INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

ÁREA DEPARTAMENTAL DE ENGENHARIA ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES

RAMO DE TELECOMUNICAÇÕES

CODIFICADOR JPEG BASEADO EM FPGA

ANDRÉ MIGUEL DE SOUSA BRILHANTE
(Licenciado)

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA DE ELECTRÓNICA E TELECOMUNICAÇÕES

Constituição do Júri:

Presidente:	Prof. Mário Pereira Vestias
Vogal (arguente):	Prof. José Manuel Peixoto do Nascimento
Vogal (orientador):	Prof. Artur Jorge Ferreira
Vogal (orientador):	Prof. Tiago Miguel Braga da Silva Dias

MAIO DE 2012

Resumo

O presente trabalho consiste na implementação em hardware de unidades funcionais dedicadas e optimizadas, para a realização das operações de codificação e descodificação, definidas na norma de codificação com perda *Joint Photographic Experts Group* (JPEG), ITU-T T.81 ISO/IEC 10918-1.

Realiza-se um estudo sobre esta norma de forma a caracterizar os seus principais blocos funcionais. A finalidade deste estudo foca-se na pesquisa e na proposta de optimizações, de forma a minimizar o hardware necessário para a realização de cada bloco, de modo a que o sistema realizado obtenha taxas de compressão elevadas, minimizando a distorção obtida. A redução de hardware de cada sistema, codificador e descodificador, é conseguida à custa da manipulação das equações dos blocos *Forward Discrete Cosine Transform* (FDCT) e Quantificação (Q) e dos blocos *Forward Discrete Cosine Transform* (IDCT) e Quantificação Inversa (IQ).

Com as conclusões retiradas do estudo e através da análise de estruturas conhecidas, descreveu-se cada bloco em *Very-High-Speed Integrated Circuits* (VHSIC) *Hardware Description Language* (VHDL) e fez-se a sua síntese em *Field Programmable Gate Array* (FPGA). Cada sistema implementado recorre à execução de cada bloco em paralelo de forma a optimizar a codificação/descodificação. Assim, para o sistema codificador, será realizada a operação da FDCT e Quantificação sobre duas matrizes diferentes e em simultâneo. O mesmo sucede para o sistema descodificador, composto pelos blocos Quantificação Inversa e IDCT. A validação de cada bloco sintetizado é executada com recurso a vectores de teste obtidos através do estudo efectuado.

Após a integração de cada bloco, verificou-se que, para imagens *greyscale* de referência com resolução de 256 linhas por 256 colunas, é necessário 820,5 μs para a codificação de uma imagem e 830,5 μs para a descodificação da mesma. Considerando uma frequência de trabalho de 100 MHz, processam-se aproximadamente 1200 imagens por segundo.

Palavras-chave: processamento digital de imagem, JPEG, codificação com perda, DCT, VHDL, FPGA.

Abstract

This thesis addresses the implementation of dedicated and optimized hardware functional units for the encoding and decoding operations defined in the ITU-T T.81 ISO / IEC 10918 -1 image coding standard, proposed by the Joint Photographic Experts Group (JPEG).

In this scope, a study of the JPEG standard is conducted to characterize its main functional blocks involved in the implementation of a lossy image encoder/decoder (codec). This study aims to investigate and to propose optimizations to the algorithms and operations involved in each block of the image codec, in order to minimize the hardware resources required for its realization and still guaranteeing high compression rates and minimum distortion. The reduction in the amount of hardware resources required to implement the codec is achieved with the proposal of several optimizations to the computation of the Forward Discrete Cosine Transform (FDCT) and the Quantization (Q) blocks of the image encoder, as well as to the Inverse Discrete Cosine Transform (IDCT) and the Inverse Quantization (IQ) blocks of the image decoder.

As a result of this study, dedicated architectures for the computation of the FDCT, IDCT, Q and IQ operations are proposed. Moreover, such functional units were also combined to build the encoding and decoding loops of the JPEG codec, in which the transform and quantization operations are computed in parallel to maximize the system performance. All these circuits were described in Very-High-Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) and synthesized in a Field Programmable Gate Array (FPGA) device for proof of concept and performance assessment. The functional validation of each of these blocks was conducted using test vectors obtained from the preliminary theoretical study.

The proposed implementation results showed that the proposed JPEG encoding and decoding systems require 820.5 μs to encode a reference grayscale image with a resolution of 256 lines by 256 columns and 830.5 μs to decode it. This allows to process 1200 images per second, operating with a clock frequency of 100 MHz.

Keywords: Digital image processing, JPEG, lossy encoding, DCT, VHDL, FPGA.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

BMP	*BitMaP*
CLB	*Configuration Logical Blocks*
CODEC	*COder/DECoder*
DCT	*Discrete Cosine Transform*
DSP	*Digital Signal Processor*
EOB	*End Of Block*
FDCT	*Forward Discrete Cosine Transform*
FFT	*Fast Fourier Transform*
FPGA	*Field Programmable Gate Array*
GIF	*Graphics Interchange Format*
IDCT	*Inverse Discrete Cosine Transform*
IOB	*Input/Output Block*
JPEG	*Joint Photographic Experts Group*
LUT	*LookUp Table*
MAC	*Multiply-Accumulate*
MPEG	*Moving Picture Experts Group*
MP3	*MPEG-1/2 Audio Layer 3*
PDA	*Personal Digital Assistant*
PNG	*Portable Network Graphics*
RLE	*Run-Length Encoding*
ROM	*Read-Only Memory*
SNR	*Signal to Noise Ratio*
SVH	*Sistema Visual Humano*
TIFF	*Tagged Image File Format*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very-High-Speed Integrated Circuits*
VLC	*Variable Length Coding*

1. Introdução

Num mundo digital em exponencial crescimento, o uso de técnicas de compressão e descompressão de informação digital é cada vez mais comum, devido à transmissão de dados com uma largura de banda finita e/ou armazenamento de ficheiros num espaço limitado. Estas técnicas reduzem significativamente o volume de dados a armazenar e/ou transmitir; por exemplo, conseguem-se reduções de 10 a 50 vezes no espaço ocupado por uma imagem [1], através da sua codificação, sem que daí resulte perda perceptível pelo Sistema Visual Humano[1] (SVH).

CODECs[2] como o Joint Photographic Experts Group (JPEG) [1], para imagem, Moving Picture Experts Group 4 [21] (MPEG4), para vídeo ou MPEG-1/2 Audio Layer 3 [22] (MP3) para áudio, conduzem a armazenamento e transmissões mais eficientes sem perturbações aparentes na qualidade da imagem, vídeo e som, respectivamente.

Neste contexto, surge a codificação de imagem com perda (*lossy encoding*) que recorre às propriedades do SVH para reduzir significativamente o volume de dados associado à imagem codificada. Devido às características fisiológicas do SVH, muita da informação armazenada no momento de aquisição da imagem pode ser removida, sem que daí resulte prejuízo na qualidade aparente da imagem. A utilização de codificação com perda justifica-se pelo facto de que o SVH não tem noção de todas as transições abruptas de luminosidade e cor [2], sendo menos sensível a transições de cor do que a transições de luminosidade. Esta característica fisiológica leva a que se possa tirar partido da codificação de imagem com perda, através da redução de amostras de cor a processar. O SVH é também menos sensível à informação contida nas componentes de alta frequência do que nas componentes de baixa frequência, sendo possível tirar partido desta característica.

No cenário de manipulação de CODECs multimédia, recorre-se a processadores especializados para a codificação de imagem com perda, como por exemplo: *Digital Signal Processor* (DSP) e *Field Programmable Gate Array* (FPGA) [3]. Esta escolha prende-se pela eficácia que estes dispositivos apresentam. O DSP consiste num processador cuja arquitectura se encontra optimizada para as operações comuns em processamento digital de sinal em tempo real, tais como a multiplicação e adição, filtragem, *Fast Fourier Transform* (FFT), entre outros algoritmos baseados em frequência, entre outras [35]. As FPGA apresentam a vantagem de serem reconfiguráveis e de possibilitarem tempos de processamento reduzidos, podendo ser adaptadas de acordo com a função a desempenhar, através da utilização de módulos de hardware sintetizado. Esta propriedade tem vindo a assumir maior importância na área da codificação de imagem e vídeo por possibilitar a construção de sistemas mais versáteis, pois desta forma é possível implementar num mesmo sistema, e em instantes temporais distintos, vários circuitos de codificação e descodificação de imagem e/ou vídeo.

[1] O processo de recolha e interpretação por parte do SVH é vulgarmente conhecido por percepção visual.
[2] O termo CODEC resulta da contracção das palavras anglo-saxónicas COder / DECoder.

1.1. Estado da arte

A fotografia alterou a forma de documentar determinados eventos. Desde o seu aparecimento, em meados do século XIX, a sua evolução tem sido constante. Com o aparecimento da era digital começaram a surgir, no início de 1990, as primeiras máquinas fotográficas digitais [7]. Com o progresso da tecnologia, o custo por equipamento começou a reduzir, massificando-se a utilização da fotografia digital. Nos dias de hoje são comuns as máquinas fotográficas digitais com resolução 14 Megapixels, por exemplo. Resoluções desta magnitude traduzem-se num desafio em relação ao espaço ocupado por cada imagem.

Com o aparecimento e proliferação de dispositivos portáteis, tais como *Personal Digital Assistant* (PDA), telemóveis ou *laptops*, o armazenamento e/ou transmissão de fotografias aumentou. No entanto, dispositivos desta natureza, apresentam baixa capacidade de armazenamento (face a dispositivos não portáteis) e estão dependentes de uma bateria. Estas características limitam a complexidade dos sistemas que permitem visualizar, armazenar, enviar e receber imagens.

Tal como a fotografia, também o vídeo tem beneficiado da evolução tecnológica. O vídeo pode ser considerado como uma série de fotografias, alinhadas sequencialmente, em instantes de tempo consecutivos. Com a introdução do vídeo digital, o problema relativo ao espaço ocupado por cada imagem assume uma dimensão acrescida. Um outro problema coloca-se ainda face à transmissão dessas mesmas imagens consecutivas: a limitação da largura de banda.

Com a utilização do CODEC JPEG [1] em sistemas tais como os descritos anteriormente, reduz-se consideravelmente o tamanho que cada imagem ocupa sem que a sua qualidade seja gravemente prejudicada. Com a redução da informação, consegue-se reduzir a largura de banda ocupada e necessária para a transmissão. Apesar desta norma apresentar alguma complexidade ao requerer o uso de transformadas no processamento de imagem, existem algoritmos que reduzem essa complexidade [23-26]. Com esses algoritmos é possível definir estruturas para um rápido processamento utilizando poucos recursos, o que se traduz num baixo consumo energético do sistema [27-29].

Obtêm-se tempos de processamento reduzidos através da implementação dessas estruturas em sistemas de hardware configurável ou dedicado. As FPGA, por exemplo, disponibilizam milhares de portas lógicas [17], permitindo a realização de sistemas cada vez mais complexos, com um custo associado cada vez menor. As aplicações variam desde aplicações militares [18], passando pelas telecomunicações [19] ou multimédia [20], fazendo com que este circuito integrado reconfigurável tenha assumido um papel relevante no mercado dos sistemas digitais.

1.2. Objectivos do trabalho

Com a crescente utilização de dispositivos portáteis que requerem maior processamento, bem como rapidez, este trabalho aborda a implementação de um sistema CODEC, da norma JPEG [1] para a codificação de imagem com perda. A norma foca-se em optimizar a

codificação e compressão de uma imagem de carácter natural (fotografias de cenas da natureza e da vida) [4].

A entrada do CODEC tem as amostras constituintes da imagem, em formato de luminância e duas crominâncias YCrCb [4]. Estas podem ser obtidas directamente, por exemplo através de uma câmara digital, ou convertidos de formatos standard, tais como *Bitmap* (BMP) [4], *Tagged Image File Format* (TIFF) [5] ou mesmo em formato binário *raw data*[3]. À saída do codificador estão presentes os coeficientes quantificados da transformada. Por sua vez, o descodificador tem à entrada os coeficientes quantificados da transformada e à saída as amostras, em formato YCrCb. Ambos os sistemas são implementados numa FPGA. Segundo a função a desempenhar, a imagem será codificada ou descodificada, respectivamente.

Como a implementação do CODEC numa FPGA consiste num sistema digital, torna-se necessário dimensionar a arquitectura para usar os recursos de hardware estritamente indispensáveis à realização das operações pretendidas. Para tal foi necessário saber quantos bits são utilizados em cada etapa do CODEC e qual a truncatura a aplicar a cada uma das operações. Para realizar essa verificação foi utilizada a ferramenta *MATrix LABoratory* (MATLAB) [11] para o estudo do tamanho dos símbolos a utilizar em cada etapa e análise da complexidade inerente a cada bloco.

No final, o protótipo a desenvolver neste projecto tem, entre outras, as seguintes aplicações:

- Codificação/descodificação de fotografias digitais [4,6];
- Utilização como co-processador dedicado ou acelerador (tratando-se de hardware dedicado que é inserido num dispositivo digital para minimizar o tempo de codificação/descodificação de imagens);
- Conversão e descarregamento de imagens, com recurso a um sistema externo, inseridas em páginas *Web* [8] de e para outros formatos (tais como *Graphics Interchange Format*, conhecido por GIF [4,9], e *Portable Network Graphics*, conhecido por PNG [4,9,10]).

1.3. CODEC JPEG

A arquitectura de um CODEC é constituída por dois elementos: Codificador e Descodificador. O Codificador tem como objectivo codificar através de uma representação eficiente de forma a comprimir/compactar a quantidade de dados a transferir e/ou armazenar. Por sua vez, o Descodificador, procura reproduzir na sua saída a imagem original, executando o processo inverso realizado no Codificador. Estes dois processos serão apresentados com maior detalhe no capítulo 2. Na Figura 1 representa-se o modelo referido anteriormente.

[3] *Raw data* consiste em dados que ainda não foram manipulados e/ou processados.

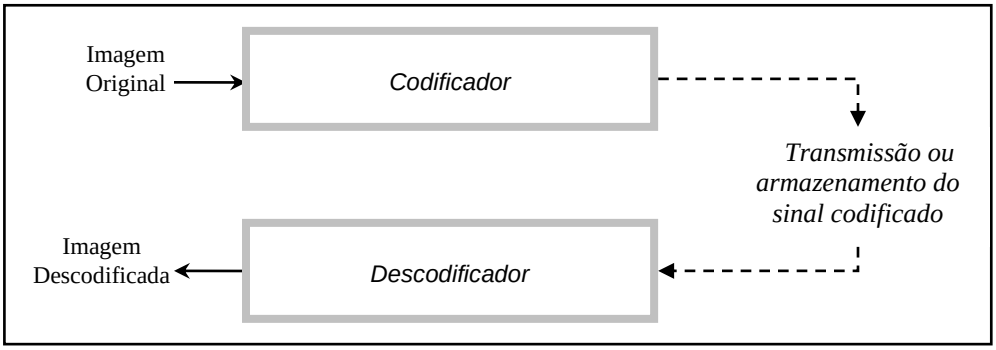


Figura 1 – Modelo de um codificador e descodificador de imagem.

