POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea Magistrale

VLSI architectures for the Steerable Discrete Cosine Transform

Relatore:
Prof. Maurizio Martina

Candidato:
Luigi Sole

Luglio 2018

Summary

In this thesis, a new directional transform has been exploited. It has the advantage of being applied to any two-dimensional separable transform, therefore, this method is applicable to a variety of signal processing fields. In particular, the aim of this thesis was to include this new directional transform in the HEVC (High Efficiency Video Coding) standard. Therefore, a directional integer DCT has been defined, called Steerable Discrete Cosine Transform (SDCT). It is obtained starting from an integer 2-D DCT unit, by steering its basis vectors in any direction. SDCT algorithm uses transform basis vectors with an orientation which is different from the vertical or horizontal one, achieving a more compact signal representation without losing important information. The architectural implementation is composed of two distinct parts: an integer two-dimensional DCT and a steerable architecture.
The integer 2-D DCT supports blocks of size 4x4, 8x8, 16x16 and 32x32. The implemented architecture uses a multiplier-free multiple constant multiplication (MCM) approach, which reduces its complexity, and allows to better exploit parallel processing architectures. In this way the throughput can reach a very high rate, while on the contrary both area occupation and power consumption are minimized.
The steerable architecture, on the other hand, has been implemented starting from an available C++ source code, and by making some choices on the internal parallelism in order to satisfy the same timing requirements of the DCT unit. While the 2D-DCT has a separable property that significantly reduces its complexity, the steerable unit is not separable, so it is a drawback for the complexity of hardware implementation. Despite the higher complexity, attempts have been made in order not to change the throughput, and limit as much as possible area occupation and power consumption. It has been decided to use two different clocks for DCT and steerable units, in order to reduce the internal parallelism of the second unit, but without changing the throughput. In particular, it has been chosen a clock for the steerable unit, which is four times faster than the one of the 2-D DCT unit.
Finally, the design has been synthesized using the UMC 65nm standard-cell library, at a frequency of 188 MHz, reaching a throughput of 3.00 Gsps. In particular, this architecture supports 8K Ultra High Definition (UHD) (7680x4320) with a frame rate of 60 Hz, which is one of the best resolutions of HEVC.

Table of contents

Chapter 1

Introduction to Image and Video Compression

In recent years, image and video applications have grown significantly, especially in applications such as medical and satellite imaging, TV broadcasting, internet and video conferencing. The image resolution of these applications is increasing, just think of the 1080p Full HD television broadcasting that has not yet completely spread in some areas, although some TV broadcasting with a 4K Ultra HD resolution are made available, and others with 8K Ultra HD will soon be available. This is leading to an increase of the memory occupation to store these higher resolutions files, and of their transmission bandwidth.Therefore, there is a need of data compression techniques that are more efficient.

The image compression techniques, such as JPEG, are able to remove the redundancy within a single image, exploiting the spatial correlation. On the other hand, the video compression techniques, in addition to the spatial correlation, exploit the temporal correlation. In practice, a video may be seen as a sequence of correlated images, therefore, this temporal correlation could be exploited to remove redundancy between them. The temporal redundancy is removed by using the motion compensation, which is an algorithm able to detect a moving object in different frames. Both image and video compression may be lossless or lossy. In the lossless approach, the number of bits are reduced, but without waste any information. On the contrary, the lossy approach reduces the number of bits by reducing or removing the less relevant informations. Due to the greater efficiency, the lossy compression technique is more common, although sometimes is mandatory the use of the lossless one.

This thesis work is structured as follows. In chapter 2 is proposed an overview of the most recent video compression standard, HEVC. In chapter 3 is presented a directional transform called steerable DCT (SDCT).

Chapter 4 shows the architectural implementation of the proposed SDCT transform. It is worth noting that the design specification have been chosen in way that this transform can be used inside HEVC standard. Finally, in chapter 5 have been reported area occupation and power consumption of the architecture, synthesized using a UMC 65nm standard-cell library.

1.1 Codec

The key instrument of image and video processing is the codec. It consists mainly of an encoder and a decoder unit. The first takes in input a video sequence and compress it into a digital bit-stream that is transmitted to the decoder units. The latter takes in input this digital bit-stream and reconstructs the video sequence, that is reproduced on the device display.

Compression codecs can be lossy or lossless. Lossless codecs do not introduce any type of artefacts, so that the decoded video precisely matches the encoded one. Obviously, this leads to a bit-stream too high, which often is not feasible. Therefore, has been introduced a lossy codec system, which reduce the bit-stream as much as possible, without an obvious change in video representation. The desirable degree of compression in the case of lossy compression is sometimes selectable by the user, especially in image compression case.

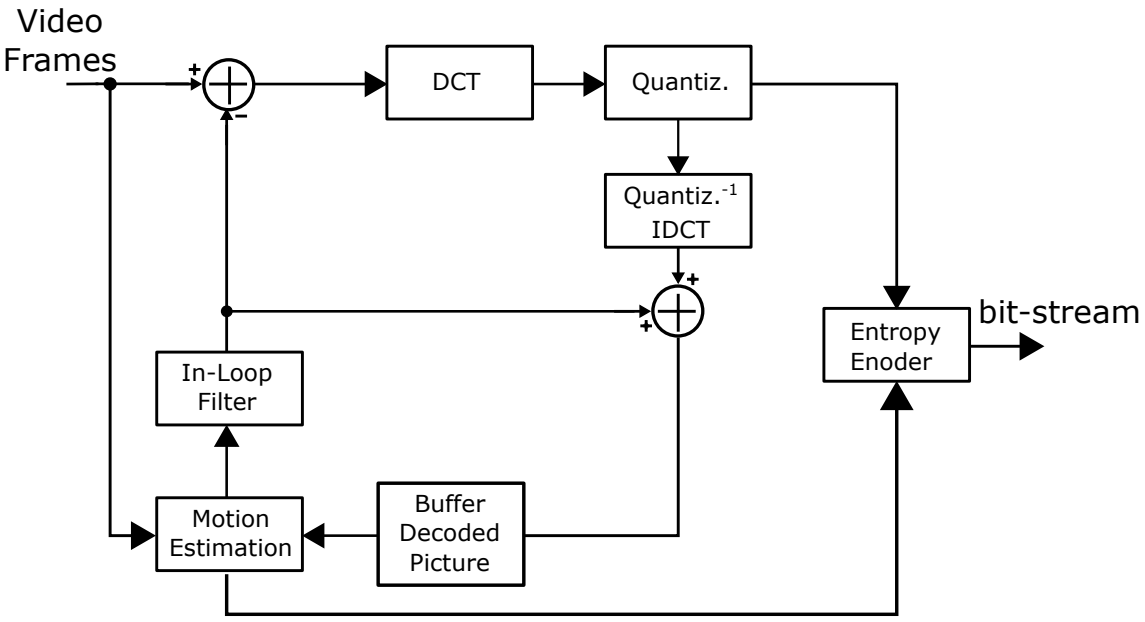


Figure 1.1. H.261 encoder diagram

Figure 1.1 shows the block diagram of H.261 encoder system, which is the earliest video compression standard, and so the easier one. A short explanation of how the codec works has been reported below. A video is nothing more than a sequence of pictures, and the encoder takes one of these pictures at a time and compresses it. First of all, the already encoded pictures are used to predict the current one, then the predicted picture is compared with the current picture and is made a difference between them. This difference, called residual, is sent to a spatial transform block (2D-DCT), then is quantized and the resulting block is sent to a coder unit. This last unit produces in output the bit-stream, that is transmitted together with additional information useful for decoding. The encoder unit includes also a decoding unit, which starting from the quantized blocks, applies an inverse quantization and transformation, so predicted samples are added, obtaining a reconstruction of the original picture. The latter is stored inside a buffer, and used to make the estimations of the following pictures. Finally, there is a block that taking as input the previous reconstructed picture and the input picture, make a motion estimation producing a motion vector, that is used to obtain the residual of the next computation. Moreover, the motion vector is sent to the coder block, that transmit it to the decoder together with other information.

The whole coded bit-stream, once sent to the decoder system, is decoded and added to the previous picture stored into the buffer, obtaining in output the current picture. So far has been briefly described the earliest video compression H.261 standard. In the following chapter, will be reported a description of the latest video standard (HEVC), focusing more attention on the functionality of each block.

1.2 Data Representation

In image and video compression, is preferred the use of a color representation different from the RGB one. Y Cb Cr is the color encoding scheme typically used for compression, because it take account of the human colour perception. Typically, it allows to better hide compression artefacts and errors to the human vision than RGB representation. This scheme use one luminance component and two chrominance components, in order to represent red, blue and green colours.

The Y component represents the brightness of the image, and it is the same signal that was used in black-and-white television. The two chrominance components Cr and Cb provide the degree to which the color differs from gray to red and blue, respectively. Since the human vision has a greater sensitivity to luminance than chrominance, the chrominance bandwidth is reduced, achieving a better compression performance. The chrominance subsampling typically used, in most video encoding and JPEG standards, is the 4:2:0 format. In this format, both blue and red chroma components have halved in both horizontal and vertical dimensions, so that there

are just one Cb and one Cr sample for each four Y samples. On the basis of this format, the luminance components are rectangular matrix of size W × H, while the chrominance are scaled to the size W/2 × H/2. Typically, each sample of each Y, Cb and Cr component has a length of 8-bit, but in some applications these may be represented with 10-bit precision.

The typical data representation is: YCbCr components with 4:2:0 sampling format and 8-bit length. In the remaining part of this work, will be considered this data representation for the encoder input and decoder output samples.

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

$$Cb = B - Y$$

$$Cr = R - Y$$

A video is nothing more than a sequence of images, that follow a certain display order. It is worth noting that, in general, this order is different from the encoding one. The order in which frames are organized is specified in group of picture (GOP) structure. The GOP can be composed by three different picture types: intra coded frame (I frame), predictive coded frame (P frame), bi-predictive coded frame (B frame). I frame does not refer to previous frames, so it is coded regardless of the other frames. P frame are coded using the interpicture prediction technique, referring to the previous coded frames, while B frame are coded by referring to previous and following frames.
GOP structure depends on the encoder configuration, then a random access configuration use a GOP different from the low delay one. In addition, GOP decides the order in which frames are encoded, and this order must be transmitted to the decoder, in order to properly reconstruct the video.

An example is provided below, to explain the difference between display and encoding order. Five frames are given, the first one of I-type, the last one of P-type, while the others are of B-type. The coding starts with the first I-type frame, which is coded without any reference to other frames. Subsequently the fifth frame is coded, referring to the first one already coded. The 3 remaining frames are of B-type, so they are coded by referring to previous and following frames. In particular, first is coded the third frame by referring to frames 1 and 5, then the second and fourth referring to adjacent frames.

1.3 Introduction to SDCT

In both image and video compression standards, the two-dimensional discrete cosine transform (2D-DCT) is absolutely the larger used transform technique. It is used in most compression standards, such as H.264, HEVC and JPEG. In particular, the HEVC standards, which is the most recent video compression technique, uses an integer approximated version of the DCT.

The DCT works efficiently in the case of smooth signals, while on the contrary it is inefficient when images contain shaped discontinuities. Indeed, in case in which discontinuities are present, the high-frequency coefficients may have a large value. Therefore, since the quantization step discard these coefficients, this will lead to poor compression performance.

Figure 1.2 shows the difference between an uncorrelated and a correlated image, making a comparison of theirs DCTs. In the case of uncorrelated image, it can be seen that the DCT shows many coefficients in the high frequency region (bottom right), and consequently a poor energy compaction. On the contrary, the correlated image, shows a good energy compaction, since all the coefficients are widespread in the low-frequency region.

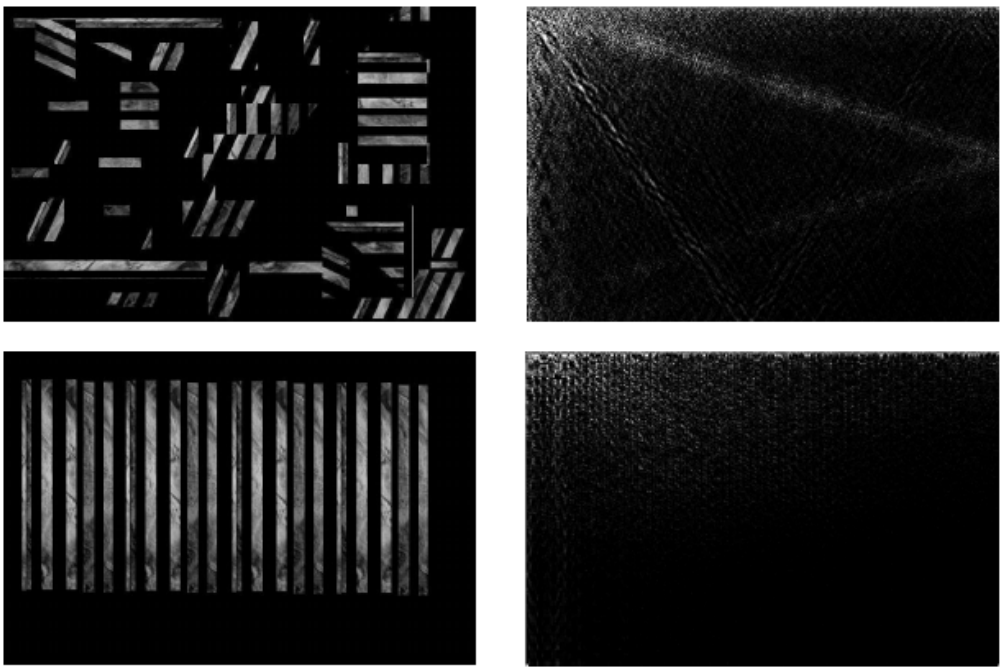


Figure 1.2. Uncorrelated and correlated images with their respective DCTs (Photo: [1])

In this work, a new transform approach has been exploited, in order to achieve better performance than DCT. In practice, a directional transform has been proposed, which is obtained by rotating the 2D-DCT vectors in every chosen direction. This directional transform is also called Steerable Discrete Cosine Transform (SDCT). SDCT algorithm is aware of image discontinuities, so it is able to minimise the amount of high-frequency coefficients, and consequently maximize the energy compaction without lost important informations. This algorithm can use a number of angles less or equal to the number of pairs of vectors rotated, but in this work it is considered the case with just one angle per block. The main disadvantage of this directional algorithm is the required cost to encode the DCT coefficients, that could exceed the coding gain. In particular, this cost increase in the case of blocks with a larger length (i.e. 16x16 and 32x32).

As it will be seen later, the above mentioned compression standards, use DCT of different sizes. More specifically, JPEG image compression uses just 2D-DCT of size 8x8, while H.264 video compression includes blocks of size 4x4, 8x8, and 16x16. Instead, the most recent HEVC standard, uses an integer version of DCT which supports blocks of size 4x4, 8x8, 16x16 and 32x32.

Since the aim of this work is to include SDCT in the HEVC standard, it has been realized an integer approximated architecture that supports blocks of size $n \times n$, with n = 4,8,16,32. Moreover, two scaled architecture, respectively with n = 4,8,16 and n = 4,8, have been proposed. This architectures can be implemented in the JPEG standard, or in the previous video compression standard H.264.
To realize this integer SDCT, first the integer 2D-DCT and subsequently the steerable unit have been implemented.

As will be reported in flowing chapter, this algorithm works efficiently when is used the all-intra configuration. Indeed, since all-intra configuration uses only intrapicture prediction, the residual still present an high spatial redundancy after the prediction operation, therefore substituting the DCT unit whit the SDCT one, it is possible to reduce this spatial redundancy and consequently improve the coding efficiency.

1.4 History of Video Compression

H.261 video compression standard is the first standard, and all subsequent video standards have been based on it. It was published in 1988 by ITU-T VCEG group. It was created to support videoconferencing transmission along ISDN lines, with a transmission rate from 64 kbit/s up to 1920 kbit/s. The pictures were divided in macroblocks of size 16x16 for luma components, and 8x8 for chroma components. An 8x8 DCT transform algorithm was used in order to reduce the spatial redundancy within a picture. It also employed the first interpicture prediction technique, to

reduce temporal redundancy.

The subsequent MPEG-1 standard was published in 1993 by ISO/IEC MPEG group. It was almost similar to H.261, but new resolutions were introduced. Moreover, the random access approach was introduced, which allows to use compression standard not only to videoconferencing applications, but also to digital stored video. In addition to forward interprediction, it also employs bidirectional prediction technique, which differentiates the encoding order from the display one.

In MPEG-2 standard, published in 1995, different profiles were defined, each of these is optimized for a given context, such as broadcasting, videoconferencing and digital stored applications. In addition, 16x8 prediction block size was introduced, and were supported resolutions up to 1920x1080 pixel.

H.263 was published on the basis of both H.261 and MPEG-1/2 standards. In addition to inter-prediction technique, also the intrapicture prediction was introduced, in order to reduce spatial redundancy between neighbouring blocks. Moreover, some deblocking filters were introduced to remove the defects caused by the image partitioning.

H.264/MPEG-4 was published in 2003 by the cooperation between ITU-T and ISO/IEC groups. It supported the picture partitioning macroblocks of size from 16x16 down to 4x4, and the transform block of size 4x4 in addition to 8x8. This added transform block size, leaded to an improved coding efficiency. Indeed, with small block it was more probable to have blocks with reduced range coefficients, and consequently with smaller coefficients at higher frequency. Moreover, this standard introduced some improvements for intrapicture prediction technique.

The latest video compression standard is HEVC, which was published in 2013, on the basis of previous H.264 standard. In the following section it has been reported a detailed description of all its features.

Chapter 2

HEVC

The High Efficiency Video Coding is the most recent video coding standard, designed by IUT-T and ISO/IEC organizations [2].
In the last years, HD video application are growing significantly, especially in mobile applications, and its resolution is also increasing. Resolutions such as 4K and 8K Ultra HD have been reached. Therefore, a new standard has been designed in order to maximize the coding efficiency.

In comparison with H.264, that is HEVC predecessor, with HEVC is possible to double the compression ratio without change the video quality, or on the other hand the video quality can be improved without increasing the bit rate. The HEVC is designed to support all the features of H.264 standard, and it can also handle higher resolutions, up to 8K UHD. Moreover, it exploit the use of parallel architectures, in order to obtain a faster and more efficient compression.

The HEVC syntax is generic, except the block structure and the bit-stream pattern which are the only standardized parts. Therefore the developer have the freedom to adjust and optimise the implementations according to the type of application. Moreover, in order to assist the developers to a better understanding of the standard, a reference software of encoder and decoder processes has been released.

