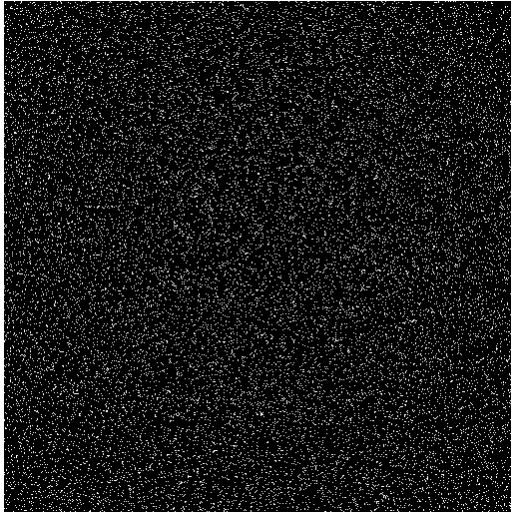
(a) DCT via IDCT
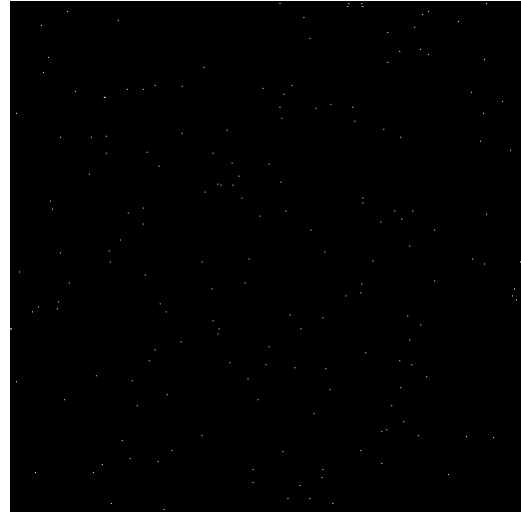
(b) binDCT via binIDCT

(c) binDCT via IDCT

Fig. 2.7: Error representation of various DCT algorithms

mented (JPEG quality ≤ 90) PSNR values are comparable. [24]

Tab. 2.4: Comparision of DCT algorithms' PSNR

Algorithm	PSNR [dB]	PSNR (rounding) [dB]
DCT by IDCT	62.96	70.80
BinDCT by binIDCT	58.91	58.93
BinDCT by IDCT	33.70	33.69

2.2   RTL

Once requirements are met and model's output is fulfilling expected behavior, RTL shall be described by Verilog HDL language. Firstly RTL design of one-dimensional bin DCT is written and then verified against previously created C model.

2.2.1   One-dimensional DCT block

RTL implementation of binDCT algorithm needs define bit width of internal signals.

Resolution of internal signals

Because of no intermediate rounding is implemented between operations in model, all internal signals within block for computing one-dimensional binDCT have full accuracy.
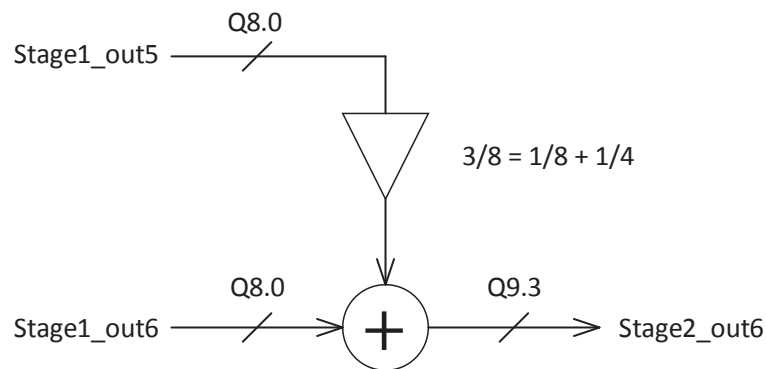
Fig. 2.8: Arithmetical operation within 1-D binDCT block

Methodology of defining this bit resolution in fixed point arithmetic is illustrated in Figure 2.8 and 2.9. 8-bit resolution of input samples is considered. After coefficient 5 and 6 is being calculated, it is followed by bit shift of sample 5 (3 bits to

the right) and thereafter adding operation with sample 6. Resulting resolution is 13 bits (10 bits for integer and 3 fractional bits).
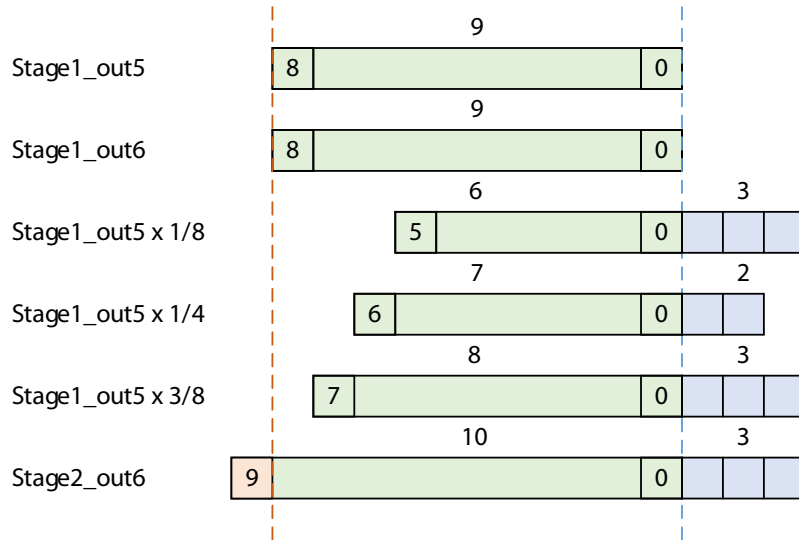
Fig. 2.9: Defining bit width of internal signals methodology

Bit resolution derived for all internal signals in 1-D binDCT block is shown in Figure 2.10. This diagram consider block parametrization of N-bit input resolution. Output coefficients have different resolution.

RTL parametrization

Block described in Verilog is fully parameterized in the terms of input samples bit resolution – N and registering between stages – REG. Combinational logic of 1-D binDCT block is divided into 6 stages. Every stage has its own parameter which describes if this stage has register on the output. Length of combinational logic can be parametrized thus some registers can be saved in signal path when smaller speed is a goal.

Clock and reset signal is brought to every stage instance. Each instance of stages has different input and output resolution as described in Figure 2.10.