O principal objectivo de um CODEC foca-se assim na compressão e descompressão de dados digitais. Um CODEC assenta numa de duas categorias: codificação sem perda (*lossless*) ou codificação com perda (*lossy*). A primeira comprime (ou compacta) sem que exista perda de informação no processo de codificação, enquanto que na segunda a perda está sempre presente, podendo no entanto ser ajustada, de acordo com as características da aplicação. O CODEC JPEG realiza operações que se encontram inseridas em ambas as categorias, ou seja, suporta os modos de codificação com e sem perda. Neste trabalho, considera-se apenas a versão de codificação com perda

Não existe um CODEC que seja ideal para qualquer tipo de aplicação ou para qualquer tipo de imagem. Imagens naturais (fotografias) constituem o tipo de imagens onde o CODEC JPEG é geralmente aplicado e para as quais foi projectado. Para outro tipo de imagens, por exemplo aquelas geradas computacionalmente, existem outros CODECs que são mais eficientes face ao JPEG, tais como GIF, PNG, entre outros.

As imagens a serem codificadas necessitam de estar no formato YCbCr. A componente Y, ou luminância, representa a intensidade da imagem. As componentes Cb e Cr, denominadas também por crominância, especificam os tons de azul e vermelho, respectivamente, numa imagem. É possível converter entre YCbCr e outros espaços de cor. Por exemplo, o formato RGB (formato vulgarmente utilizado em sistema que capturam imagens [4]) pode ser convertido de e para YCbCr através de um sistema de equações ou de uma multiplicação matricial [4].

Devido à diversidade de aplicações onde o CODEC pode ser implementado, a norma JPEG define quatro modos de operação, ou seja, modos de transmissão e visualização da imagem [30]:

- Hierárquica;
- Progressiva;
- Sequencial;
- Sem perda.

No primeiro método a imagem é codificada com diversas resoluções, de modo a que as resoluções mais baixas sejam as primeiras a serem exibidas mesmo antes de descodificar a imagem final. No segundo, a imagem é codificada através de múltiplos varrimentos às linhas e colunas, começando por apresentar uma imagem final com qualidade reduzida e à

medida que se realizam os diversos varrimentos, a qualidade da imagem vai melhorando até um resultado semelhante ao do primeiro método. No terceiro, codifica-se a imagem sequencialmente, da esquerda para a direita, de cima para baixo. No quarto e último método, o único em que não existe perda no processo de codificação, preserva-se sempre a imagem original. Como consequência, a arquitectura deste CODEC difere face aos restantes três métodos e as taxas de compressão obtidas são bastante baixas face aos outros métodos de compressão. Na realização do presente trabalho apenas será considerado o modo sequencial.

A perda neste tipo de CODEC (*lossy*) pode ser ajustada, existindo sempre um compromisso entre o tamanho da imagem a codificar (*bytes*) face à qualidade da mesma. Utilizando o CODEC, as taxas de compressão de uma imagem podem ser bastante significativas, sem que a qualidade da imagem seja gravemente penalizada. Um exemplo da utilização do CODEC JPEG encontra-se ilustrado na Figura 2. Através desta constata-se que para a mesma imagem, sem e com codificação JPEG, a qualidade visual pouco difere. No entanto, em termos de dimensão do ficheiro correspondente, verifica-se redução acima de 80% da dimensão do ficheiro JPEG comparativamente com o original.

Original
71 115 bytes

JPEG
11 732 bytes

Figura 2 – Imagem original codificada sem perda (esquerda), constituida por 256 linhas e 256 colunas e 8 bits/pixel, e codificada com perda com JPEG (direita), constituida por 256 linhas e 256 colunas e 1,43 bits/pixel.

1.4. Organização do documento

O presente trabalho trata a implementação de um CODEC JPEG optimizado para codificação com perda, utilizando a transmissão com modo de operação sequencial. A implementação deste encontra-se dividida em três partes distintas:

- Análise dos blocos do codificador e descodificador;
- Estudo/simulação de cada bloco;
- Implementação.

No segundo capítulo é abordado o CODEC JPEG analisando-se individualmente cada um dos seus blocos. Realiza-se uma descrição geral da norma. São estudadas as características da *Forward Discrete Cosine Transform* (FDCT) e *Inverse Discrete Cosine Transform* (IDCT), apresentadas as tabelas de Quantificação e Entropia a utilizar e é descrito um exemplo teórico da aplicação do CODEC numa matriz.

No terceiro capítulo são apresentadas as simulações resultantes das optimizações aplicadas no segundo capítulo e as variantes para cada bloco constituinte do CODEC JPEG. Assim, exploram-se algumas características subjacentes a cada bloco, bem como o número de bits utilizados na codificação e representação dos diferentes elementos envolvidos nos cálculos realizados internamente em cada bloco.

O quarto capítulo consiste na implementação do CODEC. Exploram-se as alternativas através das conclusões obtidas a partir do estudo descrito no terceiro capítulo. São também analisadas diferentes opções para se obter uma rápida execução da codificação/descodificação de imagens. Apresentam-se resultados de codificação e descodificação de imagens de teste de referência.

No quinto capítulo apresentam-se as principais conclusões do trabalho. Indicam-se também um conjunto de melhoramentos a realizar à estrutura introduzida no quarto capítulo que podem ser realizados como trabalho futuro.

2. Análise do CODEC JPEG

Na secção 1.3 foi introduzido o CODEC JPEG e apresentado o tipo de aplicações no qual este é utilizado. Foram também apresentados os quatro modos de operação suportados. Dos quatros métodos apresentados, este trabalho focar-se-á apenas no método sequencial, da codificação com perda (*lossy encoding*). Apesar da escolha recair sobre este método, a arquitectura do CODEC JPEG (com perda) não difere entre os restantes modos de operação. Assim, os blocos constituintes do CODEC JPEG encontram-se representados na Figura 3.

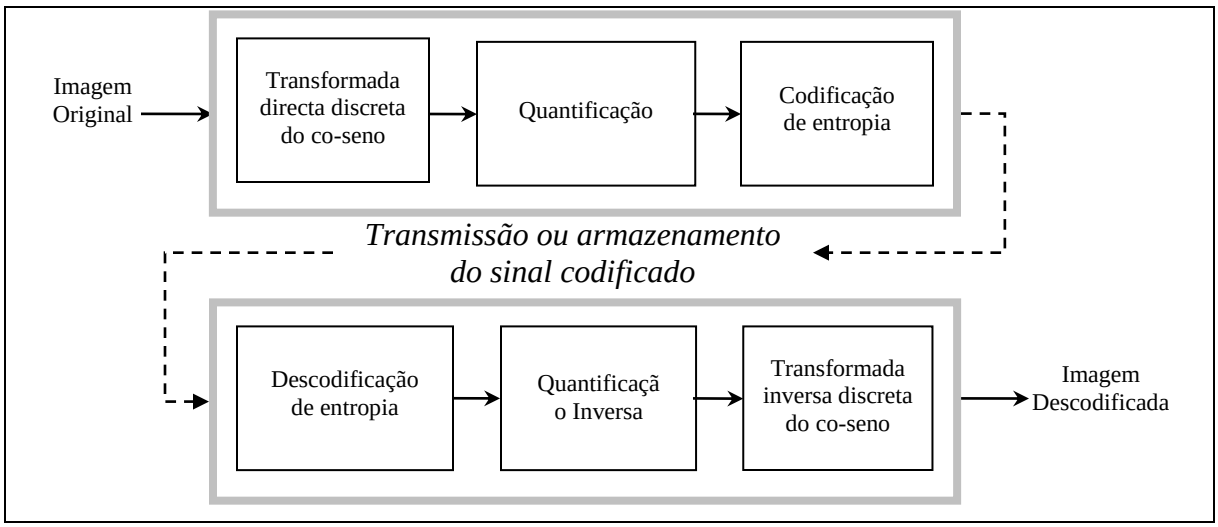


Figura 3 – Diagrama de blocos da codificação e descodificação JPEG. Imagem original, imagem codificada com perda e descodificada.

O processo de codificação encontra-se dividido em três fases, tal como se verifica pela Figura 3:

- A transformada directa discreta do co-seno (FDCT - *Forward Discrete Cosine Transform*);
- Quantificação;
- Codificação de entropia.

Numa primeira fase, as amostras de entrada são transformadas através da FDCT, a qual obtém uma representação alternativa, e mais conveniente, do sinal a codificar. As suas características serão apresentadas na secção **Erro! A origem da referência não foi encontrada.**.

O codificador JPEG introduz perda de modo controlado. Esta introdução de perda é realizada pelo bloco da Quantificação[4]. Devido às limitações do SVH, é possível descartar dados que não são relevantes para a percepção humana da imagem, resultando numa redução de dados a transferir e/ou armazenar.

[4] Ignorando os erros de precisão numérica impostos por uma implementação digital.

Por último, os dados são compactados recorredo a codificação de entropia. Deste tipo de codificação não resulta qualquer perda (*lossless*) uma vez que é realizada uma compactação, com base estatística, sobre os símbolos gerados do bloco anterior. Na aplicação do processo inverso resulta a descodificação da imagem. Esta é, geralmente, semelhante à imagem de entrada, apesar da perda introduzida na quantificação.

As amostras de entrada são codificadas tipicamente com 8 bits [4] representados no intervalo inteiro de 0 a 255. Apesar da norma definir que as amostras podem ser codificadas com 8 ou 12 bits por *pixel*, neste trabalho a escolha recai sobre a primeira devido à sua utilização ser mais comum [4]. Cada imagem é decomposta e processada em matrizes quadradas compostas pelas amostras, com 8 linhas e 8 colunas, resultando num total de 64 *pixels* [1].

Antes da aplicação da FDCT, é executado um ajustamento no nível médio do sinal, subtraindo cada elemento da matriz por 128 (para pixels representados com 8 bits). Desta forma, as amostras apresentam o formato de complemento para dois, resultando na alteração do intervalo inteiro de representação (de -128 a 127). Esta operação leva a que o nível médio da energia da imagem seja inferior, reduzindo assim a amplitude do primeiro coeficiente da matriz transformada (ver secção **Erro! A origem da referência não foi encontrada.**). Como consequência, após a IDCT, é necessário somar 128 de forma a repor o nível médio (componente DC) da matriz de amostras.

A componente da luminância assume um papel mais importante na imagem face às componentes de crominância [4]. Com base nesta característica reduz-se a amostragem nas componentes de cor sem que daí resulte numa perturbação acentuada na qualidade da imagem, reduzindo assim a quantidade de informação a processar. O esquema de amostragem mais comum utilizado em JPEG é o 4:2:0 [34], i.e., para cada varrimento horizontal ou vertical, são recolhidas quatro amostras de Y e duas amostras de Cr e Cb (sub-amostragem de crominância), tal como se pode visualizar na Figura 4.

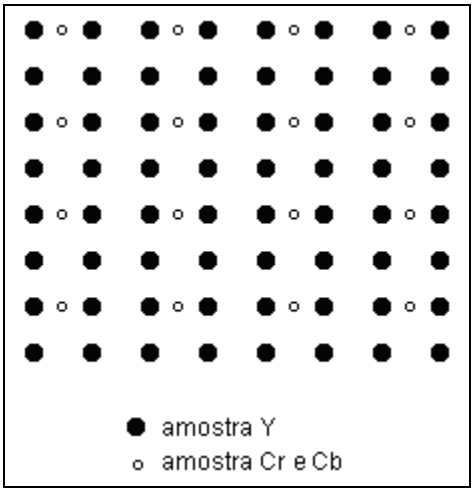


Figura 4 – Esquema de amostragem do CODEC JPEG.

Nas secções seguintes é realizada uma análise detalhada de cada bloco, e verificadas as suas propriedades mais relevantes.

2.1. Forward Discrete Cosine Transform *e sua inversa*

Na codificação da imagem, a FDCT, que recorre à transformada DCT-II [4], é aplicada sobre uma matriz quadrada. No caso da norma JPEG, a matriz é constituída por 8 linhas e 8 colunas, resultando num total de 64 elementos. Após a aplicação da transformada obtém-se a matriz de coeficientes. O número de elementos da matriz foi obtido através do compromisso que existe entre a complexidade do cálculo da FDCT e a representação espacial das frequências [34]. Um número superior a 8 elementos por linha/coluna resulta numa representação mais discretizada nas frequências, porém, existem algoritmos rápidos e com um baixo custo de implementação para 8 elementos.

Os valores dos coeficientes da FDCT a duas dimensões (2D) traduzem a importância visual que as várias frequências espaciais têm na composição do sinal de entrada de 64 amostras. Esta matriz resulta das respectivas funções de base co-seno com frequência crescente em linha e coluna em que o canto superior esquerdo corresponde às baixas frequências enquanto que o canto inferior direito corresponde às altas frequências. O primeiro coeficiente da matriz corresponde à frequência zero em ambas as dimensões e é denominado por coeficiente DC. Os restantes 63 coeficientes reflectem a variação crescente das frequências a duas dimensões, nas linhas e colunas, pelo que são denominados de coeficientes AC.

A escolha da FDCT resulta da conjugação das seguintes propriedades:

- Concentra grande parte da energia do sinal nas frequências baixas [12,13];
- É ortogonal, o que reduz a sua complexidade computacional [12];
- Existem algoritmos rápidos para a sua realização [14,15];
- Tem elevada capacidade de descorrelação estatística entre os coeficientes [12,13].

A equação da FDCT é dada por

$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s_{xy} \cos\left(\frac{(2x+1)u\pi}{2N} \right) \cos\left(\frac{(2y+1)v\pi}{2N} \right), \; u, v \in [0, N-1] \qquad \textbf{(1)}$$

em que $C_u, C_v = 1/\sqrt{2}$ quando $u, v = 0$, $C_u, C_v = 1$ nos restantes casos e $N = 8$.

A transformada inversa da FDCT (IDCT) é usada no processo de descodificação da imagem, sendo que a sua equação é dada por

$$s_{xy} = \frac{1}{4} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_u C_v S_{uv} \cos\left(\frac{(2x+1)u\pi}{2N} \right) \cos\left(\frac{(2y+1)v\pi}{2N} \right), \; x, y \in [0, N-1] \qquad \textbf{(2)}$$

em que $C_u, C_v = 1/\sqrt{2}$ quando $u, v = 0$, $C_u, C_v = 1$ nos restantes casos e $N = 8$.

Num cenário ideal, o cálculo sequencial da FDCT e IDCT resultaria numa operação sem perdas, i.e. a matriz de entrada seria exactamente igual à matriz de saída. Contudo, devido à restrição numérica imposta pelo cálculo digital (utilização de processadores com limitação numérica), existe sempre um erro associado a esta operação, apesar de ser limitado (quanto mais bits se utilizarem no cálculo dos coeficientes das transformadas, menor será o erro).

Como se verifica através de (1) e (2), o cálculo da FDCT é realizado fundamentalmente com recurso a operações de multiplicação e soma. As operações de multiplicação são as que se revelam mais exigentes (que pode ser traduzido num aumento do tempo dispendido para o cálculo desta) e, como tal, é necessário reduzir o número deste tipo de operações de forma a diminuir a complexidade total do sistema.