As already mentioned in section 1.2, for representing the input video signals, images are separated in Y, Cb and Cr components. The Y is the luminance component and represents the image brightness. The two Cr and Cb are the chrominance components and provide the degree to which the color differs from gray and blue, respectively. In order to apply a subsampling of both chrominance components, the format 4:2:0 is used.
On the basis of this format, the luminance components are rectangular matrix of size W $\times$ H, while the chrominance are scaled to the size W/2 $\times$ H/2. The input samples of both luma and chroma components are typically represented with a precision of 8 bits, but this standard also supports 10 bits samples.

2.1 Block Diagram

In the figure 2.1 the block diagram of an HEVC video encoder is shown.

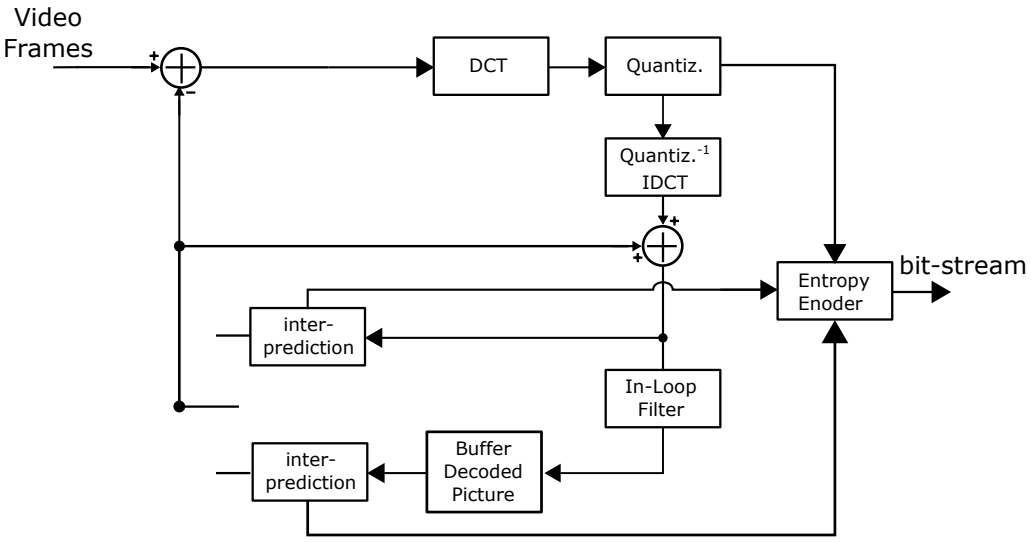


Figure 2.1. HEVC encoder diagram

As mentioned before, a video can be considered as a sequence of pictures, therefore one picture at a time is encoded. Before being encoded, each picture is divided in more blocks, each of which covers a shaped region of the image. At this point it is possible to proceed with the coding part, that can be carried out using an intrapicture or interpicture prediction. The intrapicture prediction is used for encoding the first picture of a sequence, and the pictures in presence of a random access point. This technique make a comparison between the neighbouring regions of the same picture, but it makes no comparison with the other pictures of a sequence. On the contrary, the interpicture prediction, compares the region of a picture with regions of already encoded pictures, using the motion compensation.

The results of the prediction block are subtracted to the original input picture, and its results called residual, are transformed, quantized and coded, before being transmitted. In addition, the data coming from both prediction blocks are also coded and then transmitted.

Within the block diagram of the encoder there is also a decoder, which generates the predicted picture which will then be used for the next prediction process. In practice, the residual is reconstructed by inverse quantization and inverse transform, then this is added to the prediction before going into a loop-filter which remove the quantization artefacts. The final result is stored in a buffer, and it is used for the

computation of the following prediction. The picture stored inside the buffer is a copy of the decoder output. It is worth nothing that the order of encoding process is often different from the order in which pictures arrive in input, it means that the display order is different from the encoding order.

2.2 HEVC Video Coding Features

The previous video coding standards divided the pictures in macroblocks composed of a block 16x16 of luminance components, and two blocks 8x8 of chrominance components. Nowadays, since the frames size is increasing, larger macroblocks are necessary in order to better encode the frames. In contrast, these frames are more detailed, therefore often is necessary to split these macroblocks to improve the encoding performances. Indeed, this partitioning process allows to exploit the use of parallel architecture, in order to obtain a faster and more efficient compression. In HEVC standard, these macroblocks, called coding tree unit (CTU), have a different size which is chosen directly by the encoder and may be wider. From each CTU are derived one luminance coding tree block (CTB) and two chrominance CTBs. Each luma CTB blocks has a size LxL, with L=16,32 or 64, while the chroma CTBs blocks have a size L/2xL/2, according to the 4:2:0 format.

The dimension of each CTB could be too large to decide whether to apply the inter prediction or intra prediction technique. Therefore, each CTU block can be further divided into more coding units (CU), until the minimum dimension 8x8 is reached. Just like CTU block, also CU is composed of a luminance coding block (CB) of size LxL and two chrominance CBs blocks of size L/2xL/2. The CU block is the decision block of prediction unit, but it could be still too big for the decision of the type of prediction, so it can be further divided into more prediction units (PUs).

According to the same principle, the CU blocks dimension could be too large to be transformed by the DCT unit. Indeed, in a single CB there may be both high frequency and low frequency informations. Therefore, each CU can be divided into TUs of smaller size, composed of one luma TB and two chroma TBs. TB square of size 4x4,8x8,16x16 and 32x32 feed the discrete cosine transform (DCT) unit.

Note that transform block does not need to have the same dimension of prediction block. All the decisions about the dimensions of the blocks which better encode the images are taken directly by the encoder unit, according with a function of the standard.

2.2.1 Slices and Tiles

Slices consist in a group of CTUs which are processed following the raster scanning order. Basically, a picture may be composed of one slice, or may be divided in more slices. The main feature of a slice is that it can be decoded without using the informations contained in other slices of the same image. In general there are three different kinds of slice, as reported below.

1. I slice: the intrapicture prediction mode is used to code all the CUs blocks of a slice

2. P slice: in addition to the intrapicture prediction mode, also the interpicture prediction is used to code some CUs of a slice, but with one motion-compensated signal at most.

3. B slice: in addition to the types of coding used for P slice, the interpicture prediction is used to code some CUs of a slice, but with two motion-compensated signals at most.

The number of CTUs included in a single slice depends on the video scene activity. The slice purpose is to enhance the video reconstruction despite the losses introduced by the coding process.

In order to exploit the parallel processing architectures for both coding and decoding processes, in HEVC have also been defined tiles blocks. In practice the picture is organized in rectangular regions, with typically the same number of CTUs per region, each of which region is a tile. Just like slice blocks, each tile can be decoded independently from the other. Moreover, it should be noted that a single tile may contain multiple slices and vice versa.

In HEVC are defined three different encoder configurations [3], that use one of the previously mentioned slice types, as reported below.

1. all intra (AI) configuration, in which I slices are used to encode all pictures.

2. random access (RA), which is used in broadcasting distribution and digital storage. It consists in the introduction of a random access picture at regular time intervals, e.g. every second.

3. low delay with B slices (LB), in which I slices are used only to encode the first frame. This configuration is used in application such as videoconferencing.

2.2.2 Intrapicture Prediction

This type of prediction is able to remove the spatial redundancy within a single frame of a video sequence. It is used to make prediction of the first frame of the video, and the frame which coincides with an added random access point. In practice, this type of prediction makes a comparison between the samples of the current PB and the samples of the surrounding PBs. In HEVC standard, it is supported the intrapicture prediction for only square PBs blocks of all sizes. To make this comparisons, thirty-five different prediction ways are available in HEVC standard:

- thirty-three directional prediction
- planar prediction
- DC prediction

If some of the surrounding PBs is not yet available, the HEVC automatically replace the unavailable PB with the neighbouring available one. Moreover, it should be performed a smoothing filtering in order to reduce the discontinuities among the neighbours blocks.

2.2.3 Interpicture Prediction

Successive pictures of a video sequence may contain the same moving or still object. The motion estimation technique try to recognize these common parts in different pictures, and produces the motion vector that contains the info about the shift of a determined object in two consecutive pictures. These information are used by the motion compensation algorithm in order to apply the data compression. Therefore, with these two techniques, video compression algorithm is able to reduce the temporal correlation between successive frames, and consequently reduces the quantity of data to send.

This type of prediction, called interpicture prediction, is performed on all the slice of B and P type. The first step of this prediction consists in splitting, if required, a single CB in two or four PBs, which may have also rectangular shape.In particular, if the square motion is chosen, the PBs will have a square shape. Instead, for the symmetric motion partition (SMP), PBs will have a rectangular shape of the same size. At last, the asymmetric motion partition (AMP) supports also rectangular shape of different sizes.

Once the picture is split, the prediction unit looks for blocks of equal size which contain samples similar to the blocks of the already encoded pictures.
Given the two blocks, the sum of the differences in the square between the pixels of the two blocks in the same position is carried out. If this value is grater than the threshold, it means that the blocks are different, therefore is not possible to apply

the motion compensation, and the block is sent to the transform unit as it is. On the contrary, motion compensation is applied, obtaining the motion vector. This motion vector indicates the difference between the same pixel location in the two blocks, identifying a movement in the picture. So, using this vector information, it is needed to shift the samples of the block belonging to the figure of reference, emulating the object motion, and obtaining as results the current encoded block.

It is worth noting that, interpicture prediction need a big portion of the encoding time. In particular, this time is needed by the motion estimation and compensation techniques, in order to produce a thorough prediction.

2.2.4 Spatial Transform

The most common type of spatial transform is carried out by the Discrete Cosine Transform unit(DCT). It has the aim to compact the energy of each sample, by including most of the images informations in few low-frequency coefficients, without effects on the image's quality

In HEVC, the DCT is carried out on the residual, that are obtained by subtracting the predicted block to the input one. As already mentioned, a single block may be divided in smaller blocks, therefore, the DCT unit should be able to support several block dimensions. In particular, in HEVC, the supported dimensions are 4x4, 8x8, 16x16 and 32x32. For all these dimensions, 2D-DCT is carried out by computing first the one-dimensional DCT in the horizontal direction (row), then in the vertical direction (column).

HEVC uses an integer approximated version of the DCT, which allows to reduce the required dynamic range and maximize the precision whether the input samples are integer. As a consequence of the increase of the supported dimensions, a right shift operation is needed after each 1D-DCT, in order to limit the width of each sample to 16 bits.

In the presence of blocks of size 4x4, it should be possible to apply an integer DST transform, instead of the DCT. It requires a slightly greater complexity, and provides a bit-rate reduction of 1%. It is worth using DST just for 4x4 luminance blocks, because for larger blocks the bit-rate reduction does not compensate the added complexity.

2.2.5 Quantization

This block is fed by the transformed samples, with the aim to discard the smaller values. This process improves the energy compaction, but, on the contrary, introduces some visible defects. In general, after applying the integer approximated DCT on the residual values, the TB coefficients are organized according to the frequency amplitude. The low frequency values have an higher energy than the high frequency

ones, therefore, low frequency values are more important to preserve the image's quality.

Basically, the quantization unit takes into account the energy components and discard with a different weight the various components. The HEVC algorithm, however, does not takes into account this property, and uses the Uniform Reconstruction Quantization (URQ) method. On the basis of this method, the quantization has the same effect on all the TB coefficients, regardless of their energy. The quantization level is set by the quantization parameter (QP), whose value is defined in a range from 0 to 51, and an increase of 6 of QP leads to a doubling of the quantization step. So, the higher this value, the lower the image's quality. The adaptive quantization method, with the use of scaling matrices, is also supported by HEVC.

2.2.6 In-Loop Filter

In HEVC, as it can be noticed from the figure 2.1, a in-loop processing step is applied to the decoded pictures before storing them inside the buffer. This in-loop process consists of two separated stages, which are a deblocking (DBF) filter and a SAO filter in cascade systems.

The DBF filter has the aim to reduce the artefacts resulting from the block-based coding. All the boundary samples of each TU and PU are filtered by the DFB, except when the boundary of the blocks coincides with the picture borders. Both TU and PU blocks are filtered, since in case of interpicture prediction these blocks might not be aligned.

Subsequently, SAO filtering is performed. It consists of adding to each sample of the picture, an offset value read from a look-up table which is updated by the encoder. Basically, SAO is a nonlinear system that further refines the reconstructed picture, improving both smooth part of the picture and the edge areas.

2.2.7 Entropy Encoding

The entropy encoding block, has the aim to reduce the number of bits needed to represent the information to be sent, so that coding efficiency and throughput can be further improved. Note that it is a lossless part of the data compression scheme. HEVC standards adopts the context-adaptive binary arithmetic coding (CABAC) algorithm, that compresses the quantized coefficients according to the probability estimation of some symbols.

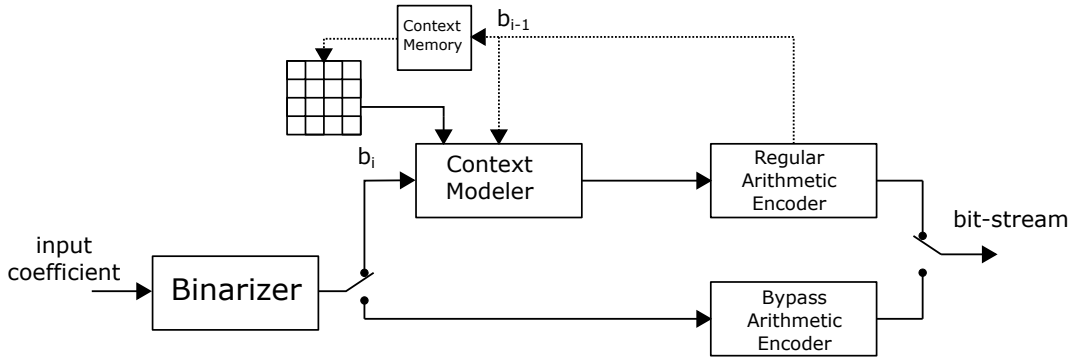


Figure 2.2. CABAC diagram

It consists mainly of three blocks, as shown in figure 2.2. The binarizer block, takes in input the quantized coefficients and convert them in binary symbols, and then bits are sent in a serial process to the context modeller block. The context modeler estimates the probability of symbols on the basis of the symbols recently codified. There is a context memory, that is a collection of models that depend from the symbols recently codified, and according to the chosen context memory, the modeller estimates the probability of any single bin to be '1' or '0'. At last, regular arithmetic encoder has the aim to encode the bin according to the probability estimated by the context modeler. Moreover, it updates the context memory on the basis of the actual estimated values, and this updated memory will be useful for the probability estimation of the next symbols.

The context modeler is the bottleneck of the system in term of speed, therefore is possible to select a bypass mode if it is required to speed-up the bit rates of the transmission. Selecting this mode, the bit string is sent to the coder block without estimate the probability, then the transmission speed may be increased, at the cost of a slightly smaller coding efficiency.

Typically, in order to encode the quantized coefficients, it is necessary to split TB blocks in 4x4 sub-blocks before scanning in a particular way. In particular, in HEVC three scanning methods are available, diagonal up-right, horizontal and vertical. One of these methods is selected according to the TB size and the type of prediction.

2.3 Improved Coding Efficiency

In comparison with H.264, which is the predecessor of HEVC, with HEVC is possible to double the compression ratio without change the video quality, or on the other hand the video quality can be improved without increasing the bit rate. However, changes in the various HEVC algorithm processes are still finding, in order to further improve the coding efficiency.

In this thesis project, a new transform technique which may increase the coding efficiency in the HEVC standard has been designed. This new transform technique is called Steerable DCT (SDCT). It is worth noting that in HEVC an integer approximation version of DCT is implemented, therefore the designed SDCT architecture is based on the same DCT algorithm.

In practice, adding a steerable unit at the output of the DCT unit, it is possible to increase the efficiency with respect the classic HEVC standard. In addition, a separable discrete cosine transforms presents some inaccuracies whem images contain sharped discontinuities [4]. These inaccuracies may be reduced by substituting the classic DCT with the Steerable SDCT.

In [5], it has been demonstrated that this SDCT transform works efficiently when the All-Intra configuration is used, while no improvements have been observed with Random Access and Low Delay configurations. Indeed, with random access and low delay configurations, both motion estimation and motion compensation are used in order to improve the encoder predictions. Therefore the residual are better compressed, and there is a greater probability that their value is zero. So, interpicture prediction leads to a better coding efficiency and the steerable transform do not leads further improvement. On the contrary, the all-intra configuration exploit only intrapicture prediction, therefore the residual spatial redundancy is still high and the steerable transform can reduce it.

Chapter 3

Steerable Discrete Cosine Transform

In this chapter a new directional transform algorithm has been presented, which consists of a two-dimensional discrete cosine transform that also can be steered in any direction.

First of all, an efficient version of the DCT algorithm has been presented, which can be implemented in HEVC video standard. The proposed DCT makes full use of parallel architectures and in addition is implemented without the use of multipliers. Finally, an overview of the Steerable DCT algorithm has been reported.

3.1 DCT Algorithm

The two-dimensional DCT of a NxN square shaped matrix is always separated into two one-dimensional N-point DCT, thanks to DCT separability property. The first 1D-DCT is computed along the vertical direction, while the second is computed along the horizontal direction.
In general, the one-dimensional N-point DCT may be expressed as follows:

$$X_k = \sqrt{\frac{2}{N}}\epsilon_k \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi(2n+1)k}{2N}\right] \quad k = 0,1,...,N-1$$
$$\epsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & otherwise \end{cases} \tag{3.1}$$

Applying the 3.1 along the vertical and horizontal direction, it is obtained in output a NxN matrix in which the top-left (0,0) element is called DC component and correspond to the average of samples sequence, while all other values are AC components. The vertical spatial frequency of AC components increases along the vertical direction, while the horizontal frequency increases along the horizontal direction. Therefore, the bottom-right (N,N) element has the highest horizontal and vertical spatial frequencies.

Starting from the output coefficients X_k it is possible to get back the starting samples, by multiplying X_k by the related cosine function. Assuming to have a DCT of length N, it can only be applied to block of the same length, then the input sequences are divided in blocks of the same size. It is worth noting that the basis function values are always the same, regardless of the input values, so these are pre-computed in order to reduce the complexity of the mathematical operations.

3.1.1 DCT Properties

The main benefit of DCT algorithm is the decorrelation properties, which consists in the elimination of redundancy between neighbouring pixels. In that way it is possible to encode independently the transformed output coefficients, and therefore the coding efficiency improves.