Block interface

Block interface is illustrated in Figure 2.12. In case of 8-bit resolution of input data, input vector of first 1-D binDCT block is 64-bits wide while output vector is 104-bits wide. Second block has 104-bits wide input and 144-bits wide output. With every clock rising edge 8 coefficients are calculated if there is at least one register at design
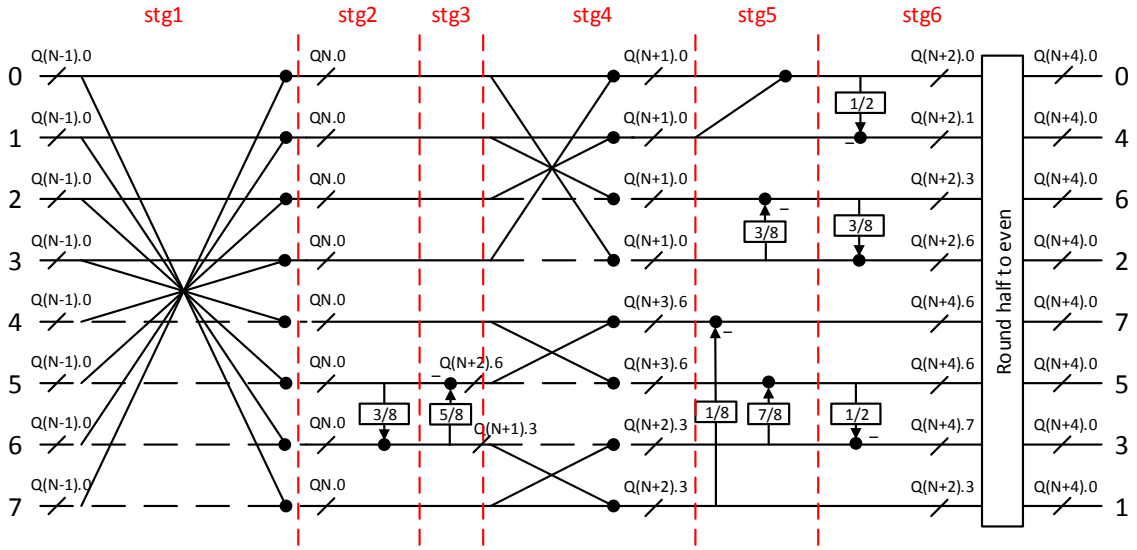
Fig. 2.10: Defining bit width of internal signals of 1-D DCT block

defined by parameter. Input and output valid data indication is controlled by FSM in upper module.

Cadence Incinsive HDL analysis

Incinsives HDL analysis helps find errors early in design process, before simulation. This tool identifies coding errors and improper RTL design styles through a comprehensive analysis of HDL source code. [38] HAL analysis was done without errors and its results are summarized in Table 2.5.

Tab. 2.5: 1-D binDCT design hierarchy summary (Cadence HAL)

	Instances	Unique
Modules	7	7
Registers	36	36
Scalar wires	3	-
Vectored wires	112	-
Always blocks	13	14
Cont. assignments	63	152
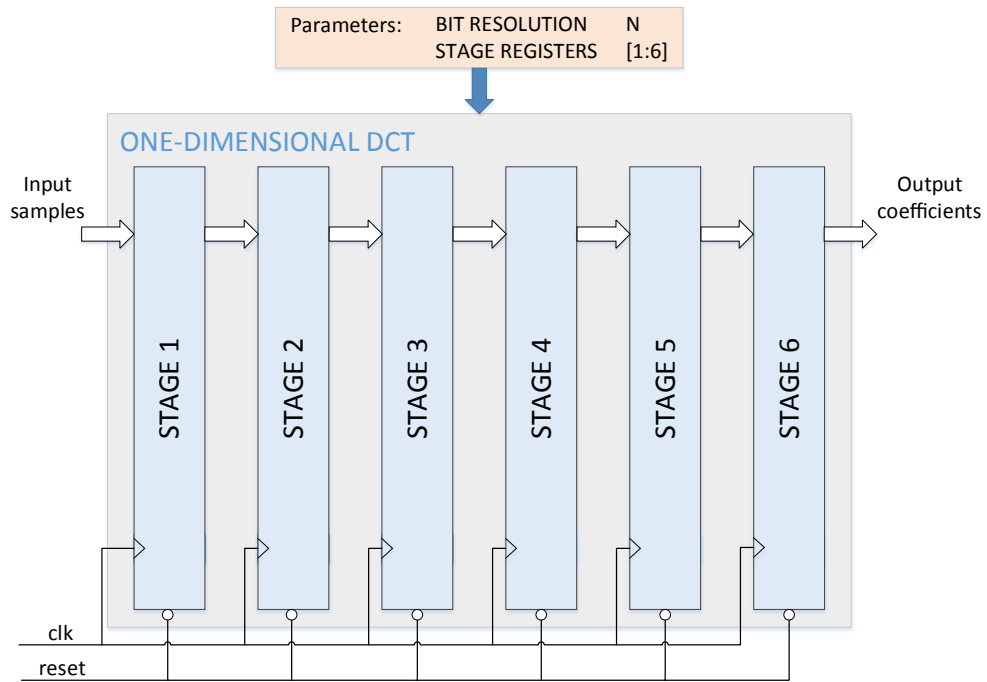
Fig. 2.11: Block diagram of 1-D binDCT RTL design
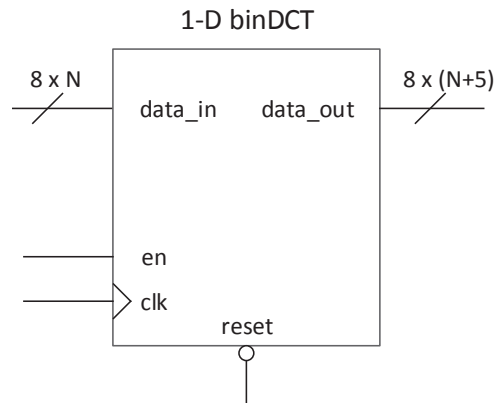
Fig. 2.12: Interface 1-D binDCT RTL design

Computation latency

Since one-dimensional binDCT block has register configuration set by parameter, its latency is variable and depended on number of registered stages. When all stages has register on their output, block's computation latency is 6 clock cycles. However, when block is generated without any registers (although it is not recommended), block has latency given only by combinational logic path.

2.2.2 Two-dimensional DCT block

Block interface

Figure 2.13 shows interface of 2-D binDCT block. There is 64-bit wide signal on the input – `data_in` (in case that input resolution is 8-bit per sample) and 144-bit wide signal on the output – `data_out`. All handshake signals are active high. Input signal named `dv_in` indicates valid data in the input, output signal `dv_out` indicates valid data in the output.