2.1.1. Optimizações sobre a FDCT

Retomando a análise de (1), e antes de realizar qualquer optimização, calcula-se o número de somas e multiplicações[5] que são necessárias no cálculo dos coeficientes da FDCT. O número total de somas é reduzido (12288), quando comparado com o número de multiplicações (49408). No entanto, verifica-se que existem optimizações que podem ser consideradas para a realização do cálculo da FDCT. Por exemplo, constantes como $\pi/2N$ ou $C_u C_v/4$ podem ser calculadas *à priori*, devido a imposições da norma e à sua gama restrita de valores.

Devido à presença destas constantes, reduzi-se o número de multiplicações através do cálculo *à priori* das mesmas. As constantes dependentes de C_u e C_v, acima referidas, tomam um conjunto de valores limitados (ver Tabela 1). Com o cálculo *à priori* destas e da constante $\pi/2N$, o número de multiplicações diminui para 32832. Por sua vez, o número de somas mantém-se.

C_v \ C_u	0	{1,.., 7}
0	0,1250	0,1768
{1,.., 7}	0,1768	0,2500

Tabela 1 – Valores do produto de C_u, C_v e 0,25, com 4 casas decimais.

Analisando individualmente a componente dos co-senos, verifica-se também que existe um número limitado de elementos a considerar. Através de (1) constata-se que os valores de *x*, *y*, *u* e *v* estão contidos num domínio de 0 a 7, inclusivamente.

Calculando apenas os argumentos de cada um dos co-senos verifica-se quais os valores que pertencem a um conjunto limitado de elementos. Esses valores encontram-se representados na Tabela 2.

Como os valores presentes na tabela anterior correspondem aos argumentos do co-seno e, sendo o co-seno uma função periódica em $2k\pi$ (sendo *k* um inteiro), calcula-se *à priori* o valor dos co-senos correspondentes a cada par *x*/*u* ou *y*/*v*, necessários para o cálculo da FDCT de uma matriz de 8x8 amostras. Esses valores encontram-se representados, com precisão de 3 casas decimais, na Tabela 3. Aplicando esta optimização no cálculo da FDCT reduz-se não só o número de somas, para 4096, bem como o número de multiplicações, para 8256.

[5] Considera-se que a divisão tem peso computacional equivalente ao de uma multiplicação.

x \ u	0	1	2	3	4	5	6	7
0	0	π/16	2π/16	3π/16	4π/16	5π/16	6π/16	7π/16
1	0	3π/16	6π/16	9π/16	12π/16	15π/16	18π/16	21π/16
2	0	5π/16	10π/16	15π/16	20π/16	25π/16	30π/16	35π/16
3	0	7π/16	14π/16	21π/16	28π/16	35π/16	42π/16	49π/16
4	0	9π/16	18π/16	27π/16	36π/16	45π/16	54π/16	63π/16
5	0	11π/16	22π/16	33π/16	44π/16	55π/16	66π/16	77π/16
6	0	13π/16	26 /16	39π/16	52π/16	65π/16	78π/16	91π/16
7	0	15π/16	30π/16	45π/16	60π/16	75π/16	90π/16	105π/16

Tabela 2 – Valores dos argumentos dos co-senos.

x \ u	0	1	2	3	4	5	6	7
0	1	0,981	0,924	0,831	0,707	0,556	0,383	0,195
1	1	0,831	0,383	-0,195	-0,707	-0,981	-0,924	-0,556
2	1	0,556	-0,383	-0,981	-0,707	0,195	0,924	0,831
3	1	0,195	-0,924	-0,556	0,707	0,831	-0,383	-0,981
4	1	-0,195	-0,924	0,556	0,707	-0,831	-0,383	0,981
5	1	-0,556	-0,383	0,981	-0,707	-0,195	0,924	-0,831
6	1	-0,831	0,383	0,195	-0,707	0,981	-0,924	0,556
7	1	-0,981	0,924	-0,831	0,707	-0,556	0,383	-0,195

Tabela 3 – Valores dos co-senos, com representação de 3 casas decimais.

Através desta última optimização, verifica-se que, para os elementos nulos de *u* e *v*, o co-seno apresenta valor unitário. Assim, existem elementos para os quais não é necessária a realização da operação multiplicação.

No caso em que $u = v = 0$, o primeiro elemento da matriz dos coeficientes, o seu cálculo é conseguido exclusivamente à base de somas. Para os coeficientes em que apenas um destes elementos (*u* ou *v*) é nulo, existirá a necessidade de aplicar uma única vez a multiplicação. Para os restantes casos não existe qualquer diferença face ao que tem sido apresentado. Assim, o cálculo da FDCT divide-se em 4 operações distintas, reduzindo o número total de multiplicações para 7232:

$$S_{vu} = \begin{cases} \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx}, u = v = 0 \\ \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16}, v = 0 \\ \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2y+1)v\pi}{16}, u = 0 \\ \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}, u, v \in [1..7] \end{cases} \quad \textbf{(3)}$$

Sendo a FDCT uma transformada separável [12], o cálculo desta pode ser obtido através da aplicação de duas DCT a uma dimensão (DCT-I). A equação da DCT-I é dada por

$$S_k = A_k \sum_{n=0}^{N-1} s_n \cos\left(\frac{(2n+1)k\pi}{2N}\right), k \in [0, N-1] \quad \textbf{(4)}$$

em que $A_k = 1/\sqrt{N}$ quando $k = 0$, $A_k = \sqrt{2/N}$ nos restantes casos e $N = 8$ para este caso particular.

Aplicando a propriedade

$$\sum_a \sum_b n_a n_b = \sum_a n_a \sum_b n_b \quad \textbf{(5)}$$

e considerando que $n_a = \sum_{x=0}^{7} \cos\left(\frac{(2x+1)u\pi}{2N}\right)$ e $n_b = \sum_{y=0}^{7} \cos\left(\frac{(2y+1)v\pi}{2N}\right)$, verifica-se que, aplicando (5) em (1) e posteriormente a propriedade descrita em (4), obtém-se

$$S'_{vu} = \sum_{x=0}^{7} \cos\left(\frac{(2x+1)u\pi}{2N}\right) \sum_{y=0}^{7} s_{xy} \cos\left(\frac{(2y+1)v\pi}{2N}\right), \; u, v \in [0, 7] \quad \textbf{(6)}$$

A igualdade entre constantes pode ser obtida através de

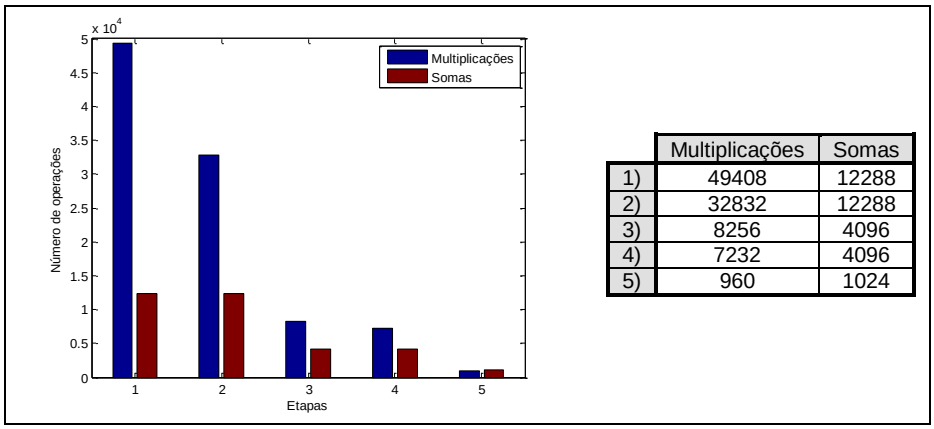
$$A_k \times A_k = C_u \times C_v \times \frac{1}{4} \quad \textbf{(7)}$$

Tendo em conta (6) e (7), obtém-se

$$\begin{aligned} S_{vu} &= \frac{1}{4} C_u C_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s_{xy} \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \\ &= \frac{1}{4} C_u C_v \sum_{x=0}^{7} \cos\left(\frac{(2x+1)u\pi}{2N}\right) \sum_{y=0}^{7} s_{xy} \cos\left(\frac{(2y+1)v\pi}{2N}\right), u, v \in [0, N-1] \end{aligned} \quad \textbf{(8)}$$

Esta optimização consiste na aplicação da DCT-I às linhas, uma por linha, e posteriormente às colunas, uma por coluna, obtendo-se assim os coeficientes da FDCT, tal como demonstrado em (8). Consequentemente, calculando todos os valores de *y* em função de um dado *x*, ou vice-versa, reduz-se o número de multiplicações a realizar para 960 e o número de somas para 1024.

Com as técnicas analisadas conclui-se que o número de operações no cálculo da FDCT pode ser drasticamente reduzido, evitando assim a utilização desnecessária de circuitos complexos e aumentando a rapidez de cálculo. Na Figura 5 ilustram-se os ganhos obtidos em termos de complexidade do cálculo da FDCT com a aplicação das técnicas descritas anteriormente.



	Multiplicações	Somas
1)	49408	12288
2)	32832	12288
3)	8256	4096
4)	7232	4096
5)	960	1024

Figura 5 – Representação das operações das várias optimizações utilizadas na FDCT (1) – FDCT (sem optimizações); 2) – Cálculo das constantes; 3) – Resolução dos co-senos; 4) – Casos particulares; 5) – Separabilidade da FDCT).

2.1.2. Optimizações sobre a IDCT

As optimizações possíveis de realizar no processo de cálculo da IDCT são semelhantes às realizadas para a FDCT. É possível calcular previamente as constantes, resolver os co-senos e aplicar a separabilidade da transformada. Porém o número de multiplicações numa fase inicial será sempre superior (57472) face ao número de multiplicações da FDCT (49408). Por sua vez, o número de somas mantém-se. Tal facto deve-se à dependência das constantes C_u e C_v com o coeficiente S_{uv}, tal como se verifica em (2).

Para o cálculo das constantes da IDCT adopta-se as mesmas condições assumidas na secção 2.1.1., ou seja, o cálculo *à priori* de constantes como $\pi/2N$ ou $C_u C_v/4$. No entanto, para que o mesmo seja verdadeiro, é necessário aplicar a relação

$$c\sum_a\sum_b n_a n_b = \sum_a\sum_b cn_a n_b \qquad \textbf{(9)}$$

Com a aplicação desta propriedade, reduzi-se o número de multiplicações para 36864. O número de somas mantém-se.

A resolução dos co-senos na IDCT é idêntica à resolução dos co-senos da FDCT (ver secção 2.1.1). Assim, e não sendo necessária qualquer alteração aos valores da Tabela 3, verifica-se uma redução de multiplicações e somas para 12288 e 4096, respectivamente.

Apesar de numa primeira análise não ser possível identificar casos particulares, à semelhança do que foi realizado para a FCDT, estes existem caso seja calculada uma matriz intermédia *à priori*, resultante do produto entre as constantes C_u e C_v e os coeficientes S_{uv}. Considerando que

$$S'_{uv} = \frac{1}{4} C_u C_v S_{uv} \qquad \textbf{(10)}$$

constata-se que o cálculo da IDCT divide-se em 4 operações distintas, reduzindo o número de multiplicações para 7232:

$$s_{xy} = \begin{cases} \sum_{u=0}^{7}\sum_{v=0}^{7} S'_{uv}, x = y = 0 \\ \sum_{u=0}^{7}\sum_{v=0}^{7} S'_{uv} \cos\frac{(x+1)u\pi}{16}, y = 0 \\ \sum_{u=0}^{7}\sum_{v=0}^{7} S'_{uv} \cos\frac{(y+1)v\pi}{16}, x = 0 \\ \sum_{u=0}^{7}\sum_{v=0}^{7} S'_{uv} \cos\frac{(x+1)u\pi}{16}\cos\frac{(y+1)v\pi}{16}, x, y \in [1,7] \end{cases} \quad \textbf{(11)}$$

Sendo as optimizações da IDCT bastante semelhantes às optimizações da FDCT também se verifica que a IDCT pode ser obtida através de duas IDCT-I. Esta última é dada por

$$s_n = \sum_{k=0}^{N-1} A_k S_k \cos\left(\frac{(n+1)k\pi}{2N}\right), n \in [0, N-1] \quad \textbf{(12)}$$

em que $A_k = 1/\sqrt{N}$ quando $k = 0$, $A_k = \sqrt{2/N}$ nos restantes casos e $N = 8$ para este caso particular.

Aplicando (12) e a propriedade descrita em (5) a (2), e considerando que $n_a = \sum_{u=0}^{7} C_u \cos\left(\frac{(x+1)u\pi}{2N}\right)$ e $n_b = \sum_{v=0}^{7} C_v \cos\left(\frac{(y+1)v\pi}{2N}\right)$, obtém-se

$$s'_{xy} = \sum_{u=0}^{7} C_u \cos\left(\frac{(x+1)u\pi}{2N}\right)\sum_{v=0}^{7} C_v s_{xy} \cos\left(\frac{(y+1)v\pi}{2N}\right), \; x, y \in [0,7] \quad \textbf{(13)}$$

A definição do valor das constantes C_u e C_v face a A_k pode ser igualmente obtida através da equação (7).

Tendo em conta estas últimas considerações, temos que

$$\begin{aligned} s_{xy} &= 0{,}25 \times \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C_u C_v S_{uv} \cos\left(\frac{(x+1)u\pi}{16}\right)\cos\left(\frac{(y+1)v\pi}{16}\right) \\ &= 0{,}25 \sum_{u=0}^{7} C_u \cos\left(\frac{(x+1)u\pi}{16}\right)\sum_{v=0}^{7} C_v S_{uv} \cos\left(\frac{(y+1)v\pi}{16}\right), x, y \in [0, N-1] \end{aligned} \quad \textbf{(14)}$$

Tal como referido para a FDCT, esta optimização consiste na aplicação da transformada nas linhas, uma para cada linha, e, posteriormente, nas colunas, uma por coluna. Desta forma, obtêm-se os coeficientes da IDCT, tal como em (14).