Moreover, implementing this transformation from spatial to frequency domain, the input samples information is compressed in a small number of output coefficients, while the other outputs have irrelevant values. So, after applying the DCT transform, a quantization scheme is used in order to discard the small coefficients, and the information is enclosed in the least possible number of coefficients. This is the energy compaction property, and the more the images are correlated, the more the energy is compacted.

Below an example of two-dimensional DCT and quantization step for a block of size 4x4 has been reported. In particular, the given example shows the HEVC behaviour, which is different from others video or image standards, such as JPEG. This is because the 2D-DCT in HEVC is an integer approximation version, and also because the quantization in HEVC is uniform, so the quantization has the same effect on all the transformed coefficients, regardless of their frequency. The quantization step (QP) is defined in a range from 0 to 51, and in this example a QP equal to 32 has been chosen, which corresponds to the default quantization value.

$$\begin{pmatrix} -2 & -1 & -3 & -1 \\ -2 & -1 & -2 & -1 \\ -2 & -2 & -2 & 0 \\ 2 & 3 & 4 & 6 \end{pmatrix} \xrightarrow{\text{DCT}} \begin{pmatrix} -32 & -228 & 160 & -99 \\ -74 & 66 & -21 & 6 \\ 32 & -9 & 0 & 21 \\ -56 & -26 & -9 & -3 \end{pmatrix} \xrightarrow{\text{QUANT}} \begin{pmatrix} -1 & -7 & 5 & -3 \\ -2 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

As it can be seen, after the quantization step, the majority of the informations is located in the low-frequency coefficients, while the coefficients at high frequency have become null. Therefore, the quantity of information to be encoded and transmitted is reduced. Once transmitted to the decoder, the processes of inverse quantization and inverse DCT reconstruct the starting pixels data. However, the reconstructed pixels are slightly different from the starting ones. This is because of the finite precision of integer DCT algorithm and the quantization error introduced.

3.1.2 Efficient Integer DCT

In this section a new way to implement the Discrete Cosine Transform (DCT) has been reported. It consists in an integer DCT version usable for different lengths equal to 4, 8, 16 and 32. This algorithm computes the DCT by using a scheme which exploit constant matrix multiplication, that has the advantage to be implemented using parallel processing architectures. The proposed algorithm has the advantage to reuse the matrix multiplication, in order to compute DCT of any length, starting from the same hardware implementation. This leads to a big advantage in terms of area occupation, as will be seen later in chapter 5. In practice, all transforms with a size greater than 4x4 can reuse operation to compute transform of a smaller size.

According to [6], the integer DCT for HEVC of length N can be calculated using an integer DCT of length N/2, and the product between a matrix of size (N/2)x(N/2) and a vector of size N/2, as reported in 3.2 and 3.3.

$$\begin{bmatrix} y(0) \\ y(2) \\ . \\ . \\ y(N-4) \\ y(N-2) \end{bmatrix} = C_{N/2} \begin{bmatrix} a(0) \\ a(1) \\ . \\ . \\ a(N/2-2) \\ a(N/2-1) \end{bmatrix} \tag{3.2}$$

and

$$\begin{bmatrix} y(1) \\ y(3) \\ . \\ . \\ y(N-3) \\ y(N-1) \end{bmatrix} = M_{N/2} \begin{bmatrix} b(0) \\ b(1) \\ . \\ . \\ b(N/2-2) \\ b(N/2-1) \end{bmatrix} \tag{3.3}$$

where

$$a(i) = x(i) + x(N-i-1)$$
$$b(i) = x(i) - x(N-i-1) \tag{3.4}$$

for $i = 0,1,...,N/2-1$.

In practice, starting from the input vector $X = [x(0),x(1),...,x(N-1)]$ of size N, first of all are derived the two vectors $a(i)$ and $b(i)$ of size (N/2), according to the equation 3.4.

Then, by multiplying the two vectors $a(i)$ and $b(i)$ by the constant matrix $C_{N/2}$ and $M_{N/2}$ respectively, the DCT output $Y = [y(0),y(1),...,y(N-1)]$ has been derived. In should be noted that the output vector of length N can be seen as two vectors of length N/2, representing the even and odd components.
$C_{N/2}$ is a matrix of size (N/2)x(N/2), $M_{N/2}$ has the same size and is derived from C_N, as reported in 3.5.

$$m_{N/2}^{i,j} = c_N^{2i+1,j} \quad for 0 \leq i,j \leq N/2-1 \tag{3.5}$$

where $c_N^{2i+1,j}$ correspond to the (2i+1,j)th entry of C_N matrix.

According to the same principle seen till now, the computation of the equation 3.2 can be further decomposed, starting from two new vectors $C_{N/4}$ and $M_{N/4}$.
What has been said so far is valid to compute an integer one-dimensional DCT. In order to obtain a two-dimensional DCT, is enough to apply the mentioned algorithm before along the vertical direction and after along the horizontal direction.
The implementation of the mentioned DCT algorithm requires $(N^2/4 + N/2 + ADD_{N/2})$ adders, $(N^2/4 + MUL_{N/2})$ multipliers and 2 shifts, where $ADD_{N/2}$ and $MUL_{N/2}$ corresponds to the number of adders and multipliers of a DCT of size N/2, respectively.

It is worth noting that the equation 3.3 can be realized by using a series of N/2 multiple constant multiplication (MCM) for each entry. This leads to an architecture more regular and with a lower complexity. In addition, the complexity can be further reduced by using a multiplier-less approach, in which the multipliers are replaced by add and shift operations.

The main component of integer one-dimensional DCT is the C_N matrix. Below, in 3.6 and 3.7 have been reported the matrices for DCT length equal to 4 and 8. It should be noted that C_4 can be derived from C_8, considering only the even rows $i = 0,2,4,6$ and the first four columns $j = 0,1,2,3$. Similarly, C_8 is derived from C_{16} and C_{16} is derived from C_{32}. The starting matrix for integer DCT of length 32 is reported in [7].

$$C_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{3.6}$$

$$C_8 = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix} \tag{3.7}$$

The hardware implementation for this integer 1D-DCT algorithm can be divided into three different stages. The first stage, called input adder unit(IAU), is responsible for calculating the addition and subtraction operations of input vector reported in equation 3.4. The second stage is the shift adder unit (SAU), which computes a series of N/2 constant multiplication, by using only shifts and adders. These intermediate SAU outputs are sent to an output adder unit (OAU), which computes the DCT output. The cascade of second and third stage computes the product matrix per vector reported in equation 3.3.

In general, considering a one-dimensional DCT of length 8, the odd output components [$y(1)$,$y(3)$,$y(5)$,$y(7)$] are calculated according to these three stages, while the even output components [$y(0)$,$y(2)$,$y(4)$,$y(6)$] are calculated using a 4-point DCT.

3.2 The Steerable Algorithm

In this section, a new transform that incorporate the directional information has been presented. This one is made on the basis of any two-dimensional transform, such as DST and DCT transforms, adding a new architecture to the existing 2-D transform. In particular, the aim of this work is to incorporate this directional property inside the integer DCT used in HEVC standard. Compared to the standard DCT algorithm, the directional transform allows to achieve a greater coding efficiency, in the case of images with shaped discontinuities.

Until now, all the directional transforms proposed incorporate these information about the direction inside the two-dimensional transform unit. In [8], has been proposed a DDCT, which consists of two separated 1D-DCT that can be performed along a direction different from the horizontal or vertical one. However, all the methods proposed so far have different drawbacks. First of all, these directional transforms require one-dimensional DCT of several lengths, which should not necessarily be power of two, and should be very short. Moreover, all these proposed methods are designed to be adopted inside a specific type of transform, such as DCT or DST, so these are not generalizable to every type of transform.

The new steerable transform method, called SDCT [4], has been reported below. This have the advantage of being adopted inside any two-dimensional transform, and can be implemented for any size from 32x32 down to 4x4, that correspond to the lengths used for the integer DCT in HEVC standard. The proposed transform can rotate the image blocks in any direction.

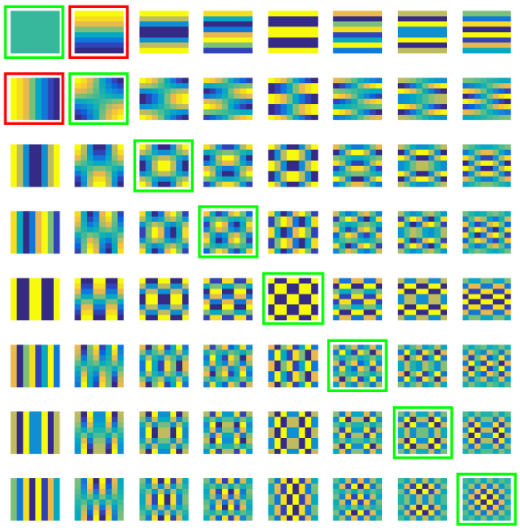


Figure 3.1. 2D-DCT basis vectors with n = 8 (Photo: [4])

In figure 3.1 the matrix form of a 2D-DCT basis of length n=8 is shown. Two eigenvectors have been highlighted in red, to highlight that the same frequency is represented in both eigenvectors, but one of them follows the horizontal direction while the other follows the vertical direction, as it can be observed. It should be noted that, except for the diagonal eigenvectors, that are not related to any other vector, all the others are related in pairs of vectors.

Given two vectors $\mathrm{v}^{(k,l)}$ and $\mathrm{v}^{(l,k)}$, corresponding to the same eigenvalue with multiplicity 2, these can be rotated according to the following system

$$\begin{bmatrix} \mathrm{v}^{(k,l)'} \\ \mathrm{v}^{(l,k)'} \end{bmatrix} = \begin{bmatrix} \cos\theta_{k,l} & \sin\theta_{k,l} \\ -\sin\theta_{k,l} & \cos\theta_{k,l} \end{bmatrix} \begin{bmatrix} \mathrm{v}^{(l,k)} \\ \mathrm{v}^{(k,l)} \end{bmatrix} \tag{3.8}$$

where the angle $\theta_{k,l}$ has the range in [0,2π].

So, applying the directional transform, all the pair of vectors $\mathrm{v}^{(k,l)}$ and $\mathrm{v}^{(l,k)}$, with $0 \leq l,\ k \leq n-1$ and $l \neq k$, are substituted by the pairs of rotated vectors $\mathrm{v}^{(k,l)'}$ and $\mathrm{v}^{(l,k)'}$, and the rotated matrix $V(\theta)$ is obtained. It is worth noting that in a matrix of size $n \times n$ are rotated only the off-diagonal elements, so the number of rotated pairs is $p = \dfrac{n(n-1)}{2}$.

Then theoretically, it can be possible to use the same number p of angles, rotating each pairs for a different angle. All the angles used are stored inside the vector θ, and are transmitted to the decoder together with directional transform output data. This directional transform can be represented as

$$V(\theta) = VR(\theta), \tag{3.9}$$

where θ is the vector that contain all the angles and $R(\theta)$ is the rotation matrix. The matrix $V = V(0)$ corresponds to the 2D-DCT output, since rotate for a zero angle is equivalent to not rotate.

It should be noted that $R(\theta)$ is composed of two different matrices as

$$R(\theta) = \Delta + \widetilde{R}(\theta) \tag{3.10}$$

where Δ is the matrix that represents the vectors that are not rotated, with ($\Delta_{ij} = 1$) for $i = j$ otherwise ($\Delta_{ij} = 0$), while $\widetilde{R}(\theta)$ is the matrix that represents the rotated vectors. Given $0 \leq k,l \leq n-1$ and $k \neq l$, if $i = kn + l$ and $j = ln + k$, then $\widetilde{R}(\theta)_{ii} = \widetilde{R}(\theta)_{jj} = cos(\theta_{k,l})$, $\widetilde{R}(\theta)_{ij} = sin(\theta_{k,l})$, and $\widetilde{R}(\theta)_{ji} = -sin(\theta_{k,l})$.

The components $\theta_{k,l}$ are ordered according to the zigzag scheme shown in figure 3.2, that is different from the classic zigzag ordering. Indeed, in this case, are considered just p elements, because the vectors are rotated in pairs so $\theta_{l,k} = \theta_{k,l}$ and in addition the diagonal $\theta_{k,k}$ are useless, since diagonal vectors are not rotated.

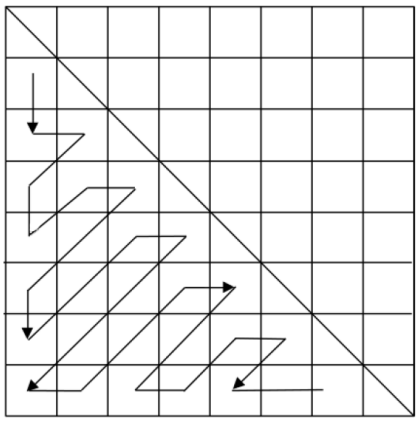


Figure 3.2. Zigzag ordering

The figure 3.3 shows the basis vectors resulting from the steerable transform with an angle $\theta = \frac{\pi}{4}$. As it can be noticed, the diagonal vectors are unchanged with respect the 2D-DCT ones, while all the others are rotated of an angle $\frac{\pi}{4}$.

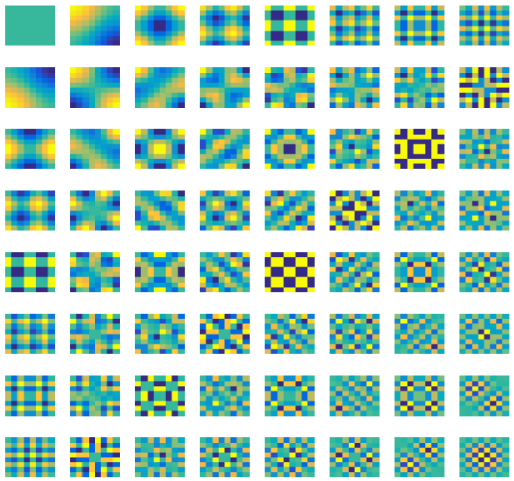


Figure 3.3. Directional DCT with $\theta = \frac{\pi}{4}$ (Photo: [4])

Considering the equation 3.9, it is worth noting that the directional transform $V(\theta)$ is a non-separable algorithm, although the transform V is separable. The complexity of transform that is not separable is far greater than a separable one, so this may be a big drawback for the implementation. However, the complexity can be decreased drastically by computing the separable transform before applying rotations.

In order to improve the compression as much as possible, the optimal angle must be found. Considering the couple of eigenvectors $c_{k,l}$ and $c_{l,k}$, according to equation 3.8, if these vectors are rotated by angle

$$\theta_{k,l} = \arctan\frac{c_{k,l}}{c_{l,k}} \tag{3.11}$$

one of the two rotated values becomes null. More specifically, $c'_{k,l} = 0$ and the energy of the couple is totally transmitted to $c'_{l,k}$. Assuming to choose one different angle for each couple of vectors, it will be possible to cancel p coefficients. This is a big advantage, since the number of coefficients decrease from n^2 to $n^2 - p$, without changing the image. So the steerable is classified like a lossless encoding process. However, all the angles selected must be transmitted to the decoder along with rotated coefficients, increasing the amount of transmitted data. For this reason, a reduced number of rotation angles is taken into consideration, although this will mean having a lower number of null coefficients. In particular, the case taken into consideration inside this work of thesis, is to use just one angle for a single block, since it has been demonstrated that using just one angle is already enough to outperform the standard 2D-DCT.

For what concerns the value that each angle can assume, q values are equally distributed in $[0, \frac{\pi}{2}]$. Obviously, higher is the number of angle q, more bits will be needed to represent it. It has been demonstrated in [5] that a number of values greater than 16 does not provide any coding improvement, since increases the bit-rate necessary to transmit the angles. Indeed the number of bits necessary to transmit each angle is equal to $log_2 q$. Therefore, a value $q = 8$ has been chosen, which is enough to obtain a good coding efficiency.

In summary, in addition to rotated coefficients, some overhead bits need to be sent. The first is a flag which is sent for each coding unit to indicate whether the rotations are applied. On the contrary, for each transform unit of a single CU are transmitted $log_2 q$ bits that indicate the angle value.

It should be noticed that the benefits of directional transform are more evident for video sequences at high resolution. Indeed, in this case the images show larger uniform regions, that leads to larger TU size. Consequently, the number of angles that has to be transmitted decreases. Considering for instance to have a single block of size 8x8 instead of four block of size 4x4. In the firts case just one angle must be transmitted, so the overhead is equal to $log_2 q$. On the contrary, in the second

case the overhead is equal to $4 \times log_2 q$. Then having a larger block the overhead is reduced by 4 times, and should be further reduced having even larger blocks of size 16x16 or 32x32.

Below two numeric examples that show how SDCT transform works have been reported. A directional transform is applied on a 4x4 transform block, using one angle for TU block and 8 angle values equally distributed in $[0, \frac{\pi}{2}]$.
Generally, the value of the angle is calculable as

$$angle = \frac{z \times \pi}{2 \times NUM\_ANGLES}$$

where z is the quantized angle that assumes a value from 0 to 7. The example in equation 3.12 corresponds to a standard 2D-DCT, since a rotation for the angle=0 is equivalent to not rotate. On the contrary, the example in 3.13 corresponds to a rotation of an angle z=7. As it can be seen, the diagonal data are the same of the previous example, while the other data are changed. This rotation produces a better signal representation, in particular it can be seen that the high-frequency data take smaller values.

- quantized angle z=0

$$\begin{pmatrix} -2 & -1 & -3 & -1 \\ -2 & -1 & -2 & -1 \\ -2 & -2 & -2 & 0 \\ 2 & 3 & 4 & 6 \end{pmatrix} \xrightarrow{\text{SDCT}} \begin{pmatrix} -32 & -228 & 160 & -99 \\ -74 & 66 & -21 & 6 \\ 32 & -9 & 0 & 21 \\ -56 & -26 & -9 & -3 \end{pmatrix} \tag{3.12}$$

- quantized angle z=7

$$\begin{pmatrix} -2 & -1 & -3 & -1 \\ -2 & -1 & -2 & -1 \\ -2 & -2 & -2 & 0 \\ 2 & 3 & 4 & 6 \end{pmatrix} \xrightarrow{\text{SDCT}} \begin{pmatrix} -32 & -239 & 163 & -108 \\ 29 & 66 & -23 & 0 \\ 1 & 6 & 0 & 19 \\ 37 & 28 & 14 & -3 \end{pmatrix} \tag{3.13}$$

Before being encoded and transmitted, these data are quantized. Considering a quantization step QP=32, it can be observed how the quantized data change depending on rotation angle value. Below have been reported two quantized output data, resulted from steerable transform with different orientation. As it can be observed, in the second case there are two additional null values. It means that, just choosing a different angle, it is possible to transmit 6 data instead of 8. Therefore, is reduced the number of bits required to transmit the encoded coefficients.

$$\begin{pmatrix} -1 & -3 & 1 & -2 \\ -6 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} -1 & -7 & 5 & -3 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{3.14}$$

3.3 Simulation Results

In this section, the coding efficiency improvements of the proposed Steerable DCT algorithm which are obtained in [5] will be shown.
In practice, the reference software HM-16.6 has been modified in order to include the integer Steerable DCT transform. The SDCT software includes mainly two parameters: the width of the rotation coefficients and the number of angles [q].
The rotation coefficients have been represented in 8 bits, whereas the number of angles, which is a power of two parameter, has been represented as a variable.