Signal `clk` is rising-edge sensitive, `reset` is asynchronous active low.
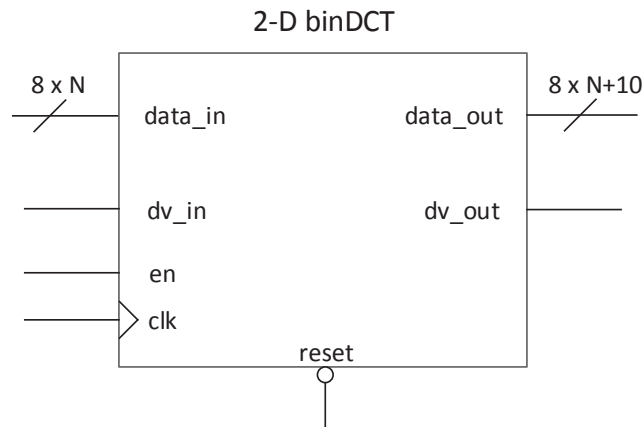
Fig. 2.13: Interface of 2-D binDCT RTL design
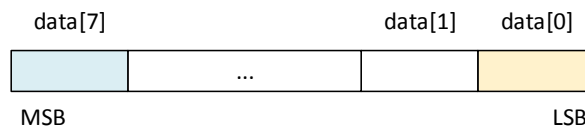
Fig. 2.14: Data order of input and output samples row/column vector

Order of input data in terms of rows/columns is immaterial. This is described in Figure 2.15. When input vector contains row samples, output vector contains column coefficient in different order – last is first.

Module overview

Figure 2.16 illustrate schematic of 2-D binDCT module. There are two blocks for computing one-dimensional binDCT, each has different bit resolution. Data are continuously input to the first 1-D block. Its output is de-multiplexed between two transpose buffers. These complementary buffers allow continuous operation of both

49

Fig. 2.15: Data order of input and output samples in 8×8 block
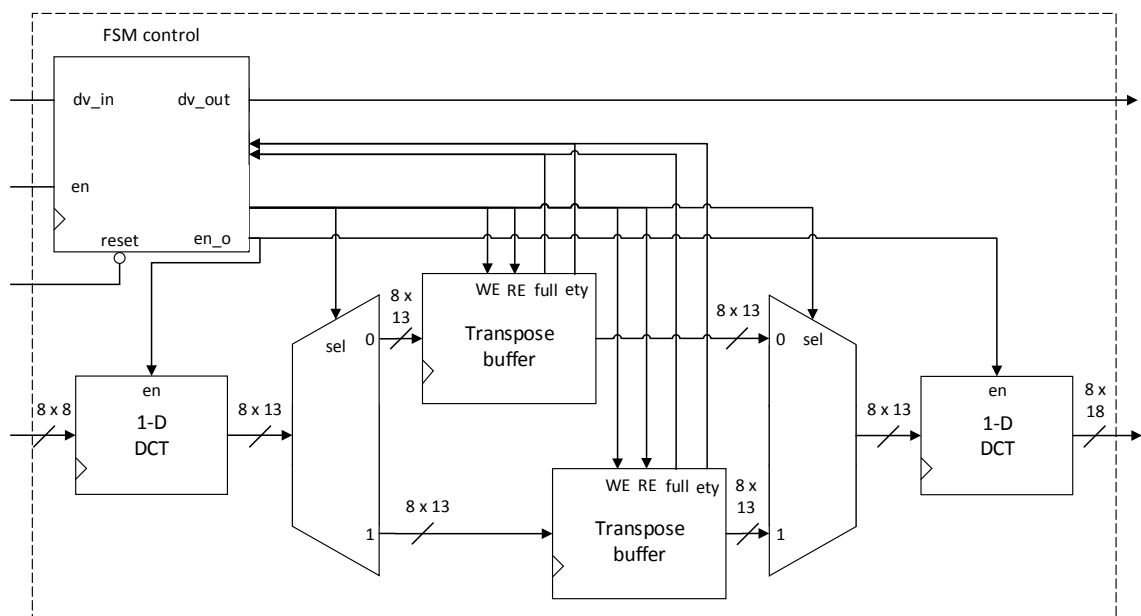
Fig. 2.16: Schematic of 2-D binDCT block

1-D blocks hence 8-samples are computed each clock cycle. Alternating of WE and RE signals is controlled by FSM as well as sel signal of multiplexor and de-multiplexor. This architecture allows continuous operation and high speed computation can be achieved.

Block latency

Total latency done by two blocks for calculating binDCT and transpose buffer can by calculated by Equation 2.4 where R is number of registers in one-dimensional binDCT block.

$$N_{\text{clk}} = 2 \cdot R + 8 \; [\text{clk cycles}] \tag{2.4}$$

De-multiplexor and multiplexor

These components are created using combinational logic only. Both have parametric bit width of input hence their combinational logic is generated according to top module settings.

Transpose buffer

Figure 2.17 shows transpose buffer module for configuration with 8-bit input resolution per sample. Its operation modes depending on inputs configuration are summarized in Table 2.6. It contains 3-bit up counter counting clock cycles and after reaches value 7 (8 clock cycles were done), `ety` or `full` flag is set active depending on current mode set by `WE` or `RE`.
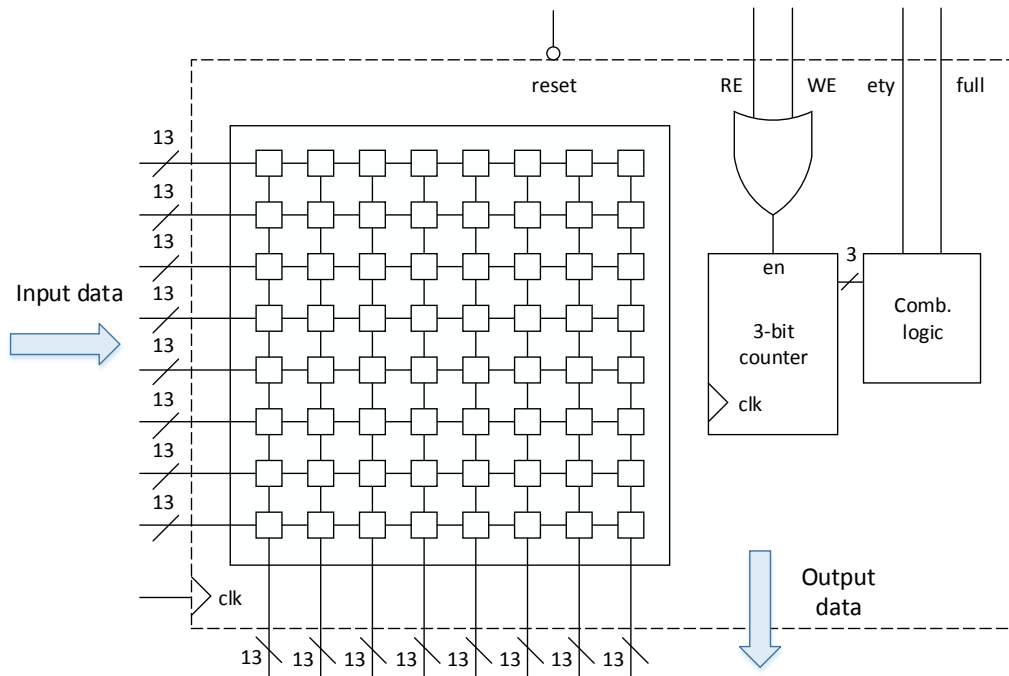
Fig. 2.17: Block diagram of 8×8 transpose buffer