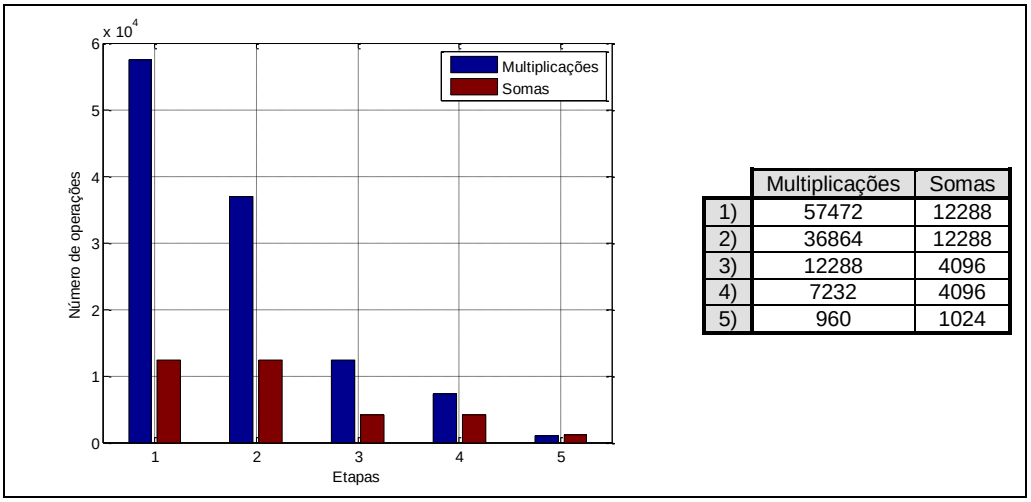
De modo a incluir as optimizações realizadas anteriormente neste cálculo, nomeadamente a do cálculo das constantes, é necessário adaptar (14) nesse sentido. A nova equação é dada por

$$s_{xy} = \sum_{u=0}^{7} \cos\left(\frac{(x+1)u\pi}{16}\right)\sum_{v=0}^{7} 0{,}25 C_u C_v S_{uv} \cos\left(\frac{(y+1)v\pi}{16}\right), x, y \in [0, N-1] \quad \textbf{(15)}$$

Assim, calculando todos os valores de *y* em função de *x*, ou vice-versa, reduz-se o número de multiplicações a realizar para 960 e o de somas para 1024.

Tal como para o cálculo da FDCT, o número de operações no cálculo da IDCT pode ser bastante reduzido através das técnicas analisadas (ver Figura 6). Porém, ao calcular *à priori*

o produto entre constantes C_u e C_v com o coeficiente S_{uv}, obtém-se o mesmo número de operações necessárias para realizar o cálculo dos coeficientes da FDCT.



	Multiplicações	Somas
1)	57472	12288
2)	36864	12288
3)	12288	4096
4)	7232	4096
5)	960	1024

Figura 6 – Representação das operações das várias optimizações utilizadas na IDCT (1) – IDCT (sem optimizações); 2) – Cálculo das constantes; 3) – Resolução dos co-senos; 4) – Casos particulares; 5) – Separabilidade da IDCT).

2.2. Quantificação e Quantificação Inversa

Devido às características do SVH, é possível eliminar parte da informação sem que com isso se destaque uma diferença acentuada entre a imagem original e a descodificada. Através do bloco de Quantificação essa limitação é explorada, resultando na redução do número de coeficientes não nulos necessários para a representação (compacta) de uma imagem. A existência de elementos nulos sequenciais permite a redução da informação a transmitir/armazenar, através da codificação de entropia. Este tema será abordado na secção 2.3.

Após a aplicação da FDCT obtém-se uma matriz que traduz as variações de frequência a duas dimensões, tal como referido na secção **Erro! A origem da referência não foi encontrada.**. Atendendo às restrições do SVH define-se uma segunda matriz, Q_{uv}, que indica quais as frequências que traduzem uma maior importância visual para o SVH, i.e., as frequências às quais o SVH é mais sensível.

No formato JPEG as imagens de cor são processadas utilizando o espaço de cor YCbCr. Apesar de a norma não definir a utilização de matrizes específicas, são apresentados exemplos para a luminância (Y) e a crominância (Cb e Cr). Essas matrizes encontram-se exemplificadas na Tabela 4 e Tabela 5, respectivamente.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tabela 4 – Tabela de quantificação da luminância (adaptado de [1]).

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Tabela 5 – Tabela de quantificação da crominância (adaptado de [1]).

Com estas duas matrizes, e através de uma divisão inteira elemento a elemento, reduz-se o número de coeficientes não nulos.

$$Sq_{uv} = round\left(\frac{S_{uv}}{Q_{uv}}\right) \quad \textbf{(16)}$$

Ao utilizar os valores das tabelas anteriores, verifica-se que o divisor assume valores elevados no canto inferior direito das matrizes. Assim, esta operação conduz a um grande número de coeficientes nulos que se revelarão úteis para a codificação de entropia (abordada na secção 2.3).

A operação de quantização inversa recupera parte da informação dos elementos não nulos através da multiplicação da mesma matriz utilizada no processo de Quantificação.

$$R_{uv} = Sq_{uv} \times Q_{uv} \quad \textbf{(17)}$$

Como se observa através da Tabela 4 e Tabela 5, os coeficientes que pertencem à gama de frequências mais altas apresentam valores mais elevados. Como uma das características

da FDCT é a conservação de energia nos coeficientes de baixa frequência [12, 13], as gamas de frequência mais altas são praticamente anuladas. Devido ao SVH não ser sensível às rápidas transições, a anulação das frequências altas não se torna problemática.

2.3. Codificação de entropia

Após a quantificação, os coeficientes DC e AC são codificados de modo distinto. Os coeficientes DC são codificados de modo diferencial, i.e. o coeficiente DC de um bloco será subtraído ao coeficiente do bloco DC anterior. Esta operação reduz ainda mais a quantidade de informação a ser armazenada e/ou transmitida.

Por sua vez, os coeficientes AC são codificados usando uma sequência em zigue-zague, tal como exemplificado na Figura 7. Esta sequência junta um grande número de coeficientes nulos, obtidos através da Quantificação. Aplicando a codificação *Run-Length Encoding* (RLE) após a sequência, reduz-se a quantidade de informação a armazenar/transmitir, através da agregação de sequências de zeros [30].

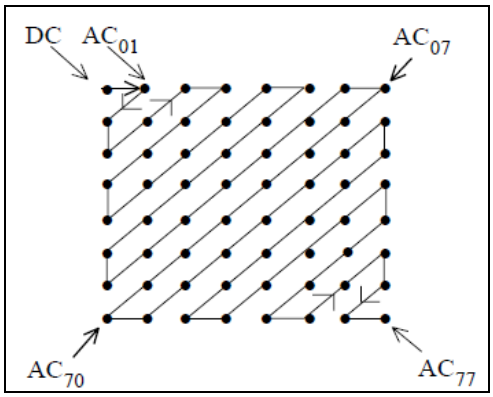


Figura 7 – Sequência zigue-zague dos coeficientes da DCT quantificados (adaptado de [1]).

Após o cálculo e quantificação dos coeficientes DC e AC, os mesmos são novamente codificados com recurso à codificação de entropia. A codificação de entropia obtém maior compressão (sem perda) dos coeficientes, através de uma análise estatística dos mesmos. A compressão é conseguida através da codificação de símbolos com maior ocorrência, de dimensão fixa, em símbolos de menor dimensão. Este processo é realizado em dois passos: conversão dos coeficientes quantificados para uma sequência intermédia e posterior conversão dessa sequência em símbolos.

Para esta nova etapa podem ser utilizados dois tipos de codificação de entropia: codificação aritmética [31] ou Huffman [32]. Apesar dos resultados obtidos utilizando a codificação aritmética serem superiores face à utilização da Huffman (taxa de compressão superior entre os 5% e 10% [30]), o tipo de codificação de entropia adoptado neste trabalho é de Huffman por ser a utilizada na implementação de referência do codificador (chamada de JPEG *baseline*). Esta opção resulta do facto da codificação aritmética apresentar uma complexidade mais elevada face à codificação de Huffman [30]. Sobre a codificação aritmética eficiente existem questões legais relativas a patentes..

A codificação de Huffman necessita que sejam especificadas pelo menos duas tabelas: uma para os coeficientes DC e outra para os coeficientes AC. Essas tabelas são utilizadas para o processo de codificação e descodificação. A obtenção destas pode ser realizada através de um levantamento estatístico das características dos coeficientes de uma dada imagem, em tempo real, ou através de tabelas predefinidas. Na norma encontra-se um conjunto de tabelas que foram obtidas através de um estudo a um elevado número de imagens mas, tal como acontece com as tabelas de Quantificação, não é exigido o uso obrigatório destas. As tabelas para 8 bits que são utilizadas no decorrer deste trabalho podem ser consultadas no Anexo C.

Os coeficientes quantificados, DC ou AC, são representados por dois símbolos na sequência intermédia. Para ambos os casos, o primeiro símbolo define qual o número de bits que será utilizado para representar a amplitude do coeficiente, denominado de SIZE. Para os coeficientes AC, o primeiro símbolo contém também a sequência de coeficientes nulos consecutivos até ao coeficiente a ser quantificado. Esta sequência, também denominada por RUNLENGTH, agrega no máximo 16 coeficientes nulos num só símbolo. Caso o número de coeficientes nulos seja superior a 16, o símbolo é codificado como (15,0). Esta representação indica que existe uma sequência de 16 zeros consecutivos em que o coeficiente seguinte é igualmente nulo. No entanto, existem casos em que a sequência de zeros é contínua até ao final da matriz. Para estas situações é utilizado o símbolo (0,0), também conhecido como EOB. Este símbolo indica que até ao 64º coeficiente não existe qualquer coeficiente não nulo. O segundo símbolo define a amplitude do coeficiente com o número de bits especificado em SIZE. Através deste processo agrupam-se os bits contíguos com valor '0', reduzindo a informação a ser codificada. Na Tabela 6 verifica-se a correspondência de cada símbolo por cada coeficiente.

DC		AC	
Símbolo 1	Símbolo 2	Símbolo 1	Símbolo 2
(SIZE)	(AMPLITUDE)	(RUNLENGTH, SIZE)	(AMPLITUDE)

Tabela 6 – Representação dos coeficientes após a sequência intermédia.

Após esta sequência intermédia, os novos símbolos são codificados segundo um código de tamanho de símbolo variável, também conhecido por *Variable Length Coding* (VLC). O primeiro símbolo é codificado com recurso às tabelas presentes no Anexo C. Por sua vez, no segundo símbolo é obtido com recurso à Tabela 7. Na secção 2.4 apresenta-se um exemplo da conversão dos símbolos.

AMPLITUDE	Símbolo 2
-1, 1	0, 1
-3, -2,2,3	00, 01, 10, 11
-7, …, -4, 4, …, 7	000, …, 011, 100, …, 111
-15, …, -8, 8, …, 15	0000, …, 0111, 1000, …, 1111
-31, …, -16, 16, …, 31	00000, …, 01111, 10000, …, 11111
-63, …, -32, 32, …, 63	000000, …, 011111, 100000, …, 111111
-127, …, -64, 64, …, 127	0000000, …, 0111111, 1000000, …, 1111111
-255, …, -128, 128, …, 255	00000000, …, 01111111, 10000000, …, 11111111
-511, …, -256, 256, …, 511	000000000, …, 011111111, 100000000, …, 111111111
-1023, …, -512, 512, …, 1023	0000000000, …, 0111111111, 1000000000, …, 1111111111
-2047, …, -1024, 1024, …, 2047	00000000000, …, 01111111111, 10000000000, …, 11111111111

Tabela 7 – Estrutura do segundo símbolo da sequência intermédia.

2.4. Exemplo de codificação de um bloco de amostras

Após a especificação de cada bloco constituinte da norma JPEG, é apresentado um exemplo da codificação de uma matriz de 8x8 amostras. Apenas será representada a matriz referente à luminância (para as matrizes da crominância, apenas as tabelas de Quantificação são alteradas), descrita na Tabela 8.

100	47	44	93	89	33	30	81
59	40	44	70	75	59	62	86
34	58	61	45	50	76	80	61
39	90	87	31	27	76	73	20
39	90	87	31	27	76	73	20
34	58	61	45	50	76	80	61
59	40	44	70	75	59	62	86
100	47	44	93	89	33	30	81

Tabela 8 – Matriz de entrada.

Como a norma sugere, é realizada a subtracção de cada elemento da matriz por 128, originando uma segunda matriz. Com a aplicação da FDCT, é gerada a matriz de coeficientes, que traduz a importância visual que as várias frequências espaciais a 2D têm na matriz apresentada na Tabela 8. A matriz de coeficientes, com precisão a 3 casas decimais, encontra-se representada na Tabela 9.

480,000	-10,197	0,000	-0,847	0,000	-0,815	0,000	-0,798
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
28,230	0,000	-1,000	0,000	159,537	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	66,287	0,000	0,098	0,000	-0,098	0,000	0,069
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,213	0,000	0,000	0,000	1,026	0,000	-1,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Tabela 9 – Matriz dos coeficientes, com representação até à 3.ª casa decimal.

Aplicando a matriz de quantificação da luminância, verifica-se que o número de elementos nulos é considerável. A matriz dos coeficientes quantificados encontra-se representada na Tabela 10.

30	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	0
0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabela 10 – Matriz dos coeficientes quantificados.

Aplicando a sequência ziguezague aos coeficientes AC, obtém-se um vector com os coeficientes quantificados. O coeficiente DC não foi codificado diferencialmente devido à inexistência de bloco anterior. Esta situação em particular é aplicada ao primeiro macrobloco de uma imagem a ser processada:

(30, -1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Com os coeficientes ordenados é aplicada a codificação de Huffman. Através da matriz obtida, e com recurso às tabelas especificadas, a sequência codificada será a seguinte:

(5) (30); (0,1) (-1); (1,2) (2); (15,2) (3); (5,3) (-4); (0,0)

Após obtida a sequência intermédia, são aplicados os códigos de comprimento variável. Recorrendo às tabelas do Anexo C, obtêm-se os símbolos presentes na Tabela 11.

	(5) (30)	(0,1) (-1)	(1,2) (2)	(15,2) (3)	(5,3) (-4)
Símbolo 1	110	00	11011	1111111111110110	1111111111100111110
Símbolo 2	00001	0	10	11	011

Tabela 11 – Correspondência entre os símbolos obtidos através da sequência e o código de comprimento variável.

No final, a sequência de bits obtida será composta por:

110 00001, 00 0, 11011 10, 1111111111110110 11, 1111111111100111110 011, 1010 (EOB)

3. Análise Experimental do CODEC

A partir da análise teórica do CODEC JPEG apresentada no capítulo 2, é necessário averiguar que optimizações adicionais podem ser realizadas para uma implementação eficiente do CODEC em FPGA. Para tal foram realizadas simulações com recurso à ferramenta MATLAB. As simulações realizadas focaram-se essencialmente no número de bits que será utilizado para a representação decimal dos elementos de cada bloco. Como o sistema é digital, o número de bits é uma das limitações impostas à partida. Quanto maior for este número, mais portas lógicas serão utilizadas e, por consequência, o consumo de energia será acrescido e uma maior área será ocupada na FPGA. Tendo em conta esta constatação, existem conceitos utilizados no domínio digital que necessitam de especial atenção:

- Representação em código binário de quantidades;
- Truncatura de dados.

Atendendo à forma de representação utilizada nos processadores digitais, código binário natural ou complemento para dois, existe a necessidade de adoptar um formato que permita a transposição da representação decimal para estasa representações binárias. Os formatos de vírgula fixa (*fixed point*) [16] e vírgula flutuante (*floating point*) [16] são os mais comuns para realizar essa transposição. O primeiro formato, vírgula fixa, será utilizado no decorrer do trabalho, uma vez que a exigência a nível computacional é menor face ao segundo [16]. O erro máximo obtido na utilização deste formato é dado por:

$$err = \pm \frac{1}{2^{N+1}} \tag{18}$$

em que N representa o número de bits a utilizar.