The modified software has been simulated for different common test conditions (CTC) of HEVC standard. The coding efficiency of this modified software, which includes the Steerable DCT transform, has been measured using the Bjøntegaard delta bit-rate (BD-Rate). This parameter indicates the bit-rate variation of the modified HM encoder, which includes the proposed Steerable transform, compared to the initial HM-16.6 HEVC encoder.

In table 3.1, the bit-rate variation for HEVC classes has been reported, according to the resolution and the number of angles. It should be noticed that a negative value of the BD-Rate indicates that the modified software is more efficient than the HEVC standard. As can be seen, there is an improvement in coding efficiency by increasing the number of angles, because in this way the encoder has more possibilities to find the optimum angle. Anyway, with more than 16 angles, the improvements reached with SDCT are overcome by the greater number of bits needed to represent the angles. As it can be observed by the results, the best solution is the adoption of 8 rotation angles.

Class	resolution	q=2	q=4	q=8	q=16
A1	8192x4096	-0.433	-0.572	-0.628	-0.645
A2	3840x1920	-0.514	-0.618	-0.634	-0.633
A	2560x1600	-0.378	-0.577	-0.682	-0.716
B	1920x1080	-0.305	-0.433	-0.472	-0.464
C	832x480	-0.277	-0.325	-0.322	-0.302
D	416x240	-0.288	-0.379	-0.364	-0.339
E	1280x720	-0.080	-0.106	-0.101	-0.089
F	1024x768	-0.176	-0.208	-0.205	-0.220

Table 3.1. Comparison of SDCT transform versus standard transform in HEVC standard, for all-intra configuration

At the same time, it should be noted that the benefits of this Steerable transform are much better for video classes with higher resolution. This is because the sequences with a higher resolution have large portion of each picture with a uniform content. Therefore, the picture can be partitioned through larger TU blocks, and consequently, fewer rotation angles must be sent to the decoder. It is worth noting that only the results related to all-intra configuration have been reported, since this is the configuration that better exploit the directional transform. In fact, the all-intra configuration uses only intrapicture prediction, so the residual still have a high spatial redundancy after the prediction operation. Therefore, by replacing the DCT unit whit the SDCT one, the spatial redundancy can be further reduced and consequently the coding efficiency is improved.

Chapter 4

Hardware Architecture

The purpose of this chapter is to derive the best architecture that implements the algorithm of the SDCT, which has been analysed in section 3.2. The main specification of this architecture was to be included in the HEVC standard, which standard has a very high throughput. In particular, this architecture supports $8K$ UltraHigh Definition (UHD) (7680x4320) with a frame rate of 60 Hz and 4:2:0 YUV format, which is one of the best resolution of HEVC. Moreover, a lot of optimizations have been performed, in order to minimize the complexity and the power consumption. In particular, this chapter has been divided in two parts. The first is about the DCT architecture, while the second is about the additional Steerable architecture, which makes this video compression standard more efficient.

4.1 2D-DCT Unit

As said in section 3.1, exploiting its separable property, the two dimensional integer DCT could be decomposed in two separate stages. In the first stage, the N-point one-dimensional integer DCT is calculated for any column of the matrix in input, generating an intermediate matrix of the same size. In the second stage, another N-point one-dimensional integer DCT is calculated for rows of intermediate results. Accordingly to this property, it is possible to implement a full-parallel architecture or a folded architecture.

The full-parallel structure is realized by two different blocks which compute the one-dimensional DCT, and a transposition buffer in the middle. During the first N cycles, the first stage receives N consecutive columns in input and computes the DCT-1D intermediate results, which are stored in the transposition buffer following the column-wise order. Subsequently, during the other N clock cycles, the second stage receives row-wise data from the transposition buffer and computes the 2D-DCT output. At the same time, the first stage could computes the intermediate output

of a new input matrix. Therefore, the throughput of the full-parallel architecture is really high, but this requires a large area occupation. In Figure 4.1 the full-parallel architecture has been reported.

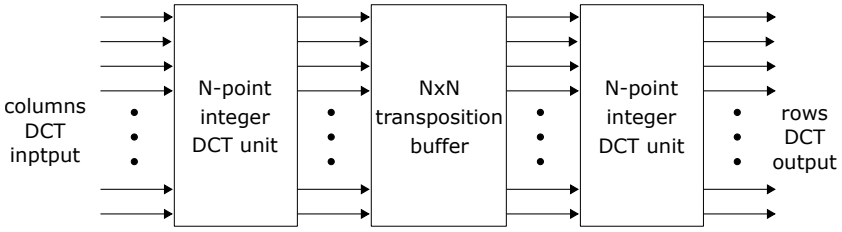


Figure 4.1. Full-parallel 2-D integer DCT architecture

To use less space, it is possible to implement a folded architecture, which is realized by a single 1D-DCT module and the transposition buffer. The functioning of this architecture differs from the full-parallel one. During the first N clock cycles, the MUXes at the input of the DCT module provide the N consecutive columns from the input and the results are stored in N consecutive columns of the transpose buffer. Subsequently, during the other N clock cycles, the MUXes at the input of the DCT module provide the rows output from transposition buffer, then the 2D-DCT output are computed.

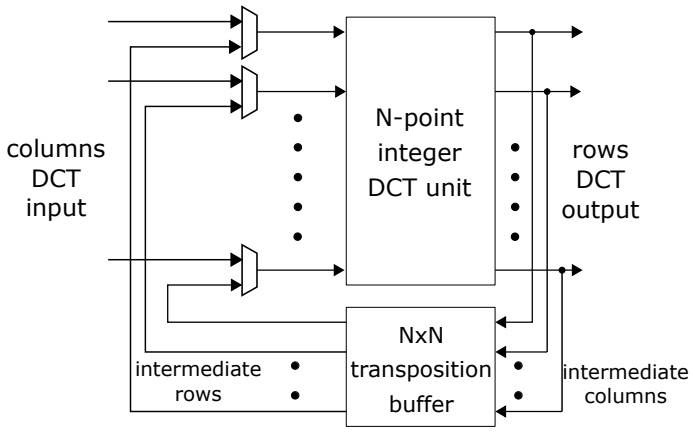


Figure 4.2. Folded 2-D integer DCT architecture

In Figure 4.2 the folded architecture has been reported. This type of structure has a throughput that is half of the full-parallel architecture, but the area is reduced significantly. In addition, the complexity of the transposition buffer is reduced as well. In facti, in the folded case, the transposition buffer requires a column-wise

writing and a row-wise reading. Instead, in the full-parallel architecture, are needed both row-wise and column-wise in writing and in reading, since the buffer must support writing and reading at the same time.

4.1.1 1D-DCT Architecture

In the following, the architecture proposed in [6] has been reported. This architecture is able to compute DCT of several dimensions, specifically size of 4, 8, 16 and 32, which are the DCT dimensions implemented in the HEVC. Moreover, it could be reused for any DCT lengths, providing always the same throughput of 32 DCT outputs per clock cycle, independently of the DCT lengths. The DCT-1D architecture for N=8, 16 and 32 is shown in Figure 4.3.

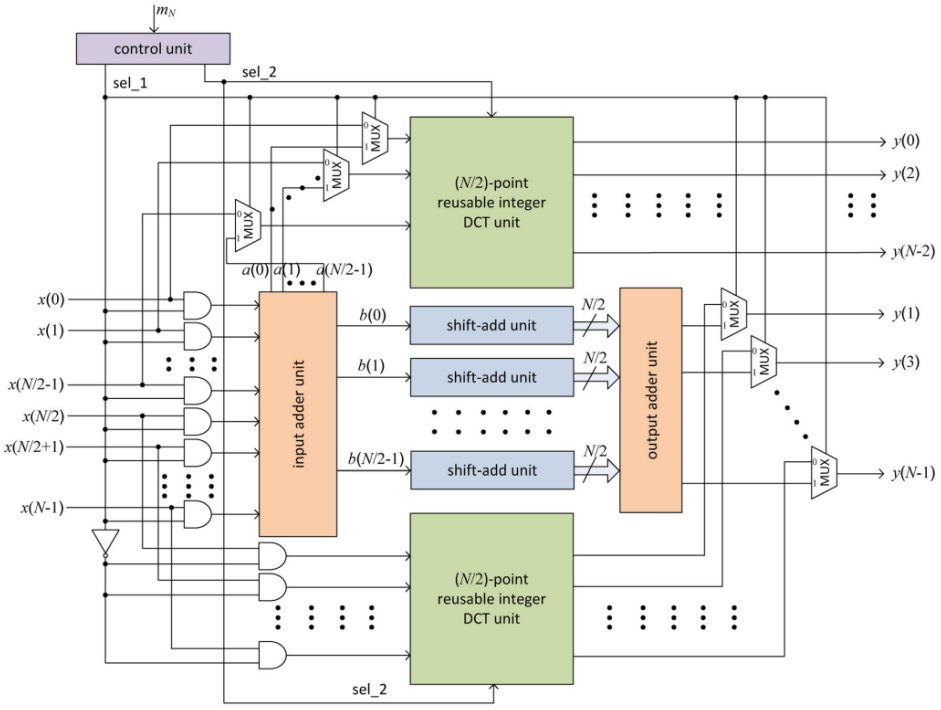


Figure 4.3. Reusable 1-D integer DCT architecture (Photo: [6])

As can be seen, there are two (N/2)-point reusable integer DCT units. The top (N/2)-point DCT has in input (N/2) MUXes, which provide to the unit the input data $[x(0),x(1),...,x(N/2-1)]$ or the output of the input adder unit (IAU) block $[a(0),a(1),...,a(N/2-1)]$, depending on the DCT dimension.

If DCT is used for N-point computation, the MUXes provide data from IAU, otherwise, for a lower DCT size, MUXes provide data from the input. The bottom (N/2)-point DCT takes the data $[x(N/2),x(N/2+1),...,x(N-1)]$, if the DCT unit is used for (N/2)-point or lower size computation, otherwise, the input AND ports disable the (N/2)-point DCT unit. The block composed of input adder unit (IAU), shift adder unit (SAU) and output adder unit (OAU), is used to compute N/2 outputs, in the case of DCT size equal to N. The input adder unit too, has in input AND ports which are driven by the complementary signal of the one that drives the bottom (N/2)-point DCT unit. That means that just one at a time between (N/2)-point DCT unit and IAU/SAU/OAU block is enabled, so the outputs of the two units are multiplexed. The shown structure has always the same throughput of 32 DCT outputs per clock cycle, so at a same time eight 4-point DCTs, or four 8-point DCTs, or two 16-point DCTs, or one 32-point DCT can be computed.

Assuming the case of a 32-point DCT, the entire architecture is composed by two 16-point DCT units, and the cascade of a 16-point IAU, a 16-point SAU and a 16-point OAU. In turn, each 16-point DCT unit, is composed by two 8-point DCT unit, and the cascade of a 8-point IAU, a 8-point SAU and a 8-point OAU. The same applies to 8-point DCT unit, and finally there is the 4-point integer DCT, which is the minimum DCT size. The architecture of the 4-point DCT is shown in Figure 4.4.

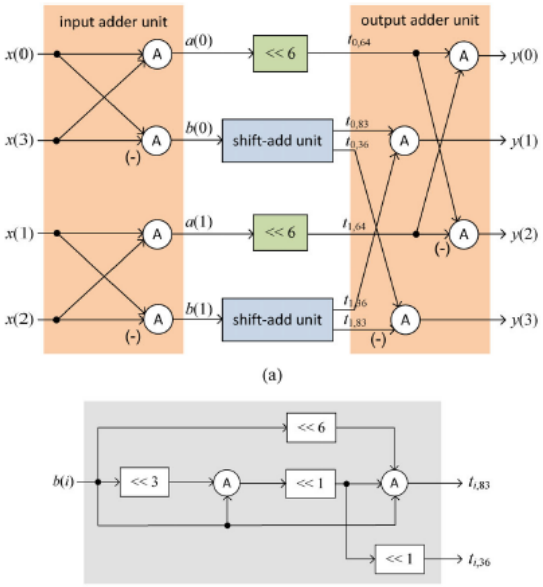


Figure 4.4. 4-point integer DCT (Photo: [6])

As already said, the computation of a DCT can be divided in three different stages. The input adder unit, which consists of four adders, is the first stage. The second stage consists of two shift-add units and two simple shift operations, which do not require any logic. Finally, the output adder unit takes the SAU outputs and compute the final results.

Figure 4.5 shows the schematic of IAU, SAU and OAU for a 8-point DCT unit, which is slightly more complex than the 4-point unit. In general, the number of adder in the IAU unit is equal to the number of point, so in that case is 8. The OAU is made of an adder tree with $log_2 N - 1$ levels, so in the case of 8-point DCT, there are two levels and $N + N/2 = 12$ adders. The SAU is surely the most complex structure, since it is able to compute multiplication of DCT coefficient with input sample. This multiplication are made without the use of multipliers, but only with adders and shifters. In the case of 8-point DCT, there are four SAU, each one is composed of 4 adders and 5 shifters.

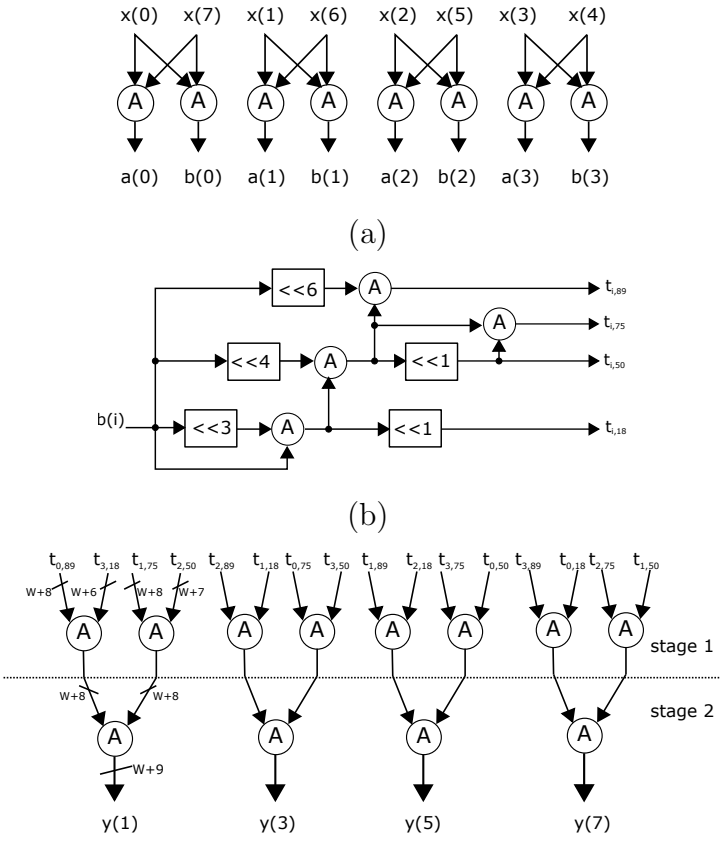


Figure 4.5. 8-point integer DCT (a) IAU strucuture (b) SAU structure (c) OAU structure

Similarly, the structures of 16-point and 32-point DCT can be obtained, which structures are more complicated. In general, the number of resources to compute one-dimensional N-point DCT has been calculated, and a comparison with reference algorithm resources has been shown in Table 4.1.

N	Reference Algorithm			Proposed Algorithm	
	MULT	ADD	SHIFT	ADD	SHIFT
4	4	8	2	14	10
8	20	28	2	50	30
16	84	100	2	186	86
32	340	372	2	682	278

Table 4.1. 1-D N-point DCT complexity

4.1.2 Pruned 1D-DCT Architecture

The input bit-depth of a two-dimensional DCT is equal to 9. This is because in HEVC the 2D-DCT unit has the task of transforming residuals, that are the difference between the current frame sample and the predicted one, both of which have an 8-bit width. As shown in Figure 4.6, data after each DCT step are limited to have 16 bits, irrespective of the number of point of the DCT. Therefore, in order to limit the length of intermediate results, a scaling process is needed after each step of the transform. In particular, the less significant log_2N-1 and log_2N+6 bits, are truncated after the first and second transform stages, respectively. More specifically, before truncating the less significant bits, a rounding technique is applied, in order to reduce the average rounding error.

The technique applied is the Round to Nearest scheme, since it is the scheme utilized in HEVC HM 16.6 software [9]. This scheme consists of adding a '1' bit in the position immediately successive to the bit you want to keep, before pruning. Therefore, the hardware required to apply this rounding scheme is just an adder after each DCT stage, and a MUX which choose the '1' bit position, that depends on the DCT number of point N.

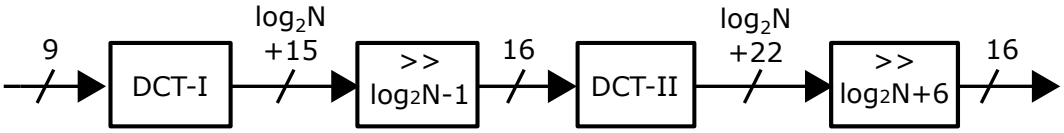


Figure 4.6. Transform chain with bit-width details

In order to reduce the complexity, it is possible to integrate the scaling operation within each 1D-DCT stage, instead of truncating at the end of each stage. As said before, in section 4.1.1, in each SAU there are some left shift operator, while the scaling operation is equivalent to a right shift operation. Therefore, as shown in Figure 4.7, it would be possible to change the SAU architecture, reducing the number of bits and consequently the adders complexity. Moreover, an additional scaling operation is applied on the SAU output, so even OAU complexity is reduced.