Tab. 2.6: Operation modes of Transpose buffer

WE	RE	Description
0	0	Input data are ignored. All outputs set to zero.
0	1	Read Enable. Input data are ignored. Output data are sample-shifted with clock rising-edge from bottom down.
1	0	Write Enable. All outputs set to zero. Input data are sample-shifted from left to right. Data from 8th column are discarded with every clock rising-edge.
1	1	Write Enable has higher priority than reading. Function is same as previous.

FSM control
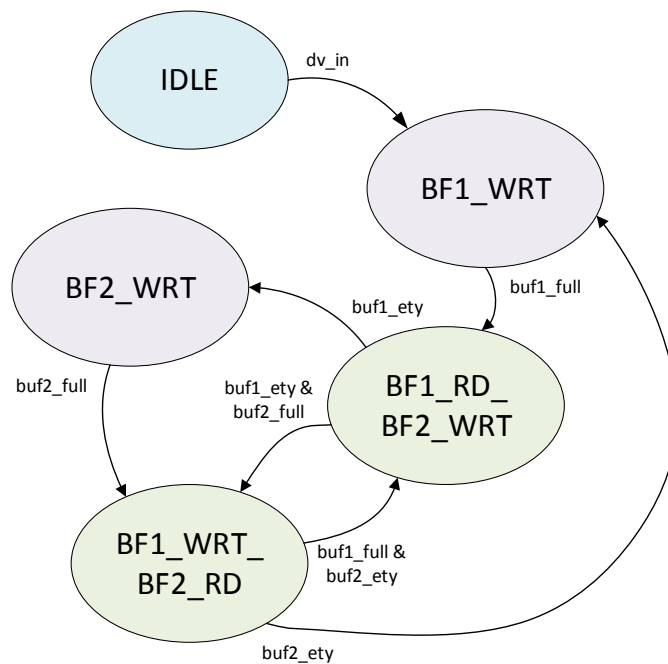
Fig. 2.18: FSM diagram

Diagram of FSM control is shown in Figure 2.18. FSM description, coding and output states are shown in Tables 2.7 and 2.8. Although binary coding is relatively slow, it has lowest number of flip-flops hence it is suitable for ASIC implementation.

After `reset` is de-asserted FSM remains in state `IDLE`. When input `dv_in` is set to '1' (which mean that there are valid data in the input), FSM switches to state `BF1_WRT` and after calculating column DCT coefficients, these are written to

transpose buffer 1. When buffer is filled with intermediate calculations, its output named `buf1_full` goes active and FSM next state is set to `BF1_RD_BF2_WRT`. There are two possibilities of FSM transition from this state – `BF2_WRT` (which is set when reading from buffer 1 is done by setting signal `buf1_ety` before writing to buffer 2 is completed) or `BF1_WRT_BF2_RD` (which is set when reading from buffer 1 is completed together with writing to buffer 2 by setting signals `buf1_ety` and `buf2_full` to '1'). This transition is done when there are maximum data rate in the input.

In all states except `IDLE`, `WE` (*write enable*) of both transpose buffers is driven by delayed `dv_in` input signal. Number of D flip-flops (which provide delay) is generated according to number of registers set by parameter.

Tab. 2.7: Outputs of FSM depending on the state (part 1 of 2)

State name	Code	Outputs	Description
IDLE	0'b000	dv_out = '0' buf1_we = '0' buf1_re = '0' buf2_we = '0' buf2_re = '0' demux_sel = '0' mux_sel = '0' dct1_en = '0' dct2_en = '0'	Idle state. Waiting for input data.
BF1_WRT	0'b001	dv_out = dv_o_reg buf1_we = dv_i_del buf1_re = '0' buf2_we = '0' buf2_re = '0' demux_sel = '0' mux_sel = '1' dct1_en = '1' dct2_en = '1'	Data from DCT1 block are written to Buffer1.

Tab. 2.8: Outputs of FSM depending on the state (part 2 of 2)

State name	Code	Outputs	Description
BF1_RD_BF2_WRT	0'b010	dv_out = dv_o_reg buf1_we = '0' buf1_re = '1' buf2_we = dv_i_del buf2_re = '0' demux_sel = '1' mux_sel = '0' dct1_en = '1' dct2_en = '1'	Data from DCT1 block are written to Buffer2. Data from Buffer1 are read and input to DCT2 block.
BF1_WRT_BF2_RD	0'b011	dv_out = dv_o_reg buf1_we = dv_i_del buf1_re = '0' buf2_we = '0' buf2_re = '1' demux_sel = '0' mux_sel = '1' dct1_en = '1' dct2_en = '1'	Data from DCT1 block are written to Buffer1. Data from Buffer2 are read and input to DCT2 block.
BF2_WRT	0'b100	dv_out = dv_o_reg buf1_we = '0' buf1_re = '0' buf2_we = '0' buf2_re = '0' demux_sel = '0' mux_sel = '0' dct1_en = '0' dct2_en = '0'	Data from DCT1 block are written to Buffer1.
others	0'b101 0'b110 0'b111	dv_out = '0' buf1_we = '0' buf1_re = '0' buf2_we = '0' buf2_re = '0' demux_sel = '0' mux_sel = '0' dct1_en = '0' dct2_en = '0'	Unused combinations.

2.3 Verification

2.3.1 Verification methodology

RTL description of design needs to be verified. Verification methodology and flow is illustrated in Figure 2.19. Testbench environment loads testcases where are defined input stimuli and output reference data, which could be loaded from file.