No formato de vírgula fixa os números são manipulados e representados por símbolos, em que ambas as partes, inteira e decimal, são representadas por um número fixo de bits. É então possível especificar quantos bits são utilizados para representar um número racional dividindo o símbolo em parte inteira (unidades, dezenas ou centenas, etc.) e parte decimal. Por exemplo, para um caso onde é apenas necessária a presença de sinal na parte inteira, e tendo no máximo 8 bits, existe profundidade para ir até à 7.ª casa decimal. O método de cálculo utilizado é baseado em potências de 2 negativas, i.e., à medida que se vai diminuindo o peso de cada bit, menor será o seu valor associado, tal como exemplificado na Tabela 12.

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
+ / -	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125

Tabela 12 – Representação do formato Q1.7.

Como a utilização do formato de vírgula fixa traduz-se numa quantificação, esta não está isenta de introduzir um erro, caso não seja possível representar o valor pretendido por um dado número de bits. Assim, para um número como 0,046875, a representação do mesmo em binário com 8 bits de precisão é ‘00000110’ (0,03125 + 0,015625 = 0,046875). Para casos em que não é possível representar, como por exemplo 0,046874, é realizada uma

operação de arredondamento utilizando um 9.º bit. Caso esse 9º bit (que não se encontra representado) seja 1 acrescenta-se uma unidade ao valor representado, caso contrário nada é realizado. Recorrendo a (18) verifica-se que o erro máximo para o exemplo apresentado acima é de 0,00390625.

Nas operações de multiplicação, o tamanho em bits do símbolo do produto equivale à soma do tamanho dos símbolos dos seus operandos. Por exemplo, para duas amostras codificadas com 8 bits cada, o produto de ambos terá um máximo de 16 bits (8 + 8). Para sequências de cálculo iterativas, como acontece no cálculo dos coeficientes da FDCT e IDCT em que o produto de uma iteração é um operando da multiplicação seguinte, o tamanho dos símbolos irá crescendo à medida que se vão realizando novas multiplicações, traduzindo-se em multiplicadores mais complexos. Um exemplo do aumento do tamanho dos símbolos encontra-se ilustrado na Figura 8.

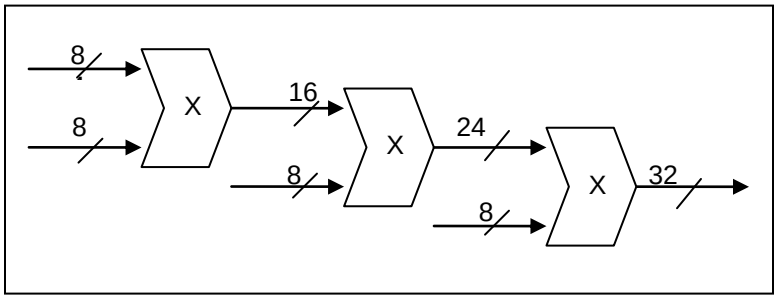


Figura 8 – Exemplo do número de bits utilizado numa multiplicação em escada.

Como se pode verificar pela Figura 8, o tamanho do símbolo vai aumentando à medida que se vão realizando sucessivas multiplicações. Como estas iterações crescem linearmente, torna-se essencial truncar o valor do produto após cada multiplicação. Desta forma limita-se o tamanho máximo do símbolo. Porém, ao truncar o produto é gerado um erro que pode comprometer a veracidade do resultado final da operação.

Ambos os métodos descritos anteriormente, o formato de vírgula fixa e a truncatura de produtos, introduzem erro. Esse erro terá que ser cuidadosamente avaliado para que a qualidade da imagem final não sofra perturbações perceptíveis pelo SVH.

Como medida de avaliação de qualidade da imagem codificada com erro, é considerada a métrica *Signal to Noise Ratio* (SNR). Esta métrica afere se o erro introduzido num dado sistema é aceitável ou não. A sua expressão é dada por

$$SNR_{[dB]} = 10 \times \log_{10}\left(\frac{\sum_x \sum_y n[x,y]^2}{\sum_x \sum_y (n[x,y] - \hat{n}[x,y])^2} \right) \quad \textbf{(19)}$$

em que n representa a imagem original e $\hat{n}$ imagem obtida após descodificação.

De modo a garantir que o CODEC a implementar seja abrangente, foram utilizadas um conjunto de imagens de teste distintas nas simulações realizadas. As imagens monocromáticas (imagem representada em níveis de cinzento) utilizadas têm resolução de 256 linhas por 256 colunas e nas imagens a cores a resolução varia entre 512 linhas por 512 colunas e 768 linhas por 512 colunas. As imagens de teste podem ser consultadas no Anexo A. Por outro lado, na implementação do CODEC será dada maior relevância a imagens

naturais face às obtidas computacionalmente atendendo a que o formato JPEG é o mais utilizado para imagens fotográficas/naturais, tal como mencionado anteriormente.

Através da métrica da SNR verificou-se, após codificação e descodificação das várias imagens de teste consideradas, quais os valores que são obtidos para cada imagem. Na Tabela 13 e na Tabela 14 encontram-se os resultados obtidos para as imagens monocromáticas e imagens a cores, respectivamente, utilizando32 e 64 bits. O número de bits utilizado para obter estes valores serve apenas como referência de modo a se poder iniciar o estudo do CODEC JPEG a implementar. A imagem monocromática "*squares*" é construída computacionalmente sendo portanto um caso particular relativamente ao seu conteúdo. A análise e avaliação desta imagem encontra-se descrita no Anexo B.

Imagem	SNR		Imagem	SNR	
	32 bits	64 bits		32 bits	64 bits
Bird	32,67317	32,67317	**Horiz**	28,00729	28,00729
Bridge	22,91603	22,91603	**Lena**	25,73171	25,73171
Camera	26,05653	26,05653	**Montage**	28,36343	28,36343
Circles	26,87683	26,87641	**Slope**	34,65596	34,65596
Crosses	14,75260	14,75260	**Squares**	Inf	Inf
Goldhill	24,26650	24,26650	**Text**	21,49284	21,49284

Tabela 13 – SNR de cada imagem monocromática para representação a 32 e 64 bits (representação até à 5.ª casa decimal).

Imagem	SNR	
	32 bits	64 bits
Lena3	31,36205	31,36205
Monarch	30,33477	30,33477
Peppers3	31,69453	31,69453
Sail	27,12250	27,12250
Tullips	29,72421	29,72421

Tabela 14 – SNR de cada imagem a cores para representação a 32 e 64 bits (representação até à 5.ª casa decimal).

Com base nos valores apresentados anteriormente, verifica-se que a utilização de 32 ou 64 bits nos cálculos para codificação de uma imagem não apresenta qualquer diferença em termos de qualidade da imagem final. Assim, o estudo a realizar em cada bloco contará com um máximo de 32 bits. Este valor será referido ao longo deste trabalho como SNR máxima ou SNR max.

No decorrer deste capítulo é apresentada a análise para apenas três imagens de carácter natural de cada tipo de imagem (monocromática e a cores). A primeira análise é realizada a imagens monocromáticas e posteriormente são analisadas as imagens a cores. As imagens monocromáticas a estudar são:

- "*camera*", devido à presença de elevadas transições de intensidade;
- "*goldhill*", uma imagem típica de uma paisagem;
- "*lena*", uma imagem típica de retrato.

As imagens a cor a analisar são:

- "*lena3*", uma imagem típica de retrato;
- "*peppers3*", devido à presença de elevadas transições de cor;

- "*sail*", uma imagem típica de paisagem.

Ao longo das próximas secções será dada maior importância à componente da luminância face à crominância, devido à resposta que o SVH apresenta a estas componentes, sendo mais sensível à primeira.

3.1. Forward Discrete Cosine Transform

Do ponto de vista teórico, e considerando cálculo simbólico, a operação FCDT-IDCT é ideal, i.e., não tem perdas (*lossless*), tal como referido na secção **Erro! A origem da referência não foi encontrada.**. Porém, devido à utilização de processamento digital, que tem uma representação numérica limitada, é inevitável a existência de erro no cálculo destas transformadas reais: como são realizadas à custa de co-senos, e estes produzem valores que são dízimas infinitas, irá existir sempre erro na sua representação devido à utilização de um número limitado de bits. Como a presença de erro é impossível de anular, torna-se necessário avaliar o erro máximo admissível no sistema a desenvolver de forma a não degradar fortemente a qualidade da imagem final.

De forma a arranjar um compromisso entre o número de bits a utilizar face à qualidade da imagem pretendida, é realizado um estudo sobre o impacto que a redução do número de bits terá. Esse estudo foi realizado utilizando como blocos funcionais um módulo de cálculo da FDCT, na versão desenvolvida neste trabalho, e a jusante deste um módulo de cálculo da IDCT, que aplica a função idct2 implementada pelo MATLAB.

Numa primeira fase é necessário calcular os valores dos co-senos e das constantes, mencionadas na secção 2.1.1, em função do número de bits a utilizar. Como ambos variam entre -1 e 1, é apenas utilizado um bit para a parte inteira (formato de vírgula fixa), sendo os restantes bits atribuídos à parte decimal. Considerando uma precisão máxima de 32 bits, o número limite de bits a considerar para a parte decimal dos valores dos co-senos e das constantes é de 31 bits. Com a utilização de apenas 3 bits no total (1 para a parte inteira e 2 para a parte decimal), a SNR obtida para algumas imagens, após o cálculo da FDCT-IDCT, é negativa, como se verifica na Figura 9. Assim, o valor mínimo considerado é de 4 bits no total (1 para a parte inteira e 3 para a parte decimal), o qual assegura SNR positiva, i.e. valores para os quais a potência do ruído não se sobrepõe à da própria imagem.

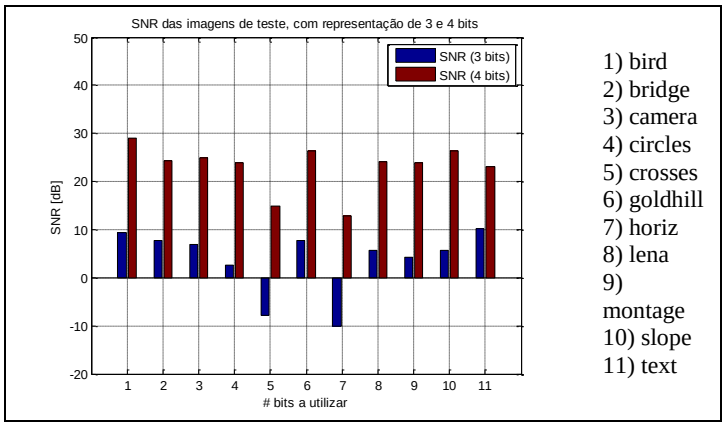


Figura 9 – SNR das imagens monocromáticas de teste, com representação de 3 e 4 bits (1 para parte inteira+ 2 ou 3, respectivamente, para a parte decimal).

Como o erro será tanto maior quanto menor for o número de bits utilizados (o intervalo de representação será mais curto à medida que se aumenta o número de bits), é expectável que, para uma dada imagem, a SNR seja tanto melhor quanto maior for o número de bits a incluir na representação simbólica dos valores dos co-senos e constantes. Nalguns casos, esta não será ideal devido à introdução de erro no arredondamento desses valores.

Na Figura 10 representa-se a SNR da FDCT das imagens de teste monocromáticas consideradas, onde é variado o número de bits a utilizar para os valores dos co-senos e das constantes. Nesta, só estão representados os valores da SNR até 9 bits, uma vez que após este número de bits os valores à saída do sistema são muito próximos ou mesmo idênticos aos valores de entrada (SNR a tender para ∞).

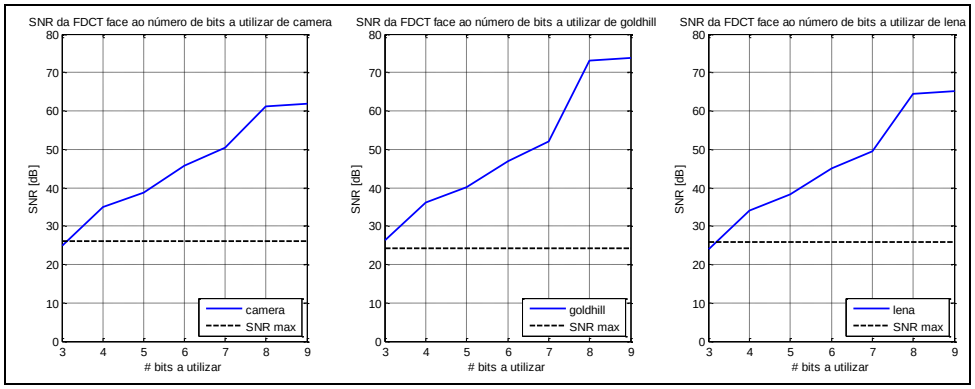


Figura 10 – SNR da FDCT desenvolvida em função do número de bits a utilizar para imagens monocromáticas.

Através da Figura 10 verifica-se que com 4 bits para representação da parte decimal dos co-senos, o valor da SNR da FDCT é superior ao valor obtido para a SNR da imagem (SNR max). Verifica-se também que com o aumento do número de bits a SNR da FDCT melhora. Este efeito deve-se à redução do erro provocado pela quantificação dos valores dos co-senos, bem como do resultado de cada operação de multiplicação da FDCT. Em alguns casos o erro provocado pode ser construtivo, o que resulta num aumento significativo da SNR, tal como se verifica na codificação dos símbolos com 8 bits. No entanto existem outros casos em que o erro é destrutivo e influência negativamente o ganho da SNR, tal como se verifica na codificação dos símbolos com 9 bits, em que o acréscimo não é tão acentuado.

De modo a verificar qual a componente que impõe um comportamento não linear, foram retiradas as constantes do cálculo da FDCT. Estas constantes são introduzidas, posteriormente, sem erro e traduzem-se numa melhor SNR, tal como representado na Figura 11.

Ao comparar a Figura 10 com a Figura 11, verifica-se que ao introduzir as constantes *à posteriori*, e sem utilizar o formato de vírgula fixa, a SNR melhora consideravelmente (a diferença mínima obtida é de 3 dB para os valores codificados com 3 bits). Codificando os valores com 8 ou mais bits, a imagem obtida é próxima ou igual à imagem original. Constata-se também que a SNR da FDCT para a gama de bits utilizada é sempre superior ao valor obtido através do sistema ideal.