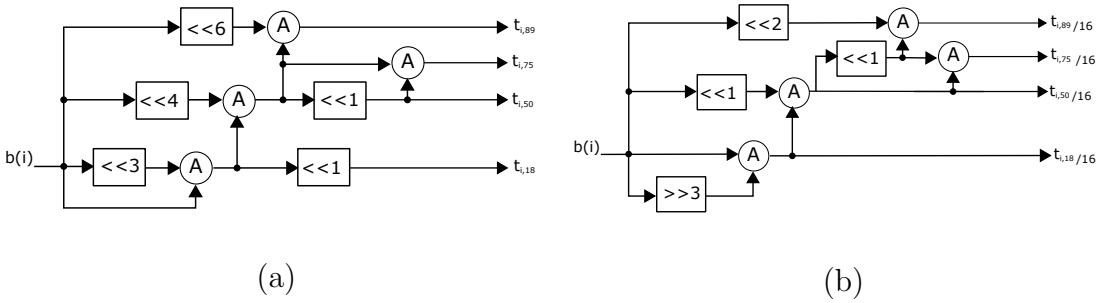


Figure 4.7. 8-point SAU (a) before and (b) after pruning

The following is a practical example to better understand the pruning scheme. Assuming the case of a 32-point DCT, the second stage (DCT-II) input data have a bit depth equal to 16, and the output should be truncated of $log_2 32 + 6 = 11$bits. In the case of the pruned architecture, instead of truncating these 11 bits at the output of the OAU, 4 bits have been truncated after the IAU computation, 4 bits after the SAU, and 3 additional bits at the output of the OAU. In this way, the shift-add unit has adders with a lowest bit width, 20 bits instead of 24 bits. The same goes for the output adder unit, which adders become 17-bit wide, while in the exact architecture adders are 27-bit wide.

In Table 4.2 are reported the information about internal word-length and pruned bits for both first and second stage, for all four DCT lengths. The forth and fifth columns indicate the number of additional bits required for the computation of the DCT output, while ninth column indicates the number of bit which need to be pruned in order to have always a 16-bit output width. The values of this ninth column correspond respectively to $log_2 N - 1$ for the first stage and $log_2 N + 6$ for the second stage. In the case of the pruned architecture, as said before, this pruning process has been integrated within each DCT stage. The sixth, seventh, and eighth columns, represent the number of bits pruned at the end of each indicated block. It is worth noting that, the sum of the values of these three columns it is just equal to the total bit pruned.

N	DCT stage	input width	IAU+SAU overflow	OAU overflow	IAU cut	SAU cut	OAU cut	total pruned	output width
4	DCT-1	9	8	0	1	0	0	1	16
	DCT-2	16	8	0	4	4	0	8	16
8	DCT-1	9	8	1	2	0	0	2	16
	DCT-2	16	8	1	4	4	1	9	16
16	DCT-1	9	8	2	3	0	0	3	16
	DCT-2	16	8	2	4	4	2	10	16
32	DCT-1	9	8	3	4	0	0	4	16
	DCT-2	16	8	3	4	4	3	11	16

Table 4.2. 2-D N-point DCT internal bit-width

4.1.3 2D-DCT Architecture

As said in Section 4.1, there are two ways to implement a two-dimensional DCT unit, a full-parallel architecture or a folded architecture. In this thesis work the folded approach have been adopted, since its throughput is enough to satisfy the project specifications.

The two-dimensional DCT is mainly composed of three blocks: a one-dimensional DCT, a transposition buffer, and a control unit (CU). Each 2D-DCT computation begins with a start signal, and the sel_dct that indicates the DCT length. The sel_dct signal is "00" for N=4 DCT length, "01" for N=8, "10" for N=16, and "11" for N=32. The input MUXes are drived by data_in_valid signal, that is asserted for the first N clock cycles.

Assuming, for instance a DCT length equal to 4, the 1D-DCT takes in input 32 data 9-bit wide each cycle, for four consecutive clock cycles. The intermediate results of this block are stored in the transposition buffer, that provides to transpose the internal results before being read. During next four clock cycles, the input MUXes switch and provide to 1D-DCT the 16-bit wide output data of the transposition memory, the data_out_valid signal become high and the 2D-DCT results are available.

4.1.4 Transposition Buffer

This module has the aim to store one-dimensional temporary results, before applying the second dimensional DCT stage. Moreover, these stored data are transposed before being read. Therefore, in this buffer data are written column-wise and read row-wise. The buffer has a size of 32x32 registers, each with a width of 16 bits. In order to facilitate the transposition, and reduce the hardware resources, there is a specific way to write the data. In practice, since the throughput is fixed and equal to 32, there are always 32 input data. Assuming a DCT length of 32, for 32 consecutive cycle, data are stored in columns. Assuming now a lower DCT length of 16, in that case data are stored for 16 consecutive cycle.

However, unlike the case with length equal to 32, in this case data should not be saved in the first 16 columns. Indeed, data[0 to 15] are saved in the first 16 columns in their respective rows [0 to 15], while data[16 to 31] are saved in the other 16 columns in their respective rows [16 to 31]. In practice, an offset equal to 16 is added to the column addressing of data[16 to 31], and the memory is divided in two blocks of size 16x16. The same goes for DCT length of 8, where there are four blocks of size 8x8, while for DCT length of 4, there are eight blocks of size 4x4. Since there is no need to write and read simultaneously, a single 5-bit address signal is used for both reading and writing addressing.

Figure 4.8 shows the hardware implementation of a 4x4 transposition buffer. It can be noted that each MUX takes in input the contents of the four rows of a single column, so that the reading is done row-wise, instead of column-wise.

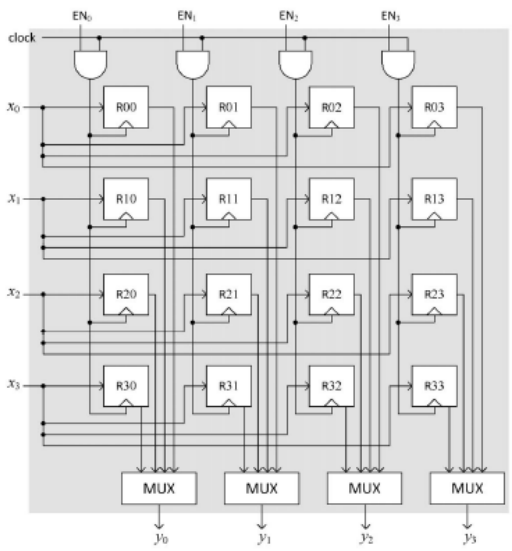


Figure 4.8. Transposition buffer schematic for N=4 (Photo: [6])

4.1.5 DCT Control Unit

The 2D-DCT block is managed by the control unit, that generates all the control signals and the memory addresses. It is composed by one FSM and just one counter. The FSM consists of two states, plus the idle state. When the start input signal is asserted, the FSM passes from the idle state to the FWR1, the counter starts counting and the write_enable signal is asserted, so that the 1D-DCT partial results are stored in the transposition buffer in the position indicated by the counter address value. The input signal sel_dct indicates the length of the DCT, and then the maximum value that the counter have to reach before resetting. Once the maximum counter value is reached, the FSM passes from the FWR1 state to the FWC1 state. In this state the FSM is responsible for generating the read memory addressing, and asserts the data_out_valid signal. The maximum counter value in this state is the same of the previous state. Once the cnt_max is reached, it means that the two-dimensional transformation is completed, and the FSM passes in FWR1 state if start signal is asserted again, otherwise it passes in the idle state.

4.2 Steerable Unit

Up till now, the integer 2D-DCT architecture has been analysed. In order to implement the SDCT algorithm, a steerable architecture is placed at the output of the DCT. As said in Section 3.2, the steerable function consists in the rotation of a couple of basis vectors. While the 2D-DCT has a separable property that significantly reduces its complexity, the steerable unit is not separable, so it is a drawback for the complexity of the hardware implementation. Despite the grater complexity, attempts have been made in order not to change the throughput, therefore a greater number of resources have been allocated.

The steerable architecture is mainly composed of five blocks: an input buffer, a rotation block, an output buffer, a control unit, and a read-only memory (ROM) which is needed to addressing the reading from the input buffer and the writing to the output buffer. In the input buffer are saved the results of the 2D-DCT, then in pairs are read, the rotation block carry out the rotation scheme, and the SDCT results are stored in the output buffer, before being available on the output. These two buffer are essential, since the rotations are not applied with the same order in which DCT results are available at the steerable input.

The rotation block requires as input the value of the angle of which rotate the pairs of data. As already said before, it would be possible choose one different angle for each pairs of data, but it is not the optimal choice since after encoding, it is necessary to send the values of these angles to the decoder. Therefore, the case in which all the data of a single matrix are rotated by the same angle, is considered.

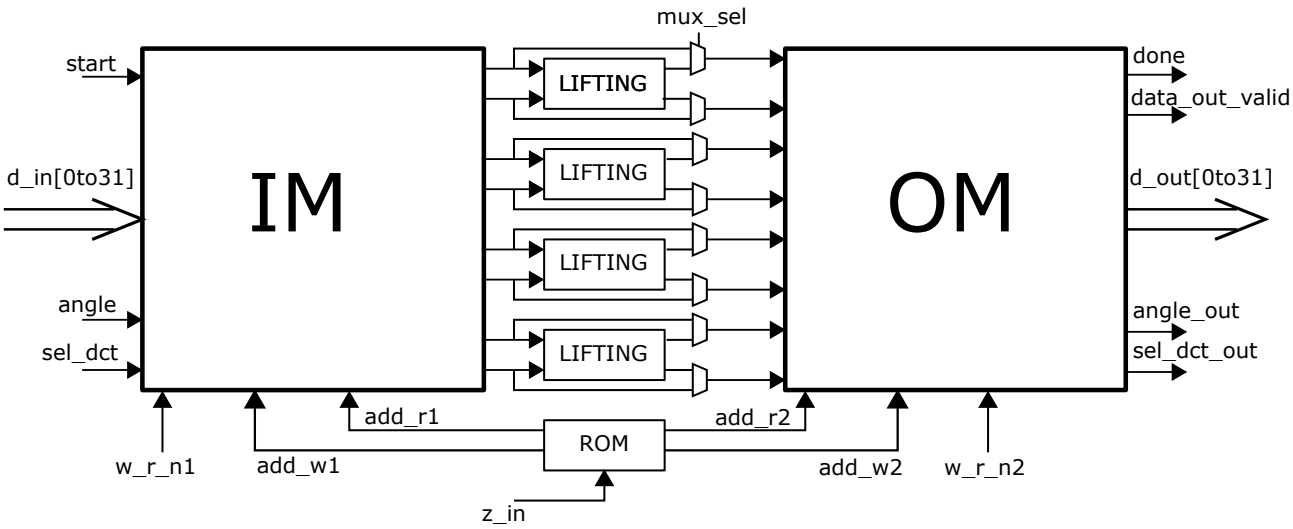


Figure 4.9. Steerable schematic

The figure 4.9 shows the schematic of the steerable top entity, that has almost the same port map of DCT unit, except the signals z_in and z_out. The signal z_in indicates the value of the rotation angle that is used for each DCT output matrix. Since the rotation are possible with eight different angle values, the range of z_in is from 0 to 8, where the value 0 means that no rotation has been applied. The output signal z_out indicates the value to which a DCT matrix has been rotated, and is sent to the decoder along with the output SDCT data.

The architecture works as follow: during the first N clock cycles the output results of 2D-DCT unit are stored in the input buffer, after that for the next N clock cycles data are rotated in pairs and stored in the output buffer. Only once the all data are stored in output buffer, this can be read for additional N cycles. That means that there is a latency of 2xN clock cycles, before data are available on exit, whereas in the 2D-DCT the latency is of N clock cycles.

However, this latency is recovered in the multiple executions, because new data are stored in the input memory while the results are available on exit. Moreover, since steerable are applied on matrices of different lengths, considering for instance the case in which a length N=32 is followed by a length N=16, in that case the first results of N=16 are provided on the exit immediately after the last results of N=32 came out, further reducing the latency.

Implementing the architecture with the same throughput of 2D-DCT unit, the area occupation of the entire architecture was too large, so it has been decided to use two different clocks. Many tests were made, with a second clock 2 times faster, 4 times faster, and 8 times faster. Finally, a clock for the steerable unit four times faster than the one of the 2D-DCT unit was chosen.

This solution leads to a large reduction of the buffers occupation area, even though the size,in terms of registers, is not reduced. Indeed, reading from the input buffer and writing to the output buffer are addressed with a zigzag scanning method, and this method requires a huge area occupation for the combinational part of both buffers. Obviously, the more is the number of input/output ports of the buffers, the larger is the buffers area occupation. In contrast, this two-clock solution requires some additional control signals, and some changes to the read-only memory.

4.2.1 Steerable Control Unit

This control unit generates all the signals to control the steerable block, and to address the two buffers. It consist of an FSM and four counters, each one with a task which will be explained below. In this section, will be explained the FSM , that is composed of 15 states. Basically, it is possible to divide states in 5 groups, according to the functionality.

In the table 4.3 has been reported the functionality of all the groups of states. All the states of a single group are similar. In other words, the task of some state is the same, but some output signal are different, and especially, each of these state has a different cnt_max. For instance, considering the states C, F, I, and L, all of them are designed to write the input buffer, and read the output buffer. However, in the state C there is a cnt_max that depends by the SDCT length (sel_dct) of the previous execution, while in the state F depends by the sel_dct of the second last execution, in I depends by the third last one and finally in L by the fourth last one. The same applies to the group D,G,H,M and the group E, E_1, E_2, E_3.

write input buffer	A
read input & write output buffer	B
write input & read output buffer	C, F, I, L
read input & write output & read output buffer	D, G, H, M
read output buffer	E, E_1, E_2, E_3

Table 4.3. FSM states functions

The state A is the one from where the steerable begins, during which in the input buffer are written the 2D-DCT results. After that, once all the N columns are written, the FSM passes to the state B, in which the data are read from the input buffer and written to the output buffer, after being rotated. Subsequently, it is necessary to read data from output buffer. In the unlikely event of single execution, the output buffer is read, but new data must not be written in the input buffer, since there are no start events. However, usually the algorithm works in multiple execution mode, so there is a new start event whenever the previous results are all written in the output buffer. Then, is necessary a state, called C, where new data are written in the input buffer, while the previous results are read from the output buffer.

Therefore, in principle, these four states are enough to execute the steerable operation, but, what has been said works only with the execution of multiple steerable with the same length. In the case of different lengths, the more are the lengths supported by the unit, the more is complicated the FSM. As said before, this unit supports lengths of N=4, 8, 16 and 32. It is precisely for this reason that so many states are required, also if some of these are similar.

There are several feasible patterns, and below have been reported some of these, in order to clarify the complex behaviour. The table 4.4 shows one of the more simple FSM execution. The execution of a steerable operation with length N=16, is followed by a new operation with a lower length, for instance N=8. In that case, after the eight columns of new data are written in the input buffer, it is necessary to read and rotate these data. The first N=16 columns of the output buffer are filled with the previous data, but not all of them have been read. So the FSM introduces an offset in the writing address, avoiding that results not yet read are overwritten. At this point, new data can be written it the output buffer, while the old are still read. The same happens with the next execution, which has a length N=4.

Instead, in general, there are no problems in the opposite situation, in which the new execution has a length higher than the previous one. The only thing evident in the table below, is that the input data of the last execution with length N=16, are written in the input memory in two consecutive states. In fact, the new data are written while the previous data are read from the output buffer, after that, since the previous execution had a lower length, the reading operation ends before writing all the new data.

memory operation	number of cycles											
Write_1	16	X	16	X	8	X	4	X	4	12	X	X
Read_1 & Write_1	X	16	X	16	X	8	X	4	X	X	16	X
Read_2	X	X	16	X	8	8	4	4	4	X	X	16

state	A	B	C	B	C	D	C	D	C	A	B	E
			16		16		8		4			16

Table 4.4. FSM states evolution, example 1

The table 4.5 shows a more complicated example, in particular it is the worst case, for which the FSM has been designed, and it includes all the states. This is the case where the largest length N=32, is followed by four times from the smallest length N=4. In this case, what happens is that while the results of the first execution are available on the output, four new execution matrix are rotated and stored in different columns of the output buffer. However, no one of these data can be read, before all data of the first execution are read from the output buffer. The final states E, E_1, E_2, E_3, are all used to read data from the output memory, but in different memory portions. Moreover, it is important to note that after an initial latency of 2xN clock cycles, there are no more latency during the execution.

memory operation	number of cycles													
Write_1	32	X	4	X	4	X	4	X	4	X	X	X	X	X
Read_1 & Write_1	X	32	X	4	X	4	X	4	X	4	X	X	X	X
Read_2	X	X	4	4	4	4	4	4	4	4	4	4	4	4

state	A	B	C	D	F	G	I	H	L	M	E3	E2	E1	E
			32								4	4	4	4

Table 4.5. FSM states evolution, example 2

The table 4.6 shows one last example to better understand the FSM behaviour.

memory operation	number of cycles											
Write_1	32	X	4	X	4	X	8	X	4	4	X	X
Read_1 & Write_1	X	32	X	4	X	4	X	8	X	X	8	X
Read_2	X	X	4	4	4	4	8	8	4	4	8	8

state	A	B	C	D	F	G	I	H	I	F	D	E
			32						4	4	8	8

Table 4.6. FSM states evolution, example 3

The control unit has in addition four counters, that are responsible to give the read and write addresses for the double buffer, in addition to control the passage between the states of the FSM.

In principle, these counters were able to count to 32 according to the SDCT length. After that, since the steerable unit has been modified introducing an additional clock, two counters of them have been modified to count up to 64 if the data parallelism was halved, or 128 if data parallelism was reduced four times. The task of each of the counters has been explained below.

Two counters are necessary in the FSM to decide the next state. Usually one counter is enough in a simple FSM, but since in this case it is needed to take into account of the previous SDCT lengths in addition to the current one, a second counter is needed. Both of them are able to count to 32, 64 or 128 according to the data parallelism. Moreover, one of these is also used to address the input buffer writing and the selection of read-only memory coefficients. The ROM is used to

address the reading from input buffer and writing to the output buffer, but an additional counter is used to select the offset of the output memory writing address. Finally, the last counter is used to address the reading from the output memory, once the SDCT results are computed.

As said before, the control unit has been designed to work with only one clock, after that a second clock has been added to reduce the area occupation without changing the throughput.