Input stimuli data are brought to DUT (Design Under Test) input then DUT output is compared with output reference data and if there is any mismatch, testcase's status is marked as *failed* otherwise is marked as *passed*.
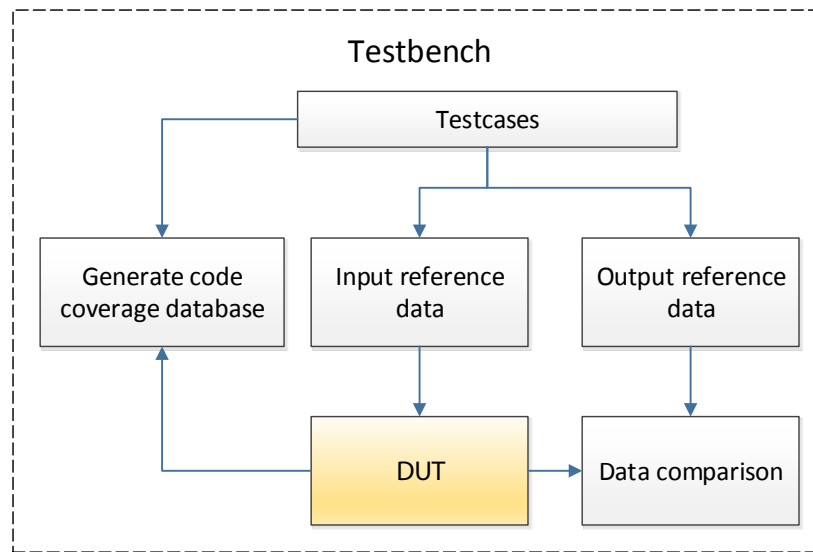
Fig. 2.19: Testbench methodology and flow

Addition to this output monitoring is also creating of code coverage database based on input stimuli data and DUT. Code coverage is a measure used to describe the degree to which the source code is tested by a particular testcase.

2.3.2 Code coverage

Once code coverage database is generated, it is opened by software tool, which allows enumerate its results. Incinsive Code Coverage Report (ICCR) is one of these tools. Reports could be obtained as standard textfiles or HTML files. ICCR allows quantify degree of code coverage of whole DUT as well as each instantiated module or sub-block.

Figure 2.20 shows code coverage report for top module and its sub-modules. Total code coverage 91% has been achieved. Most of sub-modules have code coverage

Overall Module-Based Coverage

Total	Block	Expression	Toggle	FSM
91%	94% (126/134)	95% (80/84)	98% (4186/4252)	76% (13/17)

Coverage Summary Report, Module-Based

Total	Block	Expression	Toggle	FSM	Name
98%	No Items	No Items	98% (221/225)	No Items	s3_2d_dct
100%	100% (3/3)	100% (2/2)	100% (521/521)	No Items	s3_demux_8to16
100%	100% (3/3)	100% (2/2)	100% (417/417)	No Items	s3_mux_16to8
98%	96% (25/26)	100% (27/27)	99% (218/219)	No Items	s3_8x8_reg
99%	No Items	No Items	99% (540/546)	No Items	s3_1d_dct
99%	100% (5/5)	100% (2/2)	98% (137/139)	No Items	stage1
99%	100% (5/5)	100% (2/2)	99% (301/303)	No Items	stage2
99%	100% (5/5)	100% (2/2)	99% (329/331)	No Items	stage3
99%	100% (5/5)	100% (2/2)	99% (397/399)	No Items	stage4
99%	100% (5/5)	100% (2/2)	97% (426/440)	No Items	stage5
99%	100% (39/39)	100% (14/14)	96% (641/666)	No Items	stage6
81%	81% (31/38)	86% (25/29)	83% (38/46)	76% (13/17)	s3_2d_dct_fsm

Fig. 2.20: Code coverage report

nearly 100% except FSM module. DUT has been tested to continuous data flow (`dv_out` was active during testcase running time) so FSM has not passed to states where writing to transpose buffer is not present.

All stages of one-dimensional DCT block have been covered at least 99% of transitions in total so they could be considered as successfully verified when compared to C model reference data.

2.4 Physical implementation

2.4.1 Synthesis workflow

Next step in design flow is first level of physical implementation – synthesis. As a software tool for synthesis is used dc shell version 2014.09-SP2 from Synopsys. Workflow of synthesis is shown in Figure 2.21. Constrains define rules of design and physical implementation – including target frequency, driving cells for input ports, load cells for outputs ports, maximum and minimum input delay, and much more options could be set.

Also, technological library have to be defined. It contains cells references available in current technology. Once Synthesis tool is run, after many optimizations, netlist of cells and report files are generated. Report contains information about

registers, estimation of power consumption, hierarchy, timing, references, area and if any exist, timing violators.

Implementation for ASIC utilizes different approach when compared to implementation for FPGA. When synthesis for FPGA is done, its output is maximum achievable frequency in target FPGA. Synthesizer for ASICs works differently. It is trying to achieve defined frequency with defined corners (maximum area, maximum power consumption, etc).
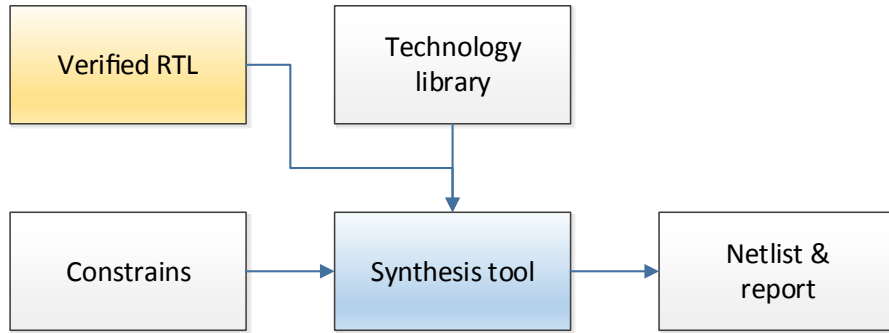
Fig. 2.21: Synthesis workflow

2.4.2 Synthesis results

Synthesis has been run for different target frequencies and register configuration depending on required performance of 2-D binDCT block. TSMC 65 nm technology process family includes General Purpose (GP), Low Power (LP), Ultra-Low Power (ULP) and LPG options. Each process supports low, standard, and high V_t options. Operating voltages range from 0.9 V to 1.26 V. [39] Low Power option has been chosen because it provides compromise between speed and power consumption.

Global operating voltage for all synthesis settings is 1.08 V.