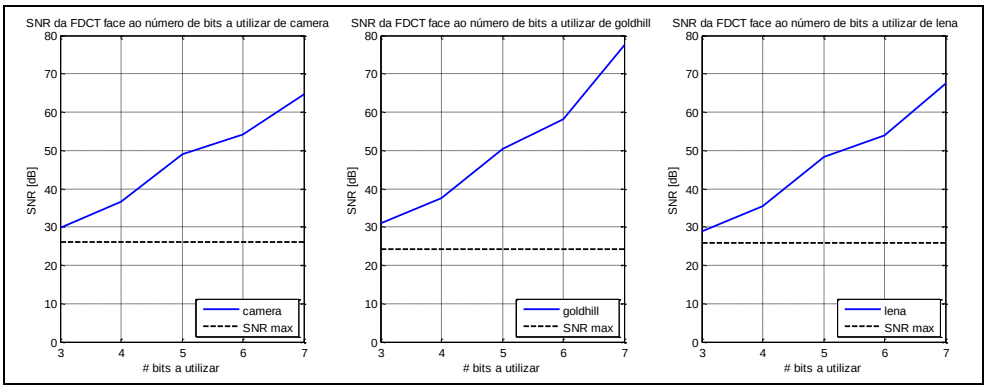


Figura 11 – SNR da FDCT desenvolvida em função do número de bits a utilizar, sem o produto das constantes para imagens monocromáticas.

Assim, optou-se por incluir as constantes no cálculo posteriormente e em conjunto com o bloco da Quantificação, que é abordado na secção 3.2. Com esta adaptação constata-se que este bloco não calcula o valor exacto dos coeficientes mas uma parte dos valores, já que as constantes foram retiradas.

Devido ao facto do tamanho dos símbolos crescerem com cada multiplicação (ver secção 3), é necessário truncar o valor de cada produto de modo a que os recursos utilizados não cresçam linearmente após cada operação de multiplicação. Como a truncatura de valores impõe erros de precisão, é necessário realizar um estudo sobre os resultados obtidos anteriormente. Na Figura 12 estão representadas as SNR para a truncatura à parte inteira e para truncatura até à 5.ª e à 10.ª casas decimais. Neste teste apenas não são atingidos os valores ideais (SNR a tender para ∞) para a truncatura real, i.e. sem parte decimal. Em relação ao número de bits a utilizar, é atingida a SNR máxima para a truncatura completa após a representação simbólica com 7 bits, i.e. com mais de 7 bits não se verifica qualquer ganho na SNR com o aumento do tamanho do símbolo. Para as restantes não existe qualquer diferença face ao que foi apresentado na Figura 11.

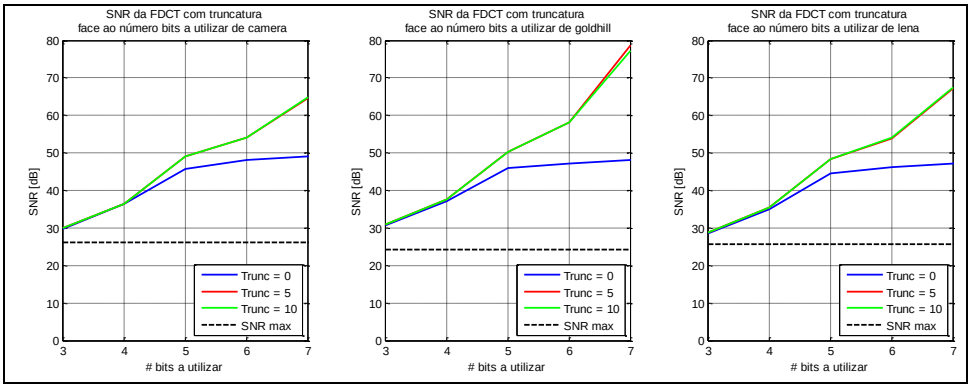


Figura 12 – SNR da FDCT desenvolvida com truncatura total e truncatura até à 5ª e à 10.ª casa decimal para imagens monocromáticas.

De modo a avaliar o compromisso que se pode obter através do número de bits e a truncatura a utilizar, foi realizado um estudo mais pormenorizado. Este estudo indica os valores das SNR desde a truncatura completa, até à truncatura de valores até à 5.ª casa decimal. Os gráficos obtidos encontram-se representados na Figura 13.

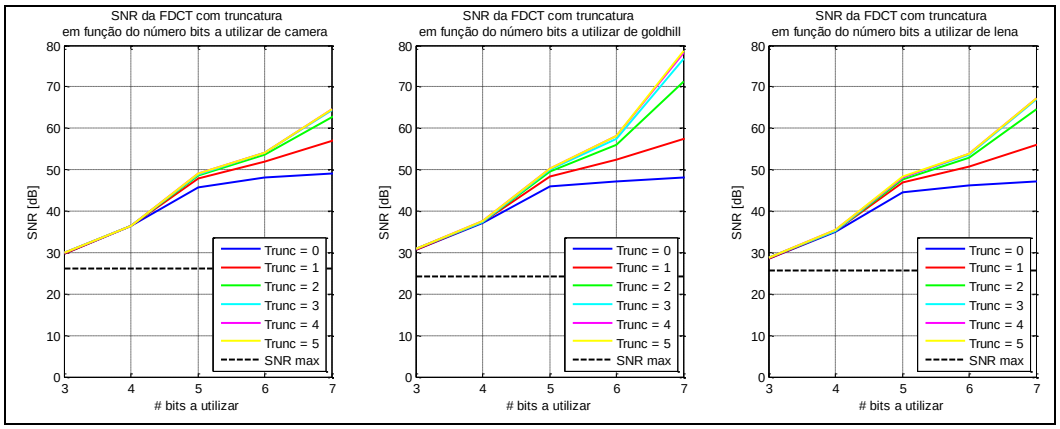


Figura 13 – SNR da FDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens monocromáticas.

Através da Figura 13, verifica-se que a a representação de valores com até 3 casas decimais é suficiente para que a perda existente no cálculo da FDCT seja praticamente nula. Assim, para este bloco é apenas necessário considerar a utilização de 8 bits para codificar os valores das amostras e truncatura de valores até à 3.ª casa decimal para se obter uma imagem semelhante ou idêntica à imagem original. No entanto, com uma representação com 4 bits e sem parte decimal, a imagem obtida, somente através do cálculo FDCT-IDCT, tem qualidade superior à utilizada no sistema implementado em MATLAB, tal como se verifica através da Figura 13.

Para as imagens a cores verifica-se o mesmo comportamento, tal como se pode observar na Figura 14.

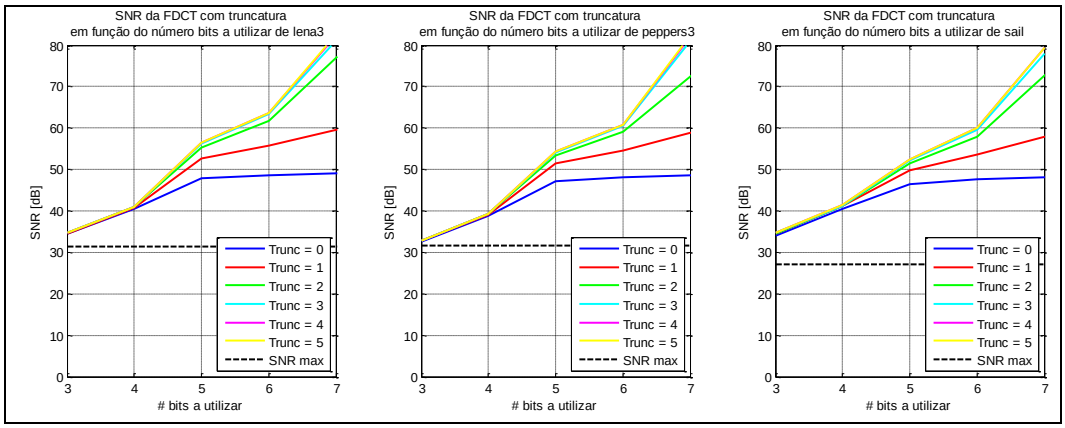


Figura 14 – SNR da FDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens a cores.

Ao analisar a Figura 14 constata-se que, tal como para as imagens monocromáticas, a representação até à 3.ª casa decimal é suficiente para que a perda inferida no cálculo da FDCT seja inferior à perda inferida pela implementação em MATLAB. Tal como para as imagens monocromáticas, obtém-se uma imagem final semelhante à imagem original utilizando truncatura até à 3ª casa decimal e uma representação simbólica de 8 bits.

Como a codificação não depende exclusivamente do bloco da FDCT, a escolha do número de bits a utilizar neste bloco, bem como a truncatura, é realizada através da análise conjunta dos blocos FDCT e Quantificação (secção 3.2).

3.2. Quantificação

Para os patamares da SNR máxima (SNR max) definidos no início deste capítulo, é necessário estudar o impacto que o bloco Quantificação tem no sistema relativamente à redução do número de bits usados para representar os valores descritos na secção anterior (secção **Erro! A origem da referência não foi encontrada.**).

A implementação deste bloco consiste numa divisão, seguida pelo arredondamento para o inteiro mais próximo do quociente obtido, tal como se verifica em (16). Devido à implementação de um circuito que realize divisões ser mais complexo face a um circuito que realize multiplicações e sendo a multiplicação a operação inversa da divisão, são calculadas as constantes com o valor inverso das constantes originais. Assim, a matriz de Quantificação a utilizar será obtida através de 1/Q, em que Q é a matriz apresentada na Tabela 4. Como as constantes $C_u C_v / 4$ não foram consideradas no estudo do bloco da FDCT, é necessário inclui-las também neste bloco.

Com as considerações apresentadas anteriormente é necessário verificar o impacto que a limitação do número de bits tem neste bloco. Como só é possível ter representação não nula em todos os coeficientes da tabela a partir de 8 bits para a representação decimal, 9 bits será o número mínimo de bits a utilizar. Como máximo, serão utilizados 31 bits para representar a parte decimal e 1 bit para representar a parte inteira.

Na Figura 15, Figura 16 e Figura 17 encontram-se representadas as SNR para cada conjunto de coeficientes gerados no bloco da FDCT, para as imagens monocromáticas, utilizando 8, 16 e 31 bits para a representação da parte decimal da matriz de quantificação. De modo a avaliar o impacto que o número de bits a utilizar tem no sistema de codificação, o estudo deste bloco foi realizado como continuação do anterior, i.e. os valores de entrada são os obtidos no bloco da FDCT proposto.

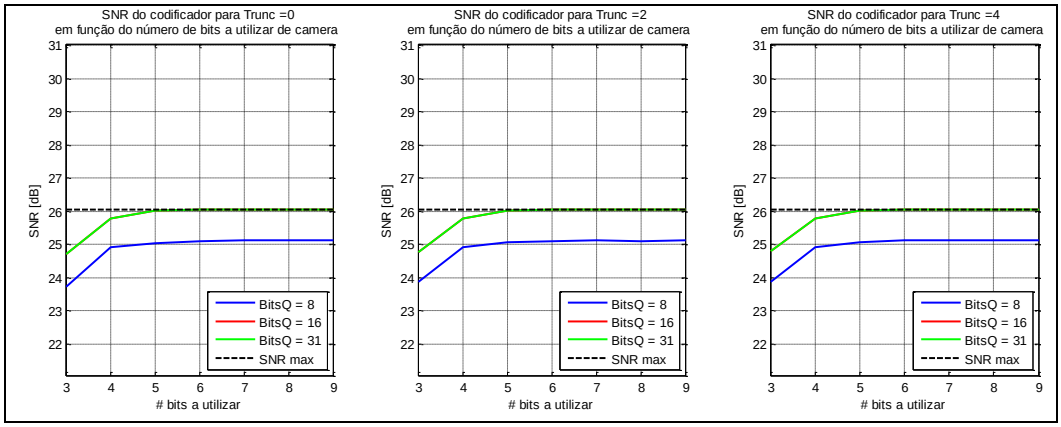


Figura 15 – SNR do bloco da Quantificação desenvolvido, em função do número de bits a utilizar nos co-senos, variando a truncatura, e na matriz de Quantificação, para a imagem *camera* para imagens monocromáticas.

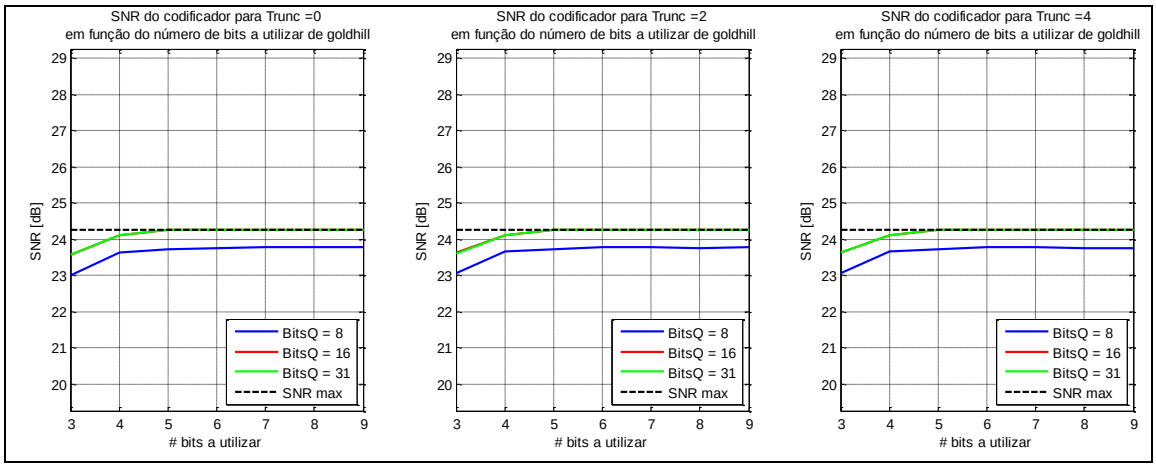


Figura 16 – SNR do bloco da Quantificação desenvolvido, em função do número de bits a utilizar nos co-senos, variando a truncatura, e na matriz de Quantificação, para a imagem *goldhill* para imagens monocromáticas.

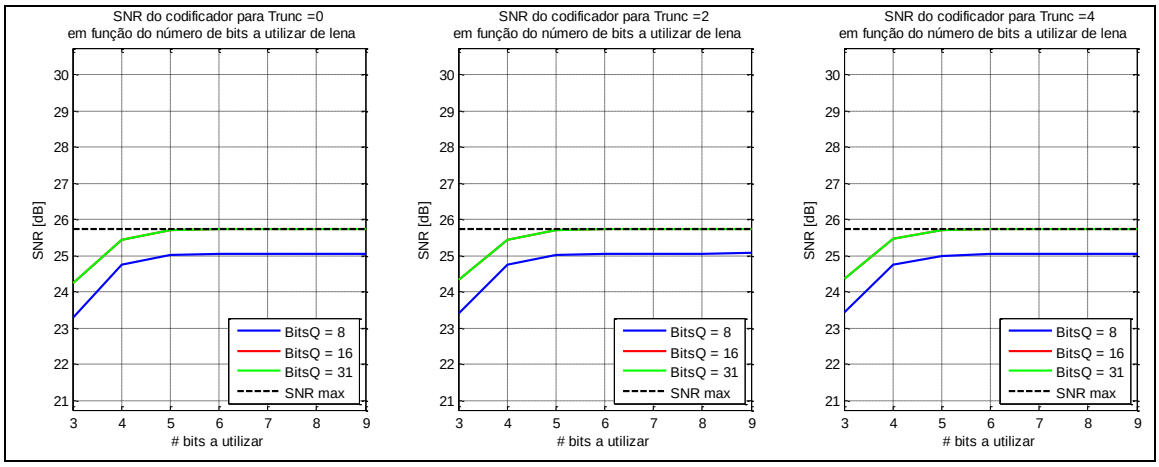


Figura 17 – SNR do bloco da Quantificação desenvolvido em função do número de bits a utilizar nos co-senos, variando a truncatura, e na matriz de Quantificação, para a imagem *lena* para imagens monocromáticas.