The functionality of each state has not been modified, but a special control in the outputs network has been introduced to make the modulo operation of the counter value by a constant. This constant varies depending on the second clock frequency. For instance, considering the case in which the second clock has a frequency four times higher then the other, the modulo operation is makes by 4.

Then, there will be four different conditions in each state, and certain extra signals will have a different value in each of these conditions. The added signals have the purpose to drive the MUXes added at the writing ports of the input memory, and the MUXes at the reading ports of the output memory. Moreover, there is an offset signal, that change the way in which the coefficients from the read-only memory are read.

4.2.2 Double Buffer

As mentioned previously, the output results of the DCT block need to be stored in a buffer before being rotated, and similarly, the rotated results are stored in a buffer before being valid at the output. Both of them have a size of 32x32 registers, each with a width of 16 bits.

These two buffers are indispensable because the rotations on the pairs of vectors are not applied with the same order in which DCT results are available at the steerable entry. In fact, the DCT results are provided for columns, and for N successive clock cycles data are put at the output of the 2D-DCT unit. Instead, the steerable algorithm computes the rotations according to a zigzag scanning method, which has been explained in the section 3.2.

In the following, an example of how the scanning method works has been reported. Considering for instance the case of SDCT length N=32, the data in position (column = 0, row = 1), is rotated with the data in position (column = 1, row = 0). It means that, once the first column is valid on the DCT output, it is not possible to rotate any data, because the second column must also be valid, before rotating these two data. On the base of the same scanning method, the data in position (0,31) is rotated with the one in position (31,0). It is the worst case, because one value of the first column needs to be rotated with one of the last column, so it is necessary to wait that the entire output matrix of 2D-DCT is valid, before rotating. That is why the input buffer is necessary, and it must have the same dimension of the transposition buffer (32x32 registers 16-bit wide).

Data are rotated following the scanning method, therefore, at the output of the rotation block they are available in this particular order. Nevertheless, at the output of the steerable block, data must have the same order of the input. Therefore, by the same principle, a second buffer of the same dimension is necessary on the output.

This buffer is exactly the opposite of the input one. In fact, in the input buffer, data are stored column wise and read following the zigzag method, while in the output buffer, data are stored following the zigzag method and read column wise.

The reading and writing processes for the zigzag method are extremely burdensome, especially because there are four different scanning method depending on the SDCT length. Therefore, in order to reduce the complexity and area occupation of this double buffer, it was decided to implement memories each with 8 input and output ports, instead of 32 ports. As said in the previous subsection, it is not a memory size reduction, and then the non-combinational area of each buffer is not reduced. On the other hand, the combinational area has been reduced dramatically, almost four times less. Moreover, this reduction has been compensated with a clock four time faster, therefore it has no effects on performance.

4.2.3 Read-only Memory

The aim of this block is to address the reading of data from the input buffer, following the zigzag method, after that, following the same address method, results are written in the output buffer.

Since data are read in pairs, two vectors are used to address respectively the first and second input of the rotation block. Moreover, it is worth noting that, these two vectors are different for each SDCT length. Considering the length N=4, each vector has eight elements, which number corresponds to the dimension of half matrix. In the following have been reported, as an example, the two vectors in the case of length N=8.

$$\begin{aligned} c1\_4[8] = \{ \quad 2, \quad 3, \quad 7, \quad 4, \quad 8, \quad 12, \quad 1, \quad 6\}; \\ c2\_4[8] = \{ \quad 5, \quad 9, \quad 10, \quad 13, \quad 14, \quad 15, \quad 11, \quad 16\}; \end{aligned}$$

Starting from these vectors, for each one have been derived two vectors, one that addresses the row while the other the column. This passage have been done for each SDCT length, and in the equation 4.1 has been reported the Matlab code. It should be noted that N corresponds to the SDCT length. The floor function performs a truncation.

$$\begin{aligned} c1\_N\_r[y] &= floor(c1[y]/N); \\ c2\_N\_r[y] &= floor(c2[y]/N); \\ c1\_N\_c[y] &= c1[i] - floor(c1[y]/N) * N; \\ c2\_N\_c[y] &= c2[i] - floor(c2[y]/N) * N; \end{aligned} \tag{4.1}$$

So, still considering the length N=4, the results stored in the read-only memory are the following.

$$\begin{aligned} c1\_4\_r[8] = \{ \quad 0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 2, \quad 0, \quad 1\}; \\ c1\_4\_c[8] = \{ \quad 1, \quad 2, \quad 2, \quad 3, \quad 3, \quad 3, \quad 0, \quad 1\}; \\ c2\_4\_r[8] = \{ \quad 1, \quad 2, \quad 2, \quad 3, \quad 3, \quad 3, \quad 0, \quad 1\}; \\ c2\_4\_c[8] = \{ \quad 0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 2, \quad 2, \quad 3\}; \end{aligned}$$

Therefore, in the read-only memory have been stored these four addressing vectors for each SDCT length. In total, there are 16 vectors of different size, which size depends on the SDCT length, allocated in four ROM.

Moreover, as already said in the section 4.1, since the unit can compute, at the same time, eight 4-point SDCTs, or four 8-point, or two 16-point, or one 32-point, a process give out the addresses accordingly to the length. While the output column-address signals are in common between input and output buffers, the output row-address signals are different, because of the writing offset of the output buffer.

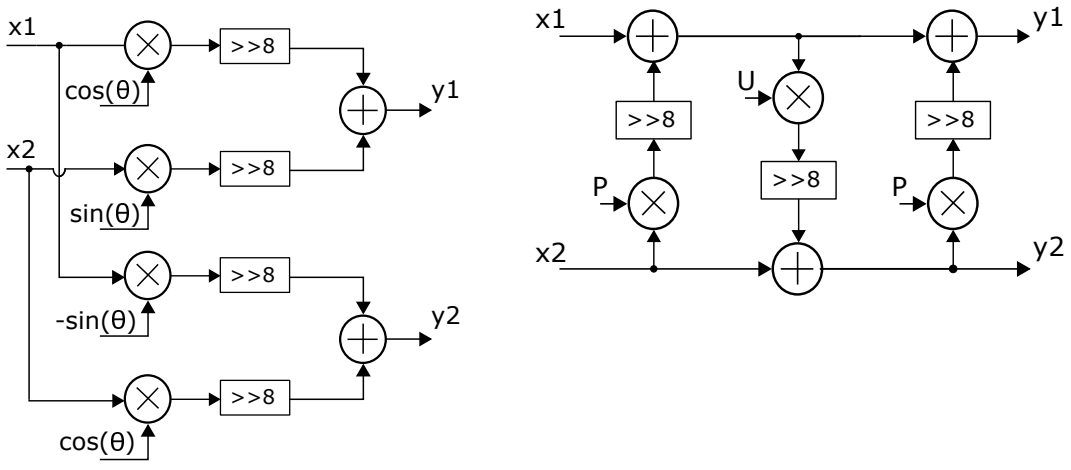
Moreover, since the second clock domain has been introduced, to reduce the number of ports of the buffers, a new additional offset signal is necessary. Indeed, since the number of ports has been reduced, the number of addresses have been reduced too, so the offset signal allows to reallocate the addresses.

4.2.4 Rotation Block

This block is essential to perform the rotations needed to apply the steerable algorithm on the results of the 2D-DCT block. As explained in section 3.2, a plane rotation corresponds to:

$$\begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} = \begin{pmatrix} 1 & \frac{1-cos\theta}{sin\theta} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -sin\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1-cos\theta}{sin\theta} \\ 0 & 1 \end{pmatrix} \tag{4.2}$$

As it can be observed from these two matrix representations, there are two methods to calculate the rotations of the angle θ. Below, in figure 4.10 the block diagrams of both methods have been reported.



pure rotation (a)

lifting scheme (b)

Figure 4.10. Rotation schemes

As it can be observed, the first method is implemented by the use of four multipliers and two adders, while the second method is implemented by three multipliers and three adders. It can be seen that the first diagram has a briefer critical path, which consists of one adder and one multiplier, while in the lifting structure, the critical path cuts a cross all the six components. The drawback of using the pure rotation scheme, consists in the use of 4 generic multipliers, which leads to a large area occupation, hence higher power consumption. The second difference between these two structures, concerns the discrepancy of the results. Indeed, in the lifting scheme the coefficients U and P are quantized. This implies that the results are slightly different, but instead the multiplication operations are simpler. In [4], [5], has been demonstrated that this little difference in the results does not lead to a

PSNR degradation. Moreover, the lifting scheme has the advantage to be perfectly invertible, so that it could be used for both coding and encoding operations.

It is worth noting that, in both cases, an adjustment of the range has been applied. This is because trigonometric functions $cos(\theta)$ and $sin(\theta)$, have an output range between 0 and 1. Therefore, in order to have only integer operations, this range has been shifted between 0 and 256, by multiplying the coefficient P and U by 256. In order to compensate this shift, it is enough to apply a right shift of eight bits, at the output of each multiplier. Below, in the equation 4.3 have been reported the equations which perform the lifting scheme, while in the equations 4.4, 4.5 have reported the values of the multiplier coefficients.

$$\begin{aligned} &lift\_temp = mtx[c1[y]-1] + ((P * mtx[c2[y]-1]) >> 8); \\ &rot\_mtx[c2[y]-1] = mtx[c2[y]-1] - ((U * lift\_temp) >> 8); \\ &rot\_mtx[c1[y]-1] = lift\_temp + ((P * rot_mtx[c2[y]-1]) >> 8); \end{aligned} \tag{4.3}$$

$$U = round((sin(angle)) * 256); \tag{4.4}$$

$$P = \begin{cases} 0 & if \quad z = 0 \\ round(((1 - cos(angle))/sin(angle)) * 256); & \Leftrightarrow \quad \text{otherwise} \end{cases} \tag{4.5}$$

In the equation 4.6, the value of the angle θ is reported. It should be noted that its value depends by the number of angles chosen. As exposed in [5], the values for which it is worth investigating are 4, 8, 16. Moreover, the better trade-off in terms of power consumption and PSNR has been reached with a value equal to 8. Therefore, the lifting architecture has been developed for a number of angles equal to 8, but it could works with a grater number of angles.

$$angle = \frac{z * \pi}{2 * NUM\_ANGLES} \tag{4.6}$$

In order to have an area occupation as small as possible, it has been decided to implement a multiplier-freee lifting architecture. Indeed, since the coefficients P and U take definite values, only adders and shifter have been used to apply the rotations. For instance, considering to have an angle z equal to 1, the value of P will be equal to 25. Therefore, the input x_2 must be multiplied by 25, but this operation can be written as sum and shift, by splitting the value 25 in power of two coefficients. In this particular case, 25 can be written as 16 + 4 + 2 + 2 + 1, so $25 * x_2$ can be computed as follows:

$$x_2 << 4 + x_2 << 2 + x_2 << 1 + x_2 << 1 + x_2$$

In order to use this multiplier-free approach, two look-up tables have been implemented. For each angle value, in these LUT are stored 6 values for the calculation of the U coefficient, and 4 values for the P coefficient. In practice, these values correspond to the shift quantity of each adder operand. Indeed, the constant P coefficient multiplier is replaced by an adder with 5 input and 4 shift unit, and the constant U coefficient multiplier is replaced by an adder with 7 input and 6 shift unit.

In tables 4.7 and 4.8 are reported the shift-add operations to compute each constant multiplication. The angle z=0 is not reported since in this case are not carried out rotations, and the SDCT outputs match the 2D-DCT ones.

z	P	P shift-add
1	25	y*16 + y*4 + y*2 + y*2 + y
2	51	y*32 + y*16 + y + y + y
3	78	y*64 + y*8 + y*4 + y + y
4	106	y*64 + y*32 + y*8 + y + y
5	137	y*128 + y*4 + y*2 + y*2 + y
6	171	y*128 + y*32 + y*8 + y*2 + y
7	210	y*128 + y*64 + y*16 + y + y

Table 4.7. Lifting shift-adder coefficients for P multiplier

z	U	U shift-add
1	50	y*16 + y*16 + y*8 + y*4 + y*4 + y + y
2	98	y*32 + y*32 + y*16 + y*8 + y*8 + y + y
3	142	y*64 + y*32 + y*32 + y*8 + y*4 + y + y
4	181	y*128 + y*16 + y*16 + y*16 + y*2 + y*2 + y
5	213	y*128 + y*32 + y*32 + y*16 + y*2 + y*2 + y
6	237	y*128 + y*64 + y*32 + y*8 + y*2 + y*2 + y
7	251	y*128 + y*64 + y*32 + y*16 + y*8 + y*2 + y

Table 4.8. Lifting shift-adder coefficients for U multiplier

In addition, four pipe levels have been introduced, since the timing constraint was not respected without them. The figure 4.11 shows the complete lifting architecture, including the pipe levels. As it can be observed, the critical path of the new structure is composed by five adders, instead of an adder and a multiplier. The shift values in the example figure, are those needed to rotate the vectors of an angle z=6, then the values of the coefficients are U=237 and P=171. In fact, as said before, the shift values are not fixed but depend on the angle value.

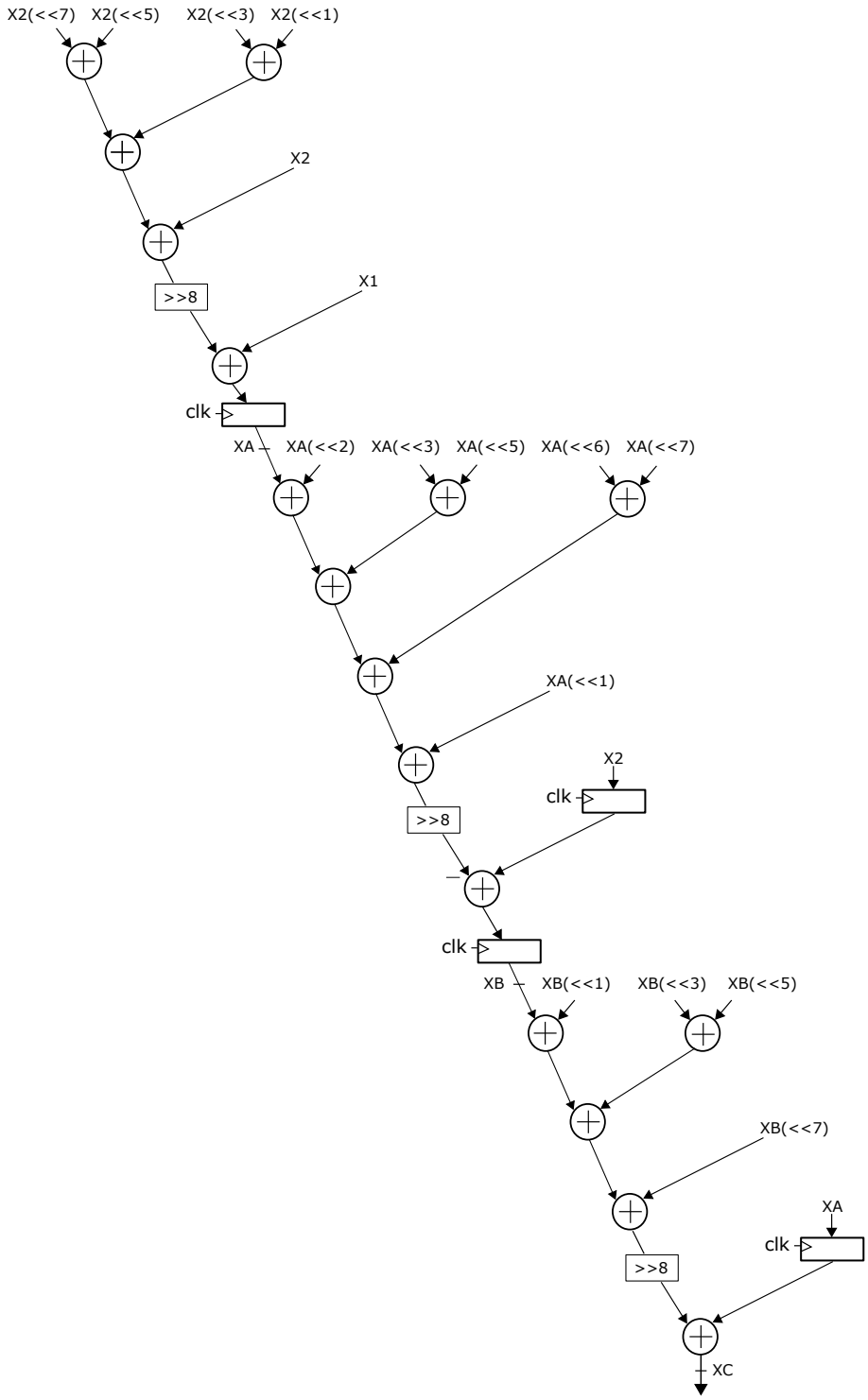


Figure 4.11. Lifting data flow graph for z=6

4.2.5 Re-timing Unit

This unit has the aim to interface the 2D-DCT unit with the steerable unit. The folded 2D-DCT unit, takes in input N successive columns, after that, for the next N cycle are not taken input data. In order to function properly, the steerable unit needs to have the same input timing of the 2D-DCT unit. Unfortunately, this is not possible without the use of an additional re-timing unit. In particular, this problem appears during the execution of SDCT operation of different lengths.

The figure 4.12 shows the case in which a SDCT computation of size N=32 is followed by other four SDCT of size N=4. The signal data_in_valid is asserted when the data in input to the 2D-DCT unit are valid, while the signal data_out_valid_int is asserted when the data in output to the 2D-DCT unit are valid, and these data should be valid at the input of the steerable unit. As it can be noticed, data_in_valid is asserted for N=32 clock cycles, after that is not valid for other 32 cycles. Then, since the following SDCT has a size N=4, it is asserted again for N=4 clock cycles. In contrast, the signal data_out_valid_int is asserted for N=32 clock cycles, after that it is not valid for N=4 cycles, before being asserted for N=4 cycles. So, it is evident that the behaviour of the signal data_in_valid differs from the one of data_out_valid_int. Therefore, a re-timing unit has been introduced. The signal data_out_valid_fifo is asserted when the data in input to the steerable unit are valid, and it can be noted that its behaviour is the same of the signal data_in_valid.