Tab. 2.9: Power estimation for frequency 4 MHz and register configuration 000001

Power group	Internal power [mW]	Switch. power [mW]	Leakage power [nW]	Total power [mW]
clock network	3.1971e-02	3.2878	8.5846	3.3198
register	3.8694e-02	4.1302e-04	4.2204e+03	4.3327e-02
combinational	3.3019e-03	2.0206e-03	3.6300e+03	8.9526e-03
Total	7.3967e-02	3.2902	7.8591e+03	3.3721

Requirements for the block are to be able compute DCT of VGA resolution in 30 fps. As seen in Table 2.1, in worst case (without any chrominance subsampling) 221.18 MBit/s has to be computed. This is achieved when working frequency is 4 MHz. Table 2.9 shows power estimation and Table 2.10 shows area report after synthesis for clock frequency 4 MHz.

Tab. 2.10: Area report for frequency 4 MHz and register configuration `000001`

Number of ports	213
Number of nets	6066
Number of cells	5356
Number of combinational cells	3454
Number of sequential cells	1896
Number of buf/inv	641
Number of references	65
Area type	**Area** $[\mu\text{m}^2]$
Combinational area:	12232.80
Buf/Inv area:	734.04
Noncombinational area:	19374.48
Total cell area:	31607.28

When correctly configured, clock is also capable running at higher frequencies. This is achieved by registering output of every stage in 1-D DCT block (parameter REG = `111111`). Table 2.11 shows power estimation and Table 2.12 shows area report for clock frequency 300 MHz.

Tab. 2.11: Power estimation for frequency 300 MHz and register configuration `111111`

Power group	Internal power [mW]	Switch. power [mW]	Leakage power [μW]	Total power [mW]
clock network	2.38	244.59	8.58	246.97
register	2.90	3.12e-02	4289.2	2.94
combinational	0.26	0.16	4135.3	0.42
Total	5.54	244.78	8433	250.33

Tab. 2.12: Area report for frequency 300 MHz and register configuration 111111

Number of ports	213
Number of nets	7807
Number of cells	7007
Number of combinational cells	4039
Number of sequential cells	2962
Number of buf/inv	766
Number of references	79
Area type	**Area** $[\mu m^2]$
Combinational area:	13974.48
Buf/Inv area:	906.12
Noncombinational area:	30501.36
Total cell area:	44475.84

Power consumption and area for different register configuration and working frequency is summarized in Table 2.13. With working frequency 4 MHz DCT coefficients of VGA resolution video stream with 30 fps can be calculated. With working frequency 300 MHz DCT block is able to calculate video stream with 8K resolution (Table 2.1) with 25 fps and 4:2:2 chrominance subsampling scheme, because it requires bandwith 13271 MBit/s and block is able to compute 19200 MBit/s at this frequency.

Tab. 2.13: Summary of synthesis with different configuration and performance comparison

Frequency [MHz]	Registers	Total power [mW]	Area $[\mu m^2]$	Bandwidth [MBit/s]
1	000001	0.84	31607.64	64
4	000001	3.37	31607.28	256
10	000001	8.53	33750.36	640
50	111111	41.97	43095.95	3200
100	111111	83.46	43100.28	6400
300	111111	250.3	44475.84	19200

CONCLUSION

Goal of this master's thesis was to create theoretical summary of Discrete Cosine Transform (DCT) definition, computational principles, areas of usage, compare different algorithms for DCT computation and clarify its usage in compression method like JPEG and MJPEG. Practical goal was to create IP block for hardware acceleration of MJPEG compression of video data with resolution of 640×480 pixels and frame rate of 30 fps.

There is comparison of some known DCT algorithms in theoretical part. Two of them have been found as VLSI-perspective solution. First of them is called Loeffler's algorithm achieves theoretical minimum of arithmetical operations needed – 11 multiplications and 29 additions. It requires floating-point multiplication hence it is not effective in term of area and speed. Second of perspective algorithms is DCT approximation using lifting scheme also called as binary DCT algorithm. It requires no multiplication – these computation steps are approximated by binary shifts and additions only. For this reason bin DCT algorithms has been chosen to be implemented.

In first step model in C language has been created and then verified against MATLAB specification as reference. Then block has been described at RTL level using Verilog 2001 language. For usage with MJPEG compression which uses 8×8 blocks for DCT computation and 8-bit resolution per pixel, it requires 64-bit input vector of data. It consist from 8 input samples (no matter if it is row or column). After 10 to 20 clock cycles (depending on the register setting) there are 144-bit wide output vector consists from 8 output DCT coefficients.

Synthesis to TSMC 65 nm low power technology revealed block's performance, area and power consumption. Performance and power consumption is dependent on working clock frequency.

For accelerating video data with resolution of 640×480 pixels and frame rate of 30 fps, at least 110.6 Mbps throughput is needed when considering 4:2:0 chrominance subsampling scheme. Otherwise at least 221.2 Mbps is needed for 4:4:4 scheme. Block synthesized to 4 MHz target clock constrain is able to provide throughput up to 256 Mbps hence it allows acceleration VGA video for both considered chrominance subsampling schemes. Its area is $31607.64\,\mu\mathrm{m}^2$ and estimated power consumption is 3.37 mW.

With respect to maximum combinational path in design and synthesized technology, there are some limits with working frequency. When runs on maximum

frequency $300\,\mathrm{MHz}$, block is capable of data throughput $19.2\,\mathrm{Gbps}$ which is enough for acceleration of 8K video (7680×3840) pixels with 8-bits per pixel and $25\,\mathrm{fps}$. Chrominance subsampling scheme is considered as 4:2:2. Current consumer electronics trends head to use this resolution in digital television broadcasting. Block's area is $44475.84\,\mu\mathrm{m}^2$ and estimated power consumption is $250.3\,\mathrm{mW}$.

All goals of this thesis have been successfully fulfilled.

BIBLIOGRAPHY

[1] RAO, K and P YIP. *Discrete cosine transform: algorithms, advantages, applications.* Boston: Academic Press, 1990, xviii, 490 p. ISBN 01-258-0203-X.