Através das figuras anteriores conclui-se que as diferenças na SNR para as várias truncaturas no bloco da FCDT são mínimas, independente do número de bits a utilizar na matriz de Quantificação. Assim, a truncatura total (apenas parte real) será utilizada após cada produto na FDCT.

No que respeita ao número de bits, verifica-se que existe uma diferença acentuada entre o patamar máximo obtido e a matriz codificada com 8 bits (para a parte decimal). No entanto, essa diferença rapidamente diminui com o aumento do número de bits, tal como se observar na Figura 18.

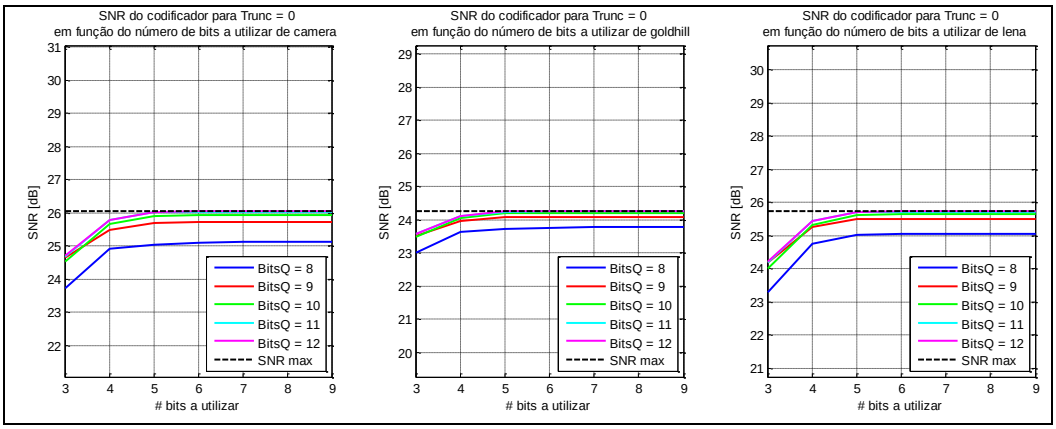


Figura 18 – SNR do bloco da Quantificação desenvolvido em função do número de bits a utilizar nos co-senos e na matriz de Quantificação para imagens monocromáticas.

Através da análise da Figura 18 conclui-se que a diferença entre o valor máximo da SNR (SNR max) e os valores de SNR obtidos a partir de 11 bits é mínima. O mesmo se verifica para as imagens a cores, tal como se observa através da Figura 19.

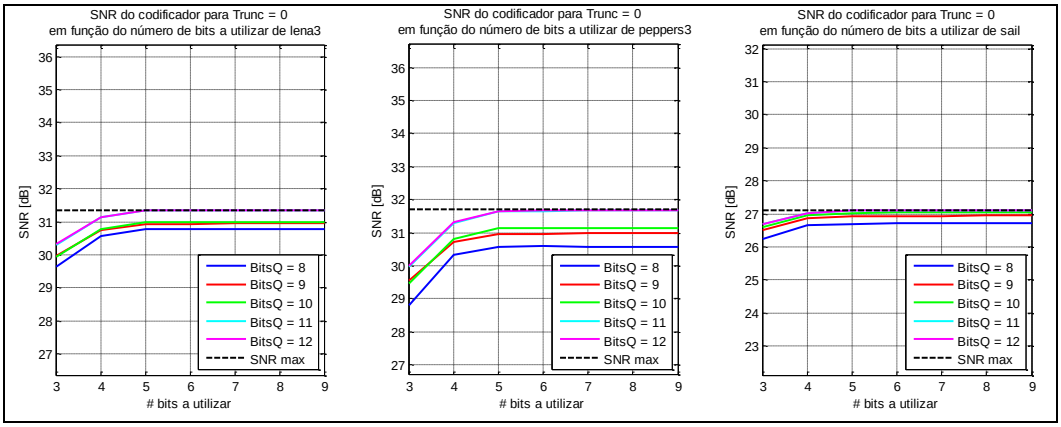


Figura 19 – SNR do bloco da Quantificação desenvolvido em função do número de bits a utilizar nos co-senos e na matriz de Quantificação para imagens a cores.

Através da consulta da Tabela 15 e da Tabela 16 conclui-se que a diferença entre 6 bits e 7 bits para a representação decimal dos co-senos no bloco da FDCT é mínima bem como a utilização de 11 ou 12 bits para o bloco da Quantificação. Assim, o bloco da FDCT terá truncatura completa após cada produto e os co-senos irão ter representação decimal de 6 bits.

camera		**Bits co-senos**			***goldhill***	**Bits co-senos**			***lena***	**Bits co-senos**		
		5	**6**	**7**		**5**	**6**	**7**		**5**	**6**	**7**
Bits Quant.	**8**	25,032	25,112	25,119	**8**	23,738	23,763	23,788	**8**	25,005	25,052	25,046
	9	25,710	25,736	25,734	**9**	24,092	24,094	24,100	**9**	25,484	25,492	25,484
	10	25,912	25,942	25,942	**10**	24,205	24,216	24,215	**10**	25,626	25,645	25,641
	11	26,019	26,035	26,035	**11**	24,247	24,254	24,256	**11**	25,696	25,711	25,715
	12	26,026	26,043	26,046	**12**	24,253	24,259	24,261	**12**	25,706	25,721	25,723
	Máx	26,057			**Máx**	24,267			**Máx**	25,732		

Tabela 15 – Valores das diferentes SNR, face ao número de bits a utilizar nos co-senos e na matriz de Quantificação, com representação de 3 casas decimais para imagens monocromáticas.

lena3		Bits co-senos			*peppers3*	Bits co-senos			*sail*	Bits co-senos		
		5	6	7		5	6	7		5	6	7
Bits Quant.	8	30,786	30,784	30,785	8	30,569	30,573	30,579	8	26,716	26,714	26,717
	9	30,944	30,944	30,943	9	30,977	30,985	30,981	9	26,943	26,944	26,944
	10	30,994	30,993	30,993	10	31,151	31,152	31,153	10	27,039	27,041	27,041
	11	31,350	31,349	31,349	11	31,672	31,673	31,673	11	27,111	27,112	27,111
	12	31,350	31,351	31,351	12	31,679	31,680	31,680	12	27,116	27,115	27,115
	Máx	31,362			Máx	31,695			Máx	27,123		

Tabela 16 – Valores das diferentes SNR, face ao número de bits a utilizar nos co-senos e na matriz de Quantificação, com representação de 3 casas decimais para imagens a cores.

Tal como afirmado anteriormente, conclui-se que com uma representação decimal com 11 e 12 bits, os valores da SNR do codificador desenvolvido são bastante próximos dos valores obtidos na Tabela 13 e na Tabela 14. Como a diferença entre a utilização de 11 e 12 bits é mínima (aproximadamente 0,01dB no máximo), a escolha recai sobre o menor número de bits.

3.3. Inverse Discrete Cosine Transform

Aplicando o procedimento descrito na secção **Erro! A origem da referência não foi encontrada.** ao bloco da IDCT, é necessário numa primeira fase calcular os valores dos co-senos e constantes em função do número de bits a utilizar. Assumindo as mesmas considerações (1 bit para a parte inteira e variação de 3 a 31 bits para a parte decimal), podemos afirmar que a SNR obtida para este bloco terá um comportamento semelhante ao bloco da FDCT, i.e. que a cada novo bit existe um acréscimo do valor da mesma. O cenário de teste é constituido pelos blocos funcionais da FDCT, que aplica a função dct2 implementada pelo MATLAB, e a jusante deste, um módulo de cálculo da IDCT, na versão desenvolvida neste trabalho.

Na Figura 20 encontra-se representada a SNR para as imagens monocromáticas utilizadas nos testes realizados, onde é variado o número de bits a utilizar para codificar os valores dos co-senos.

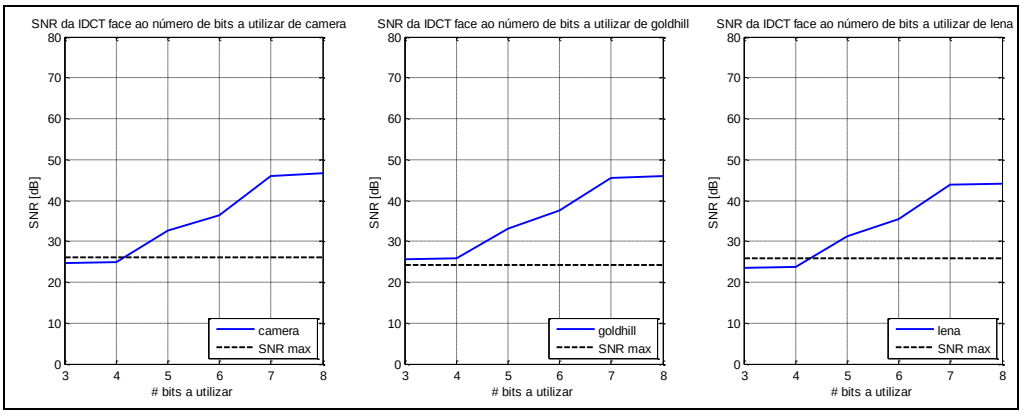


Figura 20 – SNR da IDCT desenvolvida em função do número de bits a utilizar para imagens monocromáticas.

Na Figura 20 verifica-se que o fenómeno registado para os valores da SNR na FDCT mantém-se, i.e., existe erro construtivo ou destrutivo para certas representações binárias. São necessários pelos menos 5 bits para que a SNR deste bloco seja igual ou superior à SNR obtida no sistema ideal e apenas 9 bits para que a imagem obtida seja semelhante, ou mesmo idêntica, à imagem original.

Tal como aconteceu na FDCT, é também possível separar as constantes referidas na secção 2.1.2 para o bloco da IDCT. Estas constantes vão ser inseridas a montante, no bloco da Quantificação Inversa, que é abordado na secção 3.4. Com esta alteração a SNR toma os valores representados na Figura 21.

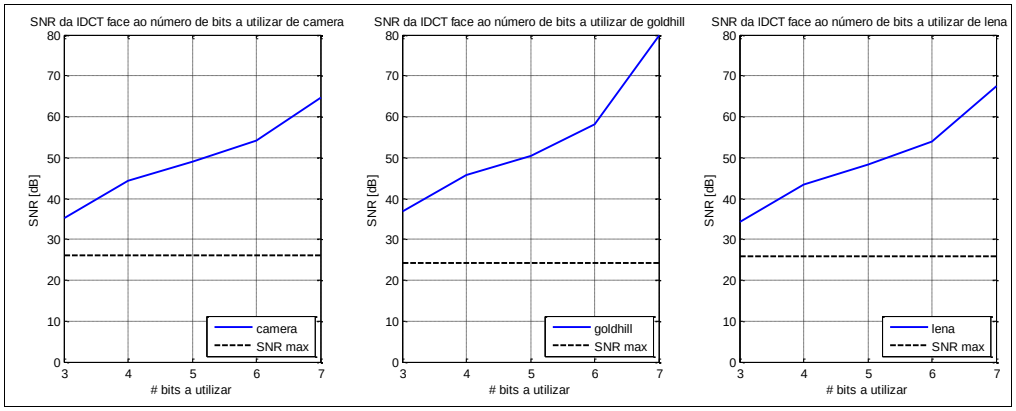


Figura 21 – SNR da IDCT desenvolvida em função do número de bits a utilizar, sem o produto das constantes para imagens monocromáticas.

Através da Figura 20 e da Figura 21, verifica-se que se obtém uma melhoria da SNR caso as constantes sejam calculadas separadamente, tal como verificado para a FDCT. Por outro lado, com a separação das constantes, é possível reutilizar os valores obtidos para os co-senos, tanto no cálculo da FDCT como no cálculo da IDCT.

Tal como se verificou para a FDCT, com esta alteração, o bloco da IDCT irá calcular parte dos coeficientes. Assim, deste ponto em diante será considerado que o bloco responsável pelo cálculo da IDCT não contempla as constantes referidas anteriormente.

No bloco da IDCT é presenciado o efeito exemplificado na Figura 8, à semelhança do que aconteceu para o bloco FDCT. Como tal, há necessidade de truncar valores para diminuir o consumo e o número de portas lógicas a utilizar. Na Figura 22 encontram-se representadas as SNR para as mesmas imagens utilizadas anteriormente, para a truncatura decimal completa (apenas parte inteira) e para truncatura até à 5.ª e à 10.ª casas decimais.

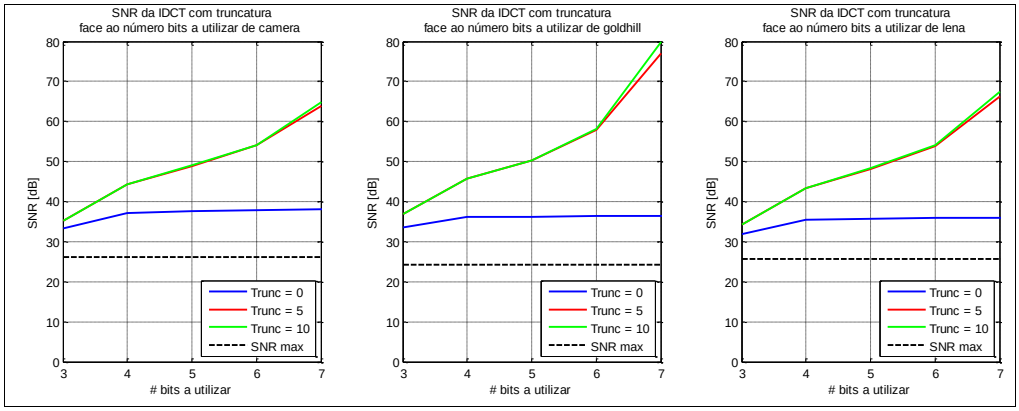


Figura 22 – SNR da IDCT desenvolvida com truncatura total e truncatura até à 5ª e à 10.ª casa decimal para imagens monocromáticas.

Tal como sucedeu com a FDCT, a utilização da truncatura completa, i.e. representação apenas com parte inteira, limita desde cedo o valor da SNR. A SNR tende para ∞ a partir de uma representação simbólica com 7 bits e truncatura a partir da 5.ª casa decimal. O número de bits a utilizar para a truncatura até à 5.ª e 10.ª casas decimais varia ligeiramente, para representação simbólica de 7 bits, logo a truncatura máxima a utilizar será até à 5ª cada decimal. Assim, e tal como realizado para o bloco da FDCT, também neste bloco é verificado qual o compromisso que se obtém entre o número de bits e a truncatura a utilizar. O gráfico obtido encontra-se representado na Figura 23.