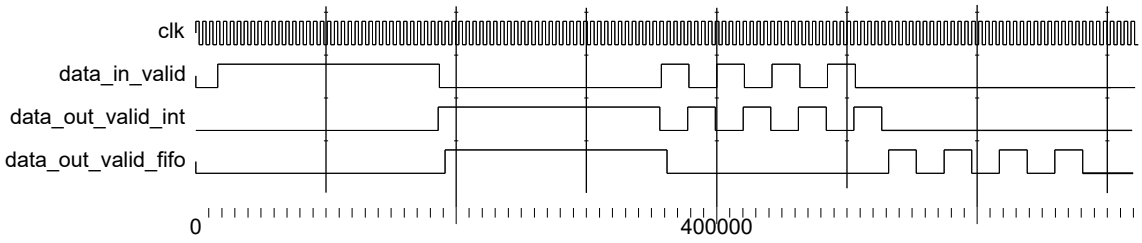


Figure 4.12. Timing diagram re-timing unit

This re-timing unit consists of a FIFO buffer, with a dimension of 32 rows x 16 columns registers, each with a width of 16 bits. The FIFO is managed by an FSM and four counters. The FSM has the same principle of that of the steerable unit. In practice, also in this FSM the states are divided in groups. All the states of a single group have the same task, but each of these state has a different cnt_max, and some output signal are different. In table 4.9 the list of all the states with the respective task has been reported.

If the data in output to the 2D-DCT unit can be rotated immediately, so the FSM is in the state A and a MUX let passes data from DCT unit, without store them. In this case data_out_valid_int coincides with data_out_valid_fifo. In all the other cases, data are stored in FIFO, before being rotated. This FIFO supports the reading and writing processes contemporaneously. Therefore, two counter are used to address the writing and reading operations in FIFO. The other two counter are needed to handle the transition between the various states of the FSM.

bypass MUX	A
write FIFO	C, E, G, I
wait	B, D, F, H
read FIFO	O, J, K, L
read & write FIFO	P, M, N

Table 4.9. FSM states functions for re-timing CU

The figure 4.13 shows two timing diagrams of SDCT execution. In the diagrams have been listed the input and output validation signals, and the states' evolution of both FSM (steerable FSM and re-timing FSM). In the first diagram, an execution with length N=32 is followed by four executions with length N=4. In the second diagram, an execution with length N=32 is followed by two N=8, one N=4, one N=16 and finally one N=8 SDCTs.

These timing diagrams has been reported in order to highlight the latency introduced by the steerable unit, which can be measured as the distance between the signal data_out_valid_int (2D-DCT data_out_valid) and data_out_valid (SDCT data_out_valid). In fact, as already said in section 4.2, the steerable introduces a latency of 2xN clock cycles, plus 4 more clock cycles latency due to the pipeline of the lifting module.

Therefore, in the event that all the SDCT have a length N=32, the latency introduced is equal to 68 clock cycles. So, in general, since N=32 is the maximum SDCT length, it corresponds to the maximum latency. However, executing SDCT of different lengths, some clock cycles of latency are recovered. In particular, in the following diagrams, the latency is respectively equal to 51 and 43 clock cycles.

In conclusion, it can be noticed that the latency does not have fixed value. However, its value is acceptable in this specific application.

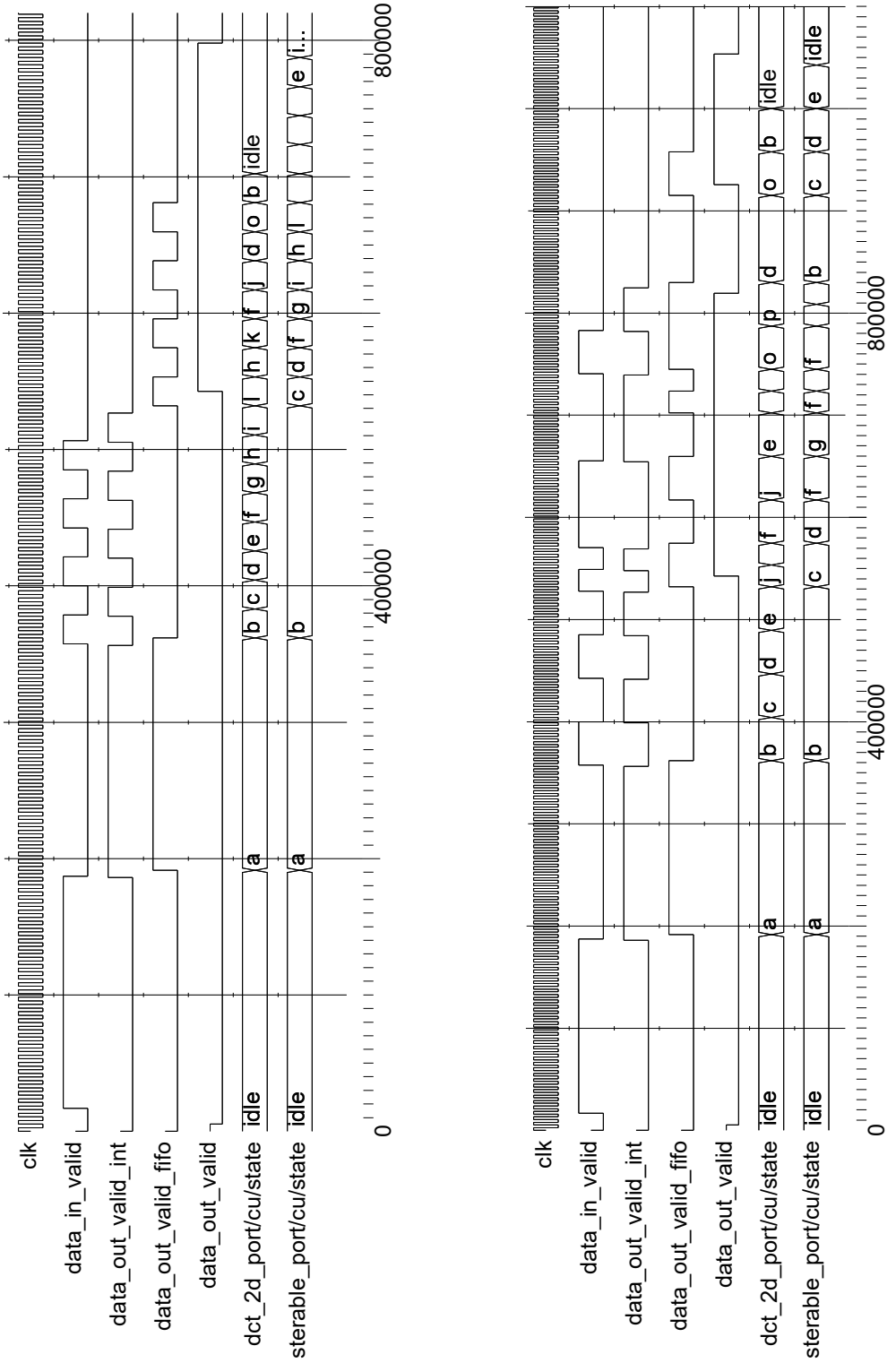


Figure 4.13. Timing diagrams of SDCT

4.2.6 SDCT Architecture

In figure 4.14 and table 4.10 the top entity of SDCT architecture with the respective input and output signals have been reported. As it can be observed, it is constituted by three different units, with the respective control units.
In general, it works as follows: firstly, as soon as the start signal arrives, the two-dimesional DCT computes the spatial transform. So, once its output data are valid, these are provided to the steerable unit, which performs the rotations according to the rotation angle. It should be noted the presence of the FIFO between the two units. This stores the data while the steerable is unavailable, to provide them as soon as the latter is ready to receive new input. It is worth noting the control structure, with three different units, one for each block.

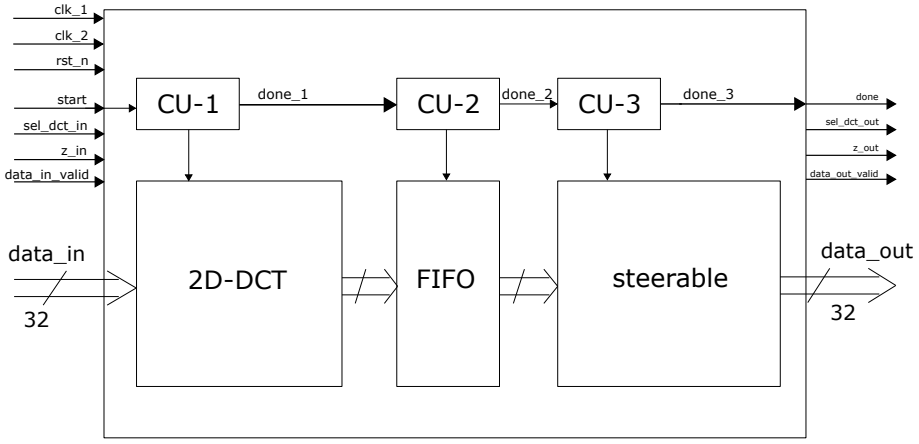


Figure 4.14. SDCT schematic

Signal	Input/Output	Signal feature
clk_1	Input	clock signal DCT unit
clk_2	Input	clock signal Steerable unit
rst_n	Input	asynchronous active-low reset signal
sel_dct	Input/Output	indicates the SDCT length
z	Input/Output	indicates the rotation angle
start	Input	indicates the start of SDCT computation
data_in[0 to 31]	Input	32 9-bit wide input data
data_in_valid	Input	indicates the input data validity
data_out[0 to 31]	Output	32 16-bit wide output data
data_out_valid	Output	indicates the output data validity

Table 4.10. In/Out signals for SDCT architecture

4.3 Reduced SDCT

The unit that has been presented so far, is able to compute SDCT of lengths 4, 8, 16 and 32. This type of structure has been designed to be implemented inside the HEVC standard. Anyway, this algorithm could be used also for video compression standards with lower constraints and for image compression standard, such as JPEG. Therefore, two reduced SDCT unit have also been realized, one that is able to compute SDCT of length 4, 8 and 16, named SDCT-16, and another one capable of computing SDCT of length 4 and 8, named SDCT-8.

These two units have a reduced throughput of 50% and 75% respectively, so they have a parallelism of 16 or 8 data, instead of 32. Therefore, the size of all the memories is reduced. In particular the length of both rows and columns of all memories is halved in the SDCT-16 unit, while is four time lower in the SDCT-8 unit. Since the area occupation of these units is much lower than the SDCT-32 unit, the parallelism of the steerable block must not be further reduced, so just one clock domain has been used for both DCT and steerable block.

A future development could be to keep the throughput unchanged. In this case the data parallelism should not be reduced, therefore only the number of columns of each memory should be changed, while the number of rows should be left unchanged.

4.4 Clock Gating

In order to further optimise the power consumption of the SDCT unit, the clock gating technique has been adopted. This technique may be applied on all those register which have an enable signals. It is applied during the synthesis, without change the VHDL code, and it does not impact on the timing performance and on the results of the unit.

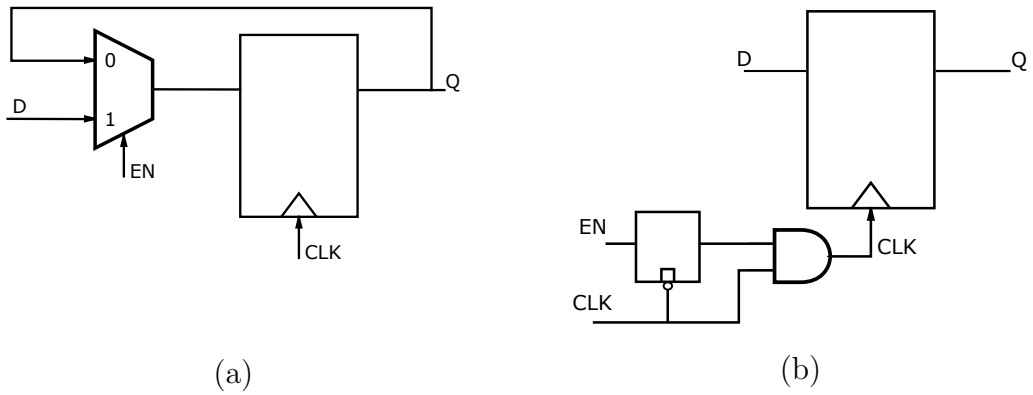


Figure 4.15. Clock gating scheme

Figure 4.15 shows the implementation of a load-enable register without clock gating in the case (a), and with clock gating in the case (b). As it can be noticed, if the clock gating technique is not adopted, the load enable register is implemented by a multiplexer driven by the enable signal and placed on the register input.

On the contrary, if the clock gating technique is adopted, there are some way to implement it. The more reliable one is to use an active-low latch that sample the enable signal, and its output is put in AND with the clock. This solution has no problem if the enable signal has some glitches, but requires that enable signal must be stable at least a setup time before the rising edge of the clock, and a hold time after the rising edge.

The clock gating leads to benefit in terms of both area occupation and power consumption. In fact, all the multiplexers at the input of the registers are replaced by some latches. It is worth noting that a latch is shared by all the registers which have the same enable signal. Therefore, the number of latch-based cells introduced will be less than the number of multiplexers removed. Moreover, the dynamic power is reduced since the registers do not commutate if the enable signal is not asserted. Another big advantage is about the clock network, and its power consumption. Indeed, applying the clock gating technique, the capacity driven by the clock network is reduced, and more importantly, the number of registers that commutate contemporaneously is much smaller.

Chapter 5

Synthesis and Results

In this chapter are shown the results in terms of area and power of SDCT architecture. More synthesis have been carried out for different frequencies, so that the SDCT can also be used for video resolutions smaller than 7680x4320 at 60 fps. Moreover, SDCT of reduced length have been synthesized, so that this directional transform can also be used for video compression standard with lower constraints and for image compression standard.
It should be noted that the synthesis have been made, first, without applying any power optimization, after that the clock gating optimization has been applied.
In this thesis work, all the designed versions have been synthesized using the UMC 65nm standard-cell library.

5.1 Design Flow

Once fixed the software parameters, the architecture of both 2D-DCT and steerable units has been defined, making some choices on the internal parallelism, in order to satisfy the HEVC timing requirements. Then, all the blocks have been described using the VHDL language. Once the correct behaviour has been verified using the ModelSim simulation environment, the design has been synthesized by the use of Synopsys Design Compiler.

The Design Compiler has the aim to generate the gate-level description, starting from VHDL code, then generates a netlist in verilog. The netlist is then simulated with ModelSim, together with another verilog file which contains the functional model of the cells for the given standard-cell library, and a standard delay format (SDF) file, which contains the delay of each cell and node of the circuit. Several tests have been conducted to verify the correct behaviour of netlist file, and to verify that timing requirements were satisfied.

Finally, the power has been estimated adopting the SAIF annotation method, using the combination of design compiler and simulation tools. This file contains the annotation of the switching activity of each cell and node of the circuit. Since the netlist complexity is quite high, the 32-bit ModelSim version is not able to generate the SAIF annotation, therefore, a 64-bit ModelSim version has been used.

However, this version has not the possibility to generate the SAIF file directly, therefore a slightly different method has been used. In practice, starting from a value-change-dump (vcd) file, generated on ModelSim, the switching activity SAIF annotation has been obtained with the Synopsys command *vcd2saif*. At this point, using the usual approach, this file has been provided to Synopsys DC, which estimates the static and dynamic power consumption.

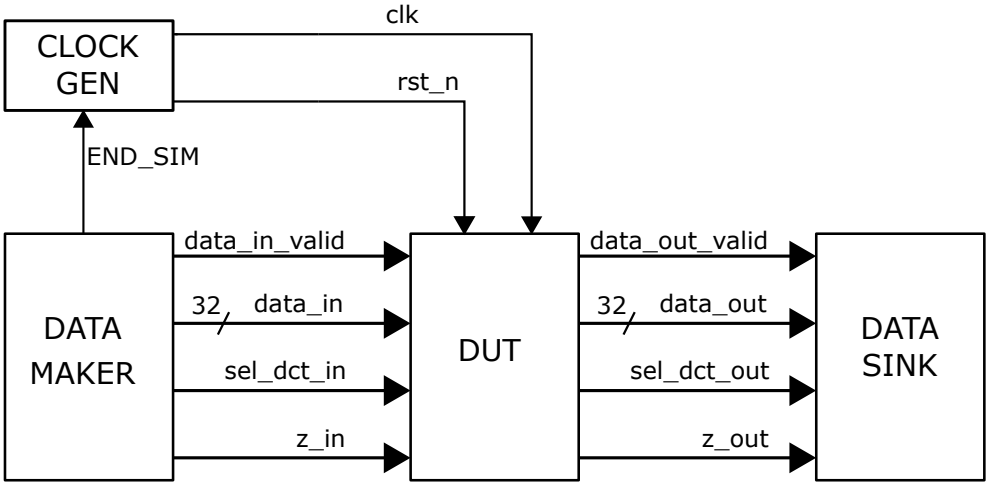


Figure 5.1. Testbench

In figure 5.1, the structure of the testbench has been reported. This testbench is used both to simulate the described SDCT unit, and the netlist in verilog produced by Design Compiler. It consists mainly of four components:

- *DUT (device under test)*, which is the designed SDCT unit
- *clock generator*, which generates the reset (rst_n) signal and the clock (clk) signal
- *data maker*, which takes the data stored in a text file and feeds the SDCT unit with these data. It is worth noting that these data perfectly reproduce the real operation of the steerable DCT unit. In fact, these are taken from the HM software, during the encoding of a video frame. In addition to the input

matrices, the data maker provides sel_dct (which indicates the size of each matrix), and z_in (which indicate the rotation angle of each matrix). Lastly it provide the END_SIM signal, which indicates the end of the simulation, and stops the clock signal.

- *data sink*, which receives the output data from SDCT unit, and save them in a second text file, in the same order in which they were provided at the input. Finally, these data are compared with the results of HM software, in order to verify the correct operation of the designed SDCT unit.

5.1.1 Design Compiler Settings

In order to satisfy the HEVC speed requirements for a video resolution of 7680x4320 and a frame rate of 60 fps, the proposed structure needs a throughput of almost 3 Gsps. As said in previous chapter, in order to save area, the architectural choice that has been adopted is the folded version, since this approach guarantees the required throughput. This folded version has a processing rate of 16 pixels per cycle, therefore the architecture needs a frequency of at least 187 MHz ($2.99 \times 10^9/16$).

In Design Compiler a clock of period 5.32 ns has been created. In addition, a clock uncertainty of 0.07 ns has been set, because jitter could affect the clock period. Moreover, a delay in input of 0.5 ns has been set, because each signal may have a time delay with respect to the clock signal. Similarly, also a delay in output of 0.5 ns has been set.

Finally, the load of each output node of the design has been set. For simplicity, the input capacitance of a buffer of UMC 65nm technology has has been set as load of each output node.