[2] ZHONGDE W. Fast algorithms for the discrete W transform and for the discrete Fourier transform. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 1984, p. 803-816. ISSN 0096-3518. DOI: 10.1109/TASSP.1984.1164399. Available from: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1164399>

[3] AHMED, N., T. NATARAJAN and K.R. RAO. Discrete Cosine Transform. In: *IEEE Transactions on computers.* New York: IEEE Computer Society, 1974, p. 90-93. ISSN 0018-9340. Available from: <https://www.ic.tu-berlin.de/fileadmin/fg121/Source-Coding_WS12/selected-readings/Ahmed_et_al.__1974.pdf>

[4] LOEFFLER, C., A. LIGTENBERG and G.S. MOSCHYTZ. Practical fast 1-D DCT algorithms with 11 multiplications. *International Conference on Acoustics, Speech, and Signal Processing.* IEEE, 1989, vol.2, p. 988-991. DOI: 10.1109/ICASSP.1989.266596. Available from: <http://www3.matapp.unimib.it/corsi-2007-2008/informatica/calcolo-numerico/jpeg/papers/11-multiplications.pdf>

[5] Discrete cosine transform (DCT). *Mathworks documentation* [online]. 2015 [cit. 2015-05-03]. Available from: <http://www.mathworks.com/help/signal/ref/dct.html>

[6] Discrete Cosine Transform. 2015. *MathWorks Documentation* [online]. [cit. 2015-05-11]. Available from: <http://www.mathworks.com/help/images/discrete-cosine-transform.html>

[7] SOVIČ, D. Kompresný štandard JPEG. *Dušanové stránky o kopresii: Základné kompresné algoritmy a štandardy* [online]. 2000 [cit. 2014-12-17]. Available from: <http://pakuj.brek.sk/jpeg/jpeg.html>

[8] WALLACE, G. The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics.* 1991. Available from: <http://www-ee.eng.hawaii.edu/~treed/EE416/Project_4/jpeg-wallace.pdf>

[9] INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS

AND GUIDELINES. 1993. In: *CCITT T.81* [online]. [cit. 2015-05-05].
Available from: <http://www.w3.org/Graphics/JPEG/itu-t81.pdf>

[10] MPEG-2 Video. *The Moving Picture Experts Group* [online]. 2006 [cit.
2014-11-30]. Available from:
<http://mpeg.chiariglione.org/standards/mpeg-2/video>

[11] MPEG-4 Visual. *The Moving Picture Experts Group* [online]. 2005 [cit.
2014-11-30]. Available from:
<http://mpeg.chiariglione.org/standards/mpeg-4/video>

[12] MJPEG vs MPEG4. In: *Understanding the differences, advantages and
disadvantages of each compression technique: White Paper* [online]. 2006 [cit.
2014-11-30]. Available from: <http:
//www.onssi.com/downloads/OnSSI_WP_compression_techniques.pdf>

[13] Microsoft Windows Bitmap Format. *FileFormat.Info* [online]. 1993 [cit.
2014-11-30]. Available from: <http://www.fileformat.info/format/bmp/
spec/b7c72ebab8064da48ae5ed0c053c67a4/view.htm>

[14] Codec Specs. *Matroska* [online]. 2005 [cit. 2014-11-30]. Available from:
<http://www.matroska.org/technical/specs/codecid/index.html>

[15] HARALICK. A Storage Efficient Way to Implement the Discrete Cosine
Transform. *IEEE Transactions on Computers.* 1976, C-25, issue 7, pp.
764-765. DOI: 10.1109/tc.1976.1674687.

[16] VETTERLI, M., M. EURASIP and H. J. NUSSBAUMER. Simple FFT and
DCT algorithms with reduced number of operations. *Signal Processing.* 1984,
vol. 6, issue 4, pp. 267-278. DOI: 10.1016/0165-1684(84)90059-8.

[17] W. CHEN, C. SMITH and S. FRALICK. A Fast Computational Algorithm
for the Discrete Cosine Transform. In: *IEEE Transactions on
Communications.* 1977, pp. 1004-1009. ISSN 0096-2244. DOI:
10.1109/TCOM.1977.1093941. Available from: <http://ieeexplore.ieee.
org/lpdocs/epic03/wrapper.htm?arnumber=1093941>

[18] SUEHIRO, N., M. HATORI. Fast algorithms for the DFT and other sinusoidal
transforms. *IEEE Transactions on Acoustics, Speech, and Signal Processing.*
1986, vol. 34, issue 3, pp. 642-644. DOI: 10.1109/tassp.1986.1164854.

[19] YIP, P., K. RAO. DIF Algorithms for DCT and DST. ICASSP '85. *IEEE
International Conference on Acoustics, Speech, and Signal Processing.* 1985,
pp. 88-121. DOI: 10.1109/icassp.1985.1168246.

[20] HEIN, D. and N. AHMED. On a real-time Walsh-Hadamard cosine transform image processor. *IEEE Trans. Electromag. Compat.* 1978, vol. EMC-20, pp. 453-457.

[21] YANG, P. and M. NARASIMHA. Prime factor decomposition of the discrete cosine transform and its hardware realization. ICASSP '85. *IEEE International Conference on Acoustics, Speech, and Signal Processing.* 1985, pp. 944-965. DOI: 10.1109/icassp.1985.1168330.

[22] HOU, H. A fast recursive algorithm for computing the discrete cosine transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 1987, vol. 35, issue 10, pp. 1455-1461. DOI: 10.1109/tassp.1987.1165060.

[23] LICHTENBERG, A. and J. H. O'NEILL. A single chip solution for an 8 by 8 two dimensional DCT. *Intl. Symp. on Circuits and System.* 1987, ISCAS 87, pp. 1128-1131.

[24] TRAN, T.D. The binDCT: fast multiplierless approximation of the DCT. *IEEE Signal Processing Letters.* 2000, vol. 7, issue 6, pp. 141-144. DOI: 10.1109/97.844633. Available from: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=844633>

[25] LIANG, J. and T.D. TRAN. Fast multiplierless approximations of the DCT with the lifting scheme. *IEEE Transactions on Signal Processing.* 2001, vol. 49, issue 12, pp. 3032-3044. DOI: 10.1109/78.969511. Available from: <http://thanglong.ece.jhu.edu/Tran/Pub/binDCT.pdf>

[26] DANG, P., P. NGUYEN and T. TRAN. 2005. BinDCT and Its Efficient VLSI Architectures for Real-Time Embedded Applications. *JOURNAL OF IMAGING SCIENCE AND TECHNOLOGY.* (2). Available from: <http://thanglong.ece.jhu.edu/Tran/Pub/binDCT-VLSI.pdf>

[27] NASRABADI, N. and R. KING. Computationally efficient discrete cosine transform algorithm. *Electronics Letters.* 1983, vol. 19, issue 1. DOI: 10.1049/el:19830017.

[28] LAMBRECHT, J. Vision models and applications to image and video processing. *Boston: Kluwer Academic*, 2001, x, 229 p. ISBN 07-923-7422-3.