At this point the synthesis can be launched, and the gate-level description is obtained starting from the VHDL code. In order to optimize the area occupation as much as possible, the *compile_ultra* command is used. The synthesis is launched through the use of the *compile_ultra* command, and the gate-level description that best fits the time constraints is carried out. In particular, this command performs a set of automatic optimizations, in order to minimize the area occupation as much as possible.

Once the synthesis is completed, have been generated the netlist in verilog and SDF files, required to perform the power estimation based on switching activity informations. In detail, the netlist contains the description of cells and nodes of the architecture, while the SDF file contains the delay of each cell and node.

5.1.2 Clock Gating Option

The area occupation and consequently the power consumption can be reduced by using the *compile_ultra -gate_clock* command, which performs a completely automatic clock gating optimization. In practice, as said in section 4.4, the clock gating technique leads to benefit in terms of both area occupation and power consumption. This happens mainly because the number of registers that commutate contemporaneously is much smaller. It leads to a reduction of the capacity that the clock network should drive, and consequently smaller clock drivers are synthesized.

The Design Compiler applies the clock gating technique by removing the input multiplexer to the register which have an enable signal, and inserts an active-low latch for each group of register. This leads to a further area reduction due to the removal of multiplexer, in addition to the power reduction because the registers do not switch if the enable signal is not asserted.

The clock gating has been performed by using the *compile_ultra -gate_clock* command. In addition, some other commands has been added during the definition of the constraints, before starting synthesis. In particular, the *set power_cg_auto_identify true* command is necessary to auto recognize clock gate cells in the netlist.

5.2 Results

In this section the results in terms of area occupation and power consumption of all the synthesized architectures, both without and with clock gating optimizations, have been reported. More specifically, the synthesized architectures are the following:

- two-dimensional DCT
- pruned two-dimensional DCT
- SDCT
- reduced SDCT-16
- reduced SDCT-8

5.2.1 Area Occupation

The area occupation results of the 2D-DCT are shown below, in table 5.1 and 5.2. In particular, in the first table the results achieved without low-power optimization are shown, while in the second one the results achieved with the clock gating technique are shown. These have been both obtained by using an operating frequency of 188MHz.

cell	total area	percentage	Combinational	Noncombinational
2D-DCT	378862 μm^2	100	305315 μm^2	73547 μm^2
1D-DCT	235611 μm^2	62	235545 μm^2	66 μm^2
TM	143251 μm^2	38	69770 μm^2	73481 μm^2

Table 5.1. Area distribution of basic 2D-DCT architecture

cell	total area	percentage	Combinational	Noncombinational
2D-DCT	349067 μm^2	100	273757 μm^2	75310 μm^2
1D-DCT	236222 μm^2	68	236151 μm^2	71 μm^2
TM	112945 μm^2	32	37606 μm^2	(73481 + 1758) μm^2

Table 5.2. Area distribution of clock gated 2D-DCT architecture

The tables 5.1 and 5.2 show in details the combinational and noncombinational contributes, for the top entity architecture and its sub-blocks. As can be seen, in both cases, the main contribution is the combinational one. It is also worth noting that the transpose buffer (TM) has a great contribution on the total area. In particular, it has a great combinational contribution, due to the huge quantity of multiplexer that allow to transpose the contents of the buffer.

Applying the clock gating technique, the area occupation decreases by approximately 10%. In detail, as it can be seen by comparing the two tables, the combinational contribution decreases, while the noncombinational one increases slightly. This is especially the case for the transpose buffer. It happens because, applying clock gating, the multiplexers of each register are substituted by some latches. Therefore, the combinational area of transpose buffer decreases by removing the MUXes, while the noncobinational area increases slightly by adding the latches.

It is worth noting that the area contribution added by inserting the latches is much lower than the one removed by removing the MUXes. Indeed, without applying clock gating, each register cell has a multiplexer. Therefore, a huge quantity of MUXes is removed thanks to clock gating. On the contrary, the latch-based cell is not added on each register, but a cell is shared by registers that share the same enable signal. In total, as it can be seen by table 5.2, the added area contribution due to latch-based cell is equal to 1758 μm^2, which corresponds to the addition of 257 latch-based cells, while 32x32x14 MUXes have been removed.

The table 5.3 shows the area occupation of the pruned version of the 2D-DCT architecture, which has been obtained by applying the clock gating method. As it can be seen, this pruned version provides a further area reduction of around 10%, with respect the integer clock gated architecture. In particular, it is worth noting that the area reduction regards mainly the 1D-DCT unit, because its precision has been reduced. On the contrary, the size of the transpose buffer of the pruned version DCT is the same as that in the full integer DCT version, since the bit-width of the data to be stored is unchanged. Therefore, the area occupation of the transpose buffer is almost unchanged.

cell	total area	percentage	Combinational	Noncombinational
2D-DCT	321372 μm^2	100	273757 μm^2	75310 μm^2
1D-DCT	208997 μm^2	64	208930 μm^2	67 μm^2
TM	112375 μm^2	36	37551 μm^2	(73481 + 1343) μm^2

Table 5.3. Area distribution of clock gated pruned 2D-DCT architecture

The architectural implementation can be seen into two distinct parts: an integer 2D-DCT and a steerable unit placed at the output of the DCT. Up to now, the area occupation of 2D-DCT has been analysed in detail, while the area occupation of the complete SDCT architecture will be analysed below.

As mentioned above in section 4.2, while the 2D-DCT has a separable property that significantly reduces its complexity, the steerable unit is not separable, so it is a drawback for the complexity of hardware implementation. Although the grater complexity, attempts have been made so as not to change the throughput, therefore the same operating frequency of 188MHz has been used. Implementing the architecture with the same throughput of 2D-DCT unit, the area occupation of the entire architecture was too large, so it has been decided to use two different clocks. Many tests were made, with a second clock 2 times faster, 4 times faster, and 8 times faster. The next table (5.4) shows the total area occupation of the various synthesized architectures, with a reference to its sub-blocks. It is not worth using a clock 8 times faster, since the area reduction with respect the 4-times solution is minimal. And consequently, the expected power consumption is too high, because the area reduction do not compensates the increase in frequency.

Therefore, the best solution is the one with a clock for the Steerable unit four times faster than the one of the 2D-DCT unit. This solution leads to a large reduction of the buffers occupation area, even though the size,in terms of registers, is not reduced. Indeed, these buffers need a lot of combinational ports (mostly MUXes). Therefore, by increasing the frequency, it is possible to decrease the parallelism and consequently the number of input/output ports of the buffers.

cell	1- total area	2-total area	4-total area	8-total area
SDCT	4337744 μm^2	3042226 μm^2	1608759 μm^2	1301522 μm^2
2D-DCT	438866	601970	455150	474167
IM	1401523	820032	495856	335932
OM	2377837	1418162	482048	319037
FIFO	86542	110594	113008	110604
ROM	5895	22228	13227	33223

Table 5.4. Area distribution of SDCT configurations with different parallelism

By comparing the results in table 5.4, it can be noticed that by reducing the data parallelism of the Steerable unit, the dimension of the input buffer (IM) and output buffer (OM) decrease considerably, while the dimension of all the other sub-blocks increases slightly. It happens because the Design Compiler try to optimize as much

as possible the area of other blocks, if the area of IM and OM buffers is too large. Therefore, reducing the buffers parallelism, Synopsys DC makes less optimization on other blocks.

Considering the chosen solution with the steerable clock four time faster, the area occupation results of the SDCT are shown below, in table 5.5 and 5.6. In particular, in the first table the results achieved without low-power optimization are shown, while in the second one are shown those achieved with the clock gating technique. These have been both obtained by using an operating frequency of 188MHz (5.32ns) for the 2D-DCT clock, and an operating frequency of 752MHz (1.33ns) for the steerable clock.

cell	total area	percentage	Combinational	Noncombinational
SDCT	1608759 μm^2	100	1118746 μm^2	490013 μm^2
2D-DCT	455150	28.3	326342	128807
IM	495856	30.8	364982	130874
OM	482048	30.0	348046	134001
FIFO	113008	7.0	43151	69857
ROM	13227	0.8	12235	991

Table 5.5. Area distribution of basic SDCT architecture

cell	total area	percentage	Combinational	Noncombinational
SDCT	1427105 μm^2	100	928603 μm^2	(488485 + 10017) μm^2
2D-DCT	398833	27.9	268505	128747
IM	451387	31.6	319793	130665
OM	426904	29.9	285902	133968
FIFO	87235	6.1	18669	68382
ROM	12759	0.9	11769	983

Table 5.6. Area distribution of clock gated SDCT architecture

As it can be observed, in both cases, the combinational contribution is the largest one. In particular, the two input and output buffers have a large quantity of combinational area. Just as previously seen for the 2D-DCT unit, also in this case the area is reduced by approximately 10% applying the clock gating technique. In table

5.6, the quantity of noncombinational contribution due to the clock gating is given in brackets. Obviously, in SDCT architecture, the added area contribution due to latch-based cell insertion is much higher than in 2D-DCT. This is because in SDCT, in addition to the transposition buffer, there are two others buffers of the same size.

Another consideration to make on this SDCT architecture regards its operating frequency. As mentioned, more synthesis have been carried out for different frequencies, so that the SDCT can also be used for smaller video resolutions. In table 5.7, the area results depending on the synthesis operating frequency are shown. As it can be noticed, the area is almost frequency independent in which its values in the three different cases are similar. A reason could be that the critical path is along the output ports of the input buffer, which is basically composed of the series of some MUXes. So, decreasing the clock frequency, the Design Compiler decreases the sizes of this MUXes, but in general the area reduction is really low. In contrast, there will be a really high power reduction, but this will be shown in the next section.

Frequeny	100MHz	150MHz	188MHz
Period (clk & clk_4)	10.00 & 2.50 ns	6.68 & 1.67ns	5.32 & 1.33ns
Noncombinational	498543 μm^2	498456 μm^2	498502 μm^2
Combinational	915787 μm^2	917105 μm^2	928603 μm^2
Total area	1414331 μm^2	1415561 μm^2	1427105 μm^2

Table 5.7. Area distribution of clock gated SDCT for different frequencies

The last two tables show the results of two reduced SDCT architectures. These architectures are designed to be implemented in image compression standard, such as JPEG, or in the previous video compression standard, such as H.264. In practice, two scaled architecture, that respectively support lengths n = 4, 8, and 16 (SDCT-16) and lengths n = 4, and 8 (SDCT-8) have been proposed.

cell	basic SDCT	clock gated SDCT
SDCT	500811	440099
2D-DCT	121353	110681
IM	173349	152401
OM	177252	154273
FIFO	25760	19582

Table 5.8. SDCT-16 area distribution @188MHz

cell	basic SDCT	clock gated SDCT
SDCT	121194	109945
2D-DCT	25937	23902
IM	32627	28406
OM	33024	28505
FIFO	27016	25984

Table 5.9. SDCT-8 area distribution @188MHz

These architectures have a really reduced dimension if compared with the complete one proposed previously. Indeed, the SDCT of length n=32 is the bottleneck of the structure since it need largest buffers, and above all, a largest number of input/output ports.

The area occupation results of reduced SDCT-16 and SDCT-8 units are shown in table 5.8 and 5.9. The two structures have been obtained by using the usual operating frequency of 188MHz (5.32ns), and without using a second clock with higher frequency for the steerable unit. Moreover, the clock gating technique has been exploited.

5.2.2 Power Consumption

The last step was the estimation of power consumption. As explained in design flow section (5.1), the power consumption is estimated by using the combination of ModelSim and Synopsys DC tools. First of all the switching activity has been obtained by simulating the netlist on ModelSim, then both switching activity and netlist are used as source files on Synopsys to estimate power consumption.

The tables show the static and dynamic power components. The dynamic component is due to the charge and discharge of the capacitors, while the static component is mainly due to the leakage currents.
The dynamic power is calculated according to the following equation:

$$P_{dynamic} = C_L \times V_{DD}^2 \times f \times \alpha$$

- C_L is the load capacity
- V_{DD} is the supply voltage
- f is the clock frequency
- α is the switching activity

The operating voltage for the adopted *UMC 65nm* library is 1.2 V.
The power consumption estimations of the 2D-DCT, both without and with clock gating optimizations, are shown below, in table 5.10. Moreover, in the last column of the table, the power estimation of the pruned 2D-DCT is shown. As it can be noticed, considering the integer DCT unit, the clock gating technique leads to a great power saving of around 25%. In addition, the pruning scheme leads to a further power saving of around 22%.
Another consideration is that the leakage power contribution for this *UMC 65nm* library is really small.

Power component	basic DCT	clock gated DCT	clock gated pruned DCT
Internal (mW)	36.55	24.88	21
Switching (mW)	17.72	15.9	12.52
Total dynamic (mW)	54.47	40.78	33.52
Leakage (μW)	33	30	30

Table 5.10. 2D-DCT power consumption @ 188MHz

In the following the power consumption results of SDCT unit are reported. As already mentioned, the SDCT architecture with a steerable clock four time faster that the DCT one has been chosen, since has the best area occupation. Therefore, the power consumption has been estimated just for this configuration. In tables 5.11, 5.12 and 5.13, the power estimation of the SDCT unit for three different frequencies have been reported (100MHz, 150MHz and 188MHz).

Power component	basic SDCT	clock gated SDCT
Internal (mW)	155.40	49.93
Switching (mW)	32.16	32.96
Total dynamic (mW)	187.68	82.99
Leakage (μW)	103	91

Table 5.11. SDCT power consumption @ 100MHz

Power component	basic SDCT	clock gated SDCT
Internal (mW)	232.36	74.05
Switching (mW)	48.84	49.46
Total dynamic (mW)	281.30	123.6
Leakage (μW)	103	92

Table 5.12. SDCT power consumption @ 150MHz

Power component	basic SDCT	clock gated SDCT
Internal (mW)	290.47	88.71
Switching (mW)	60.33	59.85
Total dynamic (mW)	350.88	148.67
Leakage (μW)	106	94

Table 5.13. SDCT power consumption @ 188MHz

Obviously, the higher is the frequency, the greater is the power consumption, due to the linear dependence between dynamic power and frequency. It is worth noting that, in the SDCT architecture, the power reduction through clock gating technique is much higher than in the DCT architecture. In general, for all the three applied frequencies, the power saving is around 65%, although, as seen before, the area reduction was around 10%. This is because the steerable unit consists mainly of two buffers, that do not consume energy if their enable signals are not asserted.

The next two tables, 5.14 and 5.15, show the power consumption estimation of the two reduced SDCT architectures. As it can be observed, the power consumption of these two reduced architectures is much lower than the complete one, although they are synthesized with the same high frequency used for HEVC.

Power component	basic SDCT	clock gated SDCT
Internal (mW)	42.52	27.86
Switching (mW)	33.43	28.97
Total dynamic (mW)	76.00	56.85
Leakage (μW)	27	27

Table 5.14. SDCT-16 power consumption @ 188MHz

Power component	basic SDCT	clock gated SDCT
Internal (mW)	10.01	6.56
Switching (mW)	6.07	7.20
Total dynamic (mW)	16.10	14.17
Leakage (μW)	7	7

Table 5.15. SDCT-8 power consumption @ 188MHz

5.3 Comparisons

In this work, a Steerable DCT implementable in HEVC standard has been proposed, starting from the integer DCT [6]. As already mentioned, the SDCT algorithm leads to an improvement of the coding efficiency of HEVC video coding.
First of all, a comparison of the designed DCT with the reference one proposed in [6] has been done. From table 5.17, it can be observed that the obtained results are in line with the reference one for both full-integer and pruned structures.

Architecture	[6]	pruned [6]	DCT	pruned DCT
Technology (nm)	90	90	65	65
Frequency (MHz)	187	187	187	187
Power (mW)	40.04	35.51	40.78	33.52
Throughput	2.992 G	2.992 G	2.992 G	2.992 G
Area (mm^2)	0.580	0.550	0.349	0.321
Energy per Sample (EPS)	13.38pJ	11.86pJ	13.63pJ	11.20pJ

Table 5.16. Comparison of DCT and pruned DCT with respect to the two state of art architectures discussed in [6]

However, for what concerns the SDCT architecture, there are no similar architectures available for which some comparisons could be done. For that reason, the only comparison that has been done is the one between the Steerable DCT transform and the integer DCT.
Obviously, since additional hardware has been added, the power consumption is higher, but this consumption should be compensated by the reduction of the bitrate needed to send the encoded data.

Finally, the summary table of all the architectures designed has been reported. In particular, all these results have been obtained with the usual operating frequency of 187MHz, which is the frequency indicated in order to adopt these transforms in the HEVC standard.

Architecture	DCT	pruned DCT	SDCT	SDCT-16	SDCT-8
Technology (nm)	65	65	65	65	65
Frequency (MHz)	187	187	187	187	187
Power (mW)	40.78	33.52	148.67	56.85	14.17
Throughput	2.992 G	2.992 G	2.992 G	1.496 G	0.748 G
Area (mm^2)	0.349	0.321	1.427	0.444	0.110

Table 5.17. Comparison of the designed architectures

As it can be noticed, the area and power results of the SDCT-16 are around 60% smaller than the complete SDCT. On the other hand, the SDCT-8 results are around 75% smaller than the SDCT-16 and 90% smaller than the complete SDCT. On the contrary, the throughput is reduced of 50% and 75% respectively.
It is worth noting that SDCT-8 architecture could be used in JPEG image standard to replace the typical DCT algorithm. In fact, the Steerable DCT could be useful to reduce the image artefacts introduced by the encoding of images that contain shaped discontinuities.

Bibliography

[1] [Online]. Available: https://www.edn.com/Home/PrintView?contentItemId=4438600

[2] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "Hevc complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[4] G. Fracastoro, S. M. Fosson, and E. Magli, "Steerable discrete cosine transform," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 303–314, 2017.

[5] M. Masera, G. Fracastoro, M. Martina, and E. Magli, "A framework for designing directional linear transforms with application to video compression and encryption," *Signal Processing: Image Communication*, 2018.

[6] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer dct architectures for hevc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 168–178, 2014.

[7] M. Budagavi, A. Fuldseth, G. Bjontegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (hevc) standard," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1029–1041, 2013.

[8] B. Zeng and J. Fu, "Directional discrete cosine transforms-a new framework for image coding," *IEEE transactions on circuits and systems for video technology*, vol. 18, no. 3, pp. 305–313, 2008.

[9] J. C. T. on Video Coding (JCT-VC). Hm 16.6 reference software. [Online]. Available: https://hevc.hhi.fraunhofer.de/