[29] Rods and Cones. *Hyperphysics* [online]. 2014 [cit. 2015-04-23]. Available from: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/rodcone.html>

[30] JPEG File Interchange Format. *W3.org* [online]. 1992 [cit. 2015-04-23]. Available from: <http://www.w3.org/Graphics/JPEG/jfif3.pdf>

[31] Fixed Point vs Floating Point. *Imperial College London [online]*. 2005 [cit. 2015-05-27]. Available from: <http://www.ee.ic.ac.uk/pcheung/teaching/ee3_Study_Project/lecture%205(4).pdf>

[32] Fixed versus Floating Point. *The Scientist and Engineer's Guide to Digital Signal Processing* [online]. 2011 [cit. 2015-04-30]. Available from: <http://www.dspguide.com/ch28/4.htm>

[33] BHASKER, J. *A systemC primer: [system design, IP, RTL, embedded software, verification]*. Allentown, Pa: Star Galaxy Publ, 2002. ISBN 09-650-3918-8.

[34] SystemC. *Accellera Systems Initiative* [online]. 2014 [cit. 2015-05-03]. Available from: <http://accellera.org/downloads/standards/systemc>

[35] *IEEE standard for floating-point arithmetic.* New York, NY: Institute of Electrical and Electronics Engineers, 2008. ISBN 978-073-8157-528. Available from: <http://www.math.fsu.edu/~gallivan/courses/FCM1/IEEE-fpstandard-2008.pdf.gz>

[36] An introduction to different rounding algorithms. *EE Times* [online]. 2006 [cit. 2015-05-16]. Available from: <http://www.eetimes.com/document.asp?doc_id=1274485>

[37] INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES. 1993. In: *CCITT T.81* [online]. [cit. 2015-05-05]. Available from: <http://www.w3.org/Graphics/JPEG/itu-t81.pdf>

[38] HDL analysis (HAL) User Guide. *Cadence* [online]. 2004 [cit. 2015-05-17]. Available from: <http://support.cadence.com/wps/mypoc/cos?uri=deeplinkmin:DocumentViewer;src=pubs;q=/hal/hal9.2/titlecopy.html>

[39] 65nm Technology. *TSMC* [online]. 2010 [cit. 2015-05-26]. Available from: <http://www.tsmc.com/english/dedicatedFoundry/technology/65nm.htm>

[40] SMITH, M. *Application-specific integrated circuits.* Reading, Mass: Addison-Wesley, 2008. ISBN 03-216-0275-7.

[41] VEENDRICK, Harry. *Deep-submicron CMOS-ICs: from basics to ASICs.* 2. Engl. ed. Dordrecht [u.a.]: Kluwer Acad. Publ, 2000. ISBN 90-440-0111-6.

[42] CILETTI, Michael D. *Advanced digital design with the Verilog HDL.* Upper Saddle River: Prentice Hall, 2003, xxi, 982 p. ISBN 01-308-9161-4.

[43] BHASKER, Jayaram. *A Verilog HDL primer.* 2nd ed. Allentown, PA: Star Galaxy Pub., 1999, xix, 294 p. ISBN 09-650-3917-X.

[44] BERGERON, Janick. *Writing testbenches: functional verification of HDL models.* Boston: Kluwer Academic, 2000, xxii, 354 p. ISBN 07-923-7766-4.

[45] *The Verilog Golden reference guide.* Version 2.0. Ringwood, GB: Doulos, 2003. ISBN 09-537-2804-8.

[46] *Low power design methodologies.* Editor Jan M Rabaey, Massoud Pedram. Boston: Kluwer Academic Publishers, 1996, x, 367 p. ISBN 07-923-9630-8.

[47] WILLIAMS, J. *Digital VLSI design with verilog: a textbook from Silicon Valley Technical Institute.* Dordrecht: Springer, 2008, xxiii, 435 p. ISBN 978-1-4020-8445-4.

[48] JAIN, K. *Fundamentals of digital image processing.* Englewood Cliffs, NJ: Prentice Hall, 1989, xxi, 569 p. ISBN 01-333-6165-9.

[49] *SystemVerilog Golden Reference Guide.* 2004. Ringwood, UK: Doulos Ltd. ISBN 0-9547345-0-5.

[50] *Asic World* [online]. 2014 [cit. 2015-05-23]. Available from: <http://www.asic-world.com>

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

DCT   Discrete Cosine Transform

DFT   Discrete Fourier Transform

MDCT   Modified Discrete Cosine Transform

VLSI   Very-Large-Scale integration

HDL   Hardware Description Language

RTL   Register Transfer Level

CMOS   Complementary-Metal-Oxide-Semiconductor

ASIC   Application-Specific Integrated Circuit

JPEG   Joint Photographic Experts Group

MPEG   Moving Picture Experts Group

MP3   MPEG-1 or MPEG-2 Audio Layer III

KLT   Karhunen-Loeve transform

AVC   Advanced Video Codec

CPU   Central Processing Unit

DSP   Digital Signal Processing

FPGA   Field Programable Logic Array

DIT   Decimation-in-Time

DIF   Decimation-in-Frequency

DHT   Discrete Hardley Transform

WHT   Walsh-Hadamard Transform

PFA   Prime Factor Algorithm

binDCT   Binary Discrete Cosine Transform

fps   Frames per second

PSNR   Peak Signal-to-Noise Ration

MSE   Mean-Squared Error

FSM   Finite State Machine

HAL   HDL Analysis

DUT   Design Under Test

HTML   Hyper-Text Markup Language

ICCR   Incinsive Code Coverage Report

TSMC   Taiwan Semiconductor Manufacturing Company

IP   Intellectual Property

LIST OF USED SOFTWARE

- MATLAB (R2014a)
- T_EXnic Center 2.02
- Microsoft Visio 2010
- Adobe Illustrator CS5.1
- Microsoft Visual Studio 2010
- Paint.NET
- Red Hat Enterprise Linux Workstation 6.5 (Santiago)
- Cadence Incinsive 13.10.003
- Synopsys 2014.09-SP2
- NEdit 2.3.1

LIST OF APPENDICES

A   CONTENTS OF THE ATTACHED CD

Tab. A.1: List of folders and their content description

Folder name	Description
c	Model in C and reference images. Model can be run by `c_model_golded_vector` file. It generates output images and both input and output vectors for the RTL simulation.
CONSTRAINS	Contains constrains files for synthesis.
nccoex	Simulation of RTL and HEL debug. Can be run by `run_simulation` and `run_hal` scripts.
RTL	Verilog files with RTL description of design.
synopsys	Synthesis folder. Contains synthesis results and reports. Synthesis can be run by `run_synopsys` script.
TESTBENCH	Contains testbench and textcases files as well as input and output test vectors generated by model
verdi	Folder for verdi debuging tool.