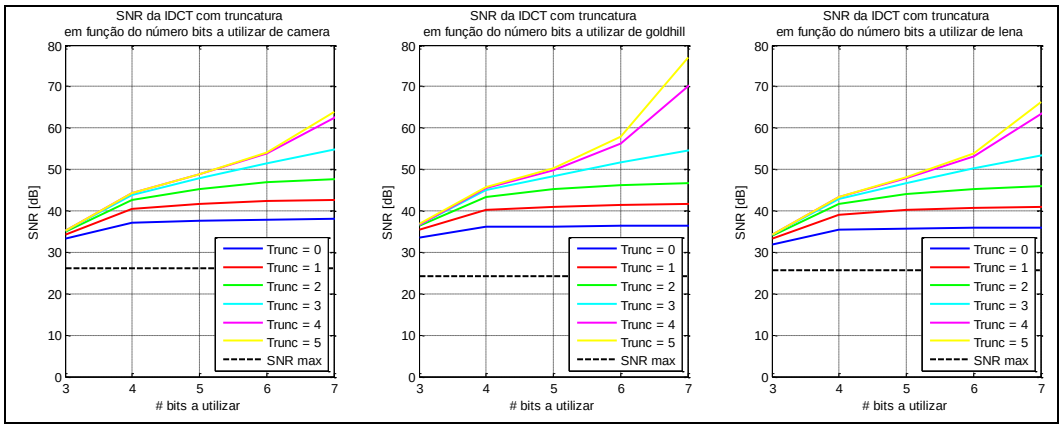


Figura 23 – SNR da IDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens monocromáticas.

Da Figura 23 constata-se que, para qualquer número de bits e qualquer truncatura a utilizar, a SNR da IDCT obtida é superior aos valores obtidos da SNR máxima. Para o cálculo dstes valores não foram considerados os blocos de quantificação (responsáveis por introduzir perda no sistema, tal como referido na secção 2.2). O mesmo resultado verifica-se para as imagens a cor, tal como se observa na Figura 24.

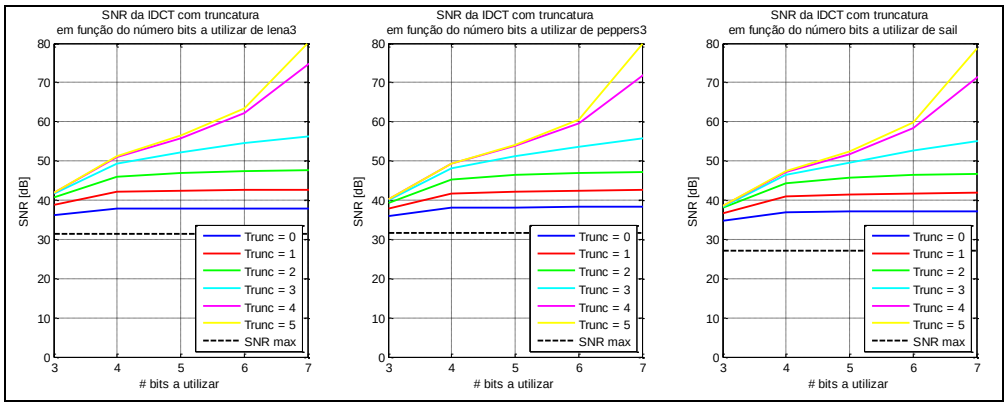


Figura 24 – SNR da IDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens a cor.

Como a descodificação não depende somente do bloco da IDCT, a escolha do número de bits a utilizar neste bloco, bem como a truncatura a utilizar, é realizada através da análise conjunta com o bloco Quantificação Inversa (secção 3.4) e IDCT.

3.4. Quantificação Inversa

O bloco de Quantificação Inversa realiza apenas uma multiplicação, tal como pode ser observado a partir de (18). De forma semelhante ao bloco de Quantificação, também neste caso é necessário considerar as constantes que foram extraídas do bloco da IDCT (ver secção **Erro! A origem da referência não foi encontrada.**). Como a presença de parte decimal é inevitável, são também analisadas as limitações que o número de bits a utilizar na nova matriz de Quantificação Inversa impõem. Devido à presença de parte inteira, será necessário utilizar 9 bits para representar esta mesma parte. Assim, o número máximo de bits a utilizar para representar a parte decimal será 23 e o mínimo será 0.

Na Figura 25 representam-se as SNR obtidas para as imagens de teste utilizadas, quando variado o número de bits na matriz de Quantificação Inversa. Ao analisar esta figura, verifica-se que a partir de 3 bits a diferença para o patamar máximo obtido (SNR max) é mínima.

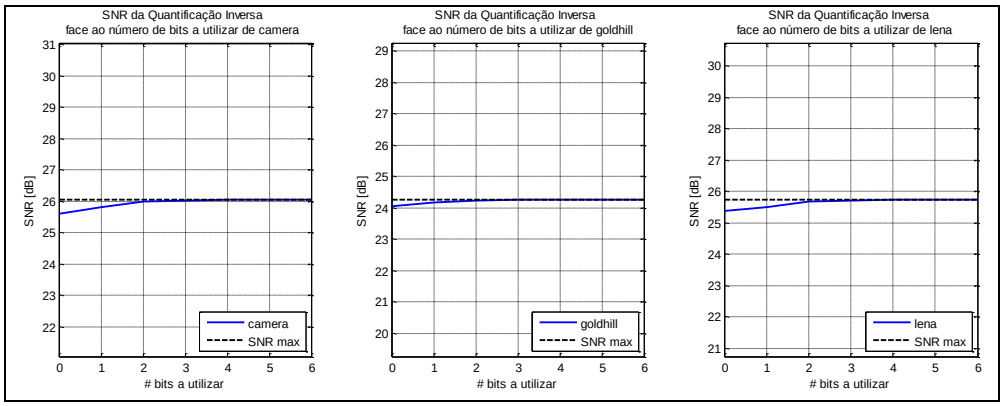


Figura 25 – SNR da Quantificação Inversa desenvolvida em função do número de bits a utilizar para imagens monocromáticas.

Tal como nos blocos anteriores, existe a limitação de truncar o número de bits a utilizar. Na Figura 26 representam-se as SNR para truncatura completa e truncatura até à 5.ª e 10.ª casas decimais.

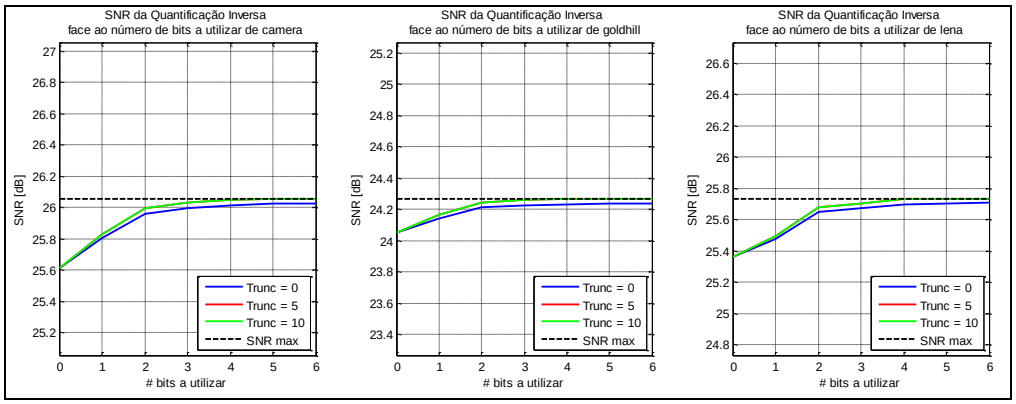


Figura 26 – SNR da Quantificação Inversa desenvolvida com truncatura total e truncatura até à 5.ª e à 10.ª casa decimal para imagens monocromáticas.

Na Figura 26 verifica-se que a partir da truncatura até à 5.ª casa decimal a diferença obtida para o patamar máximo é praticamente nula, utilizando 4 ou mais bits. De modo a apurar qual a truncatura e o número de bits a utilizar no sistema a desenvolver, foi realizado um estudo mais pormenorizado, para que este não degrade a qualidade da imagem descodificada de forma perceptivel pelo SVH e para que seja o mais próximo aos valores das SNR máximas (SNR max) obtidas para cada uma das imagens monocromáticas. Os impactos provocados pela truncatura podem ser consultados na Figura 27. Através desta verifica-se quais os valores das SNR desde a truncatura completa, até à truncatura de valores até à 5.ª casa decimal.

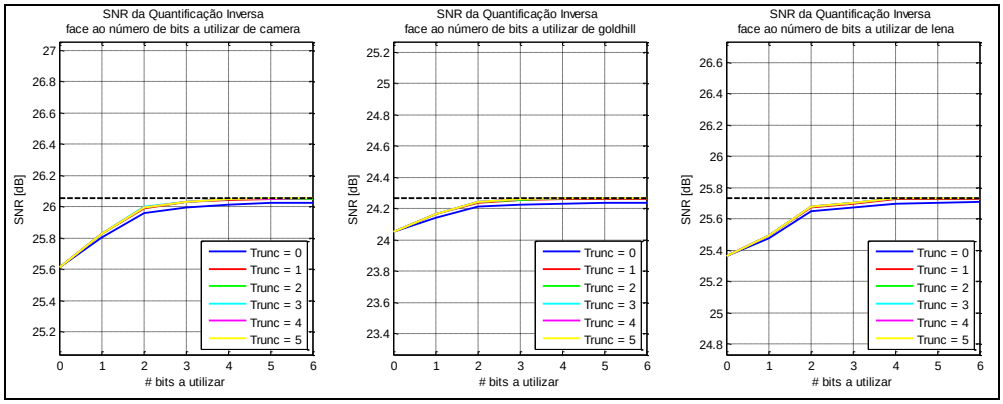


Figura 27 – SNR da FDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens monocromáticas.

Na Figura 27, para uma truncatura acima da 1.ª casa decimal, a diferença a partir da utilização de 4 bits deixa de ser notória face ao patamar máximo obtido, tal como afirmado anteriormente. No entanto, com a utilização de 7 bits a diferença da SNR obtida face à SNR máxima (SNR max) de cada imagem é bastante reduzida, sendo em alguns casos ligeiramente superior, tal como se verifica através da Tabela 17.

camera		Bits Quantificação			*goldhill*	Bits Quantificação			*lena*	Bits Quantificação		
		5	6	7		5	6	7		5	6	7
Truncatura	0	26,025	26,024	26,026	0	24,236	24,236	24,237	0	25,703	25,706	25,708
	1	26,051	26,051	26,056	1	24,261	24,262	24,270	1	25,724	25,728	25,732
	2	26,053	26,050	26,052	2	24,265	24,264	24,265	2	25,729	25,731	25,730
	3	26,055	26,055	26,054	3	24,267	24,266	24,267	3	25,732	25,730	25,730
	4	26,053	26,055	26,055	4	24,267	24,268	24,265	4	25,731	25,733	25,730
	Máx	26,057			Máx	24,267			Máx	25,732		

Tabela 17 – Valores das diferentes SNR, face ao número de bits e truncatura a utilizar na matriz de Quantificação Inversa, com representação de 3 casas decimais para imagens monocromáticas.

Para as imagens a cor as conclusões retiradas para as imagens monocromáticas são semelhantes, tal como se pode verificar na Figura 28.

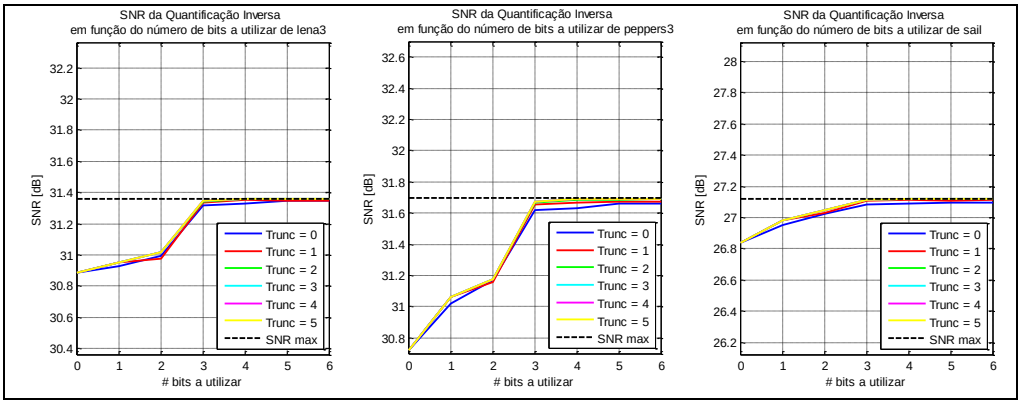


Figura 28 – SNR da FDCT desenvolvida com truncatura total e truncatura até à 5.ª casa decimal para imagens a cor.

Recorrendo à Tabela 18 verifica-se que com 7 bits, a diferença que existe face à SNR máxima obtida por cada imagem é reduzida.

lena3		Bits Quantificação			*peppers3*	Bits Quantificação			*sail*	Bits Quantificação		
		5	6	7		5	6	7		5	6	7
Truncatura	0	31,344	31,345	31,346	0	31,663	31,663	31,666	0	27,096	27,096	27,099
	1	31,349	31,348	31,361	1	31,673	31,674	31,688	1	27,110	27,111	27,123
	2	31,359	31,359	31,360	2	31,688	31,689	31,690	2	27,121	27,122	27,121
	3	31,360	31,359	31,362	3	31,692	31,693	31,695	3	27,121	27,121	27,122
	4	31,360	31,360	31,360	4	31,693	31,694	31,692	4	27,121	27,122	27,122
	Máx	31,362			Máx	31,695			Máx	27,123		

Tabela 18 – Valores das diferentes SNR, face ao número de bits e truncatura a utilizar na matriz de Quantificação Inversa, com representação de 3 casas decimais para imagens a cor.

Através da Tabela 17 e da Tabela 18 realiza-se uma escolha mais assertiva em relação ao número de bits e à truncatura a utilizar na matriz. Ao analisar as truncaturas desde a 1.ª até à 4.ª casa decimal, verifica-se que a diferença entre cada SNR pouco varia, independentemente do número de bits considerado (a variação máxima em valor absoluto é inferior a 0,01dB). Para truncatura completa, a diferença já é maior face às restantes truncaturas. Com estas duas afirmações, a escolha da truncatura a utilizar no bloco da Quantificação Inversa recai sobre a truncatura até à 1.ª casa decimal. No que ao número de bits diz respeito, serão utilizados 7 bits para representar a parte decimal de cada elemento, devido aos resultados serem bastante próximos (variações inferiores a 0,01 dB) face aos valores da SNR obtidos para cada imagem (SNR max).

Com o bloco da Quantificação Inversa definido falta apenas realizar o estudo integrando este bloco e o bloco da IDCT. Sabendo que serão utilizados no máximo 10 bits e truncatura até à 5.ª casa decimal, foi realizado um estudo para determinar qual o número de bits e a truncatura mais indicados para o bloco da IDCT. Na Figura 29 e na Figura 30 apresentam-se as SNR obtidas para as imagens de teste utilizadas, em função do número de bits da truncatura no bloco da IDCT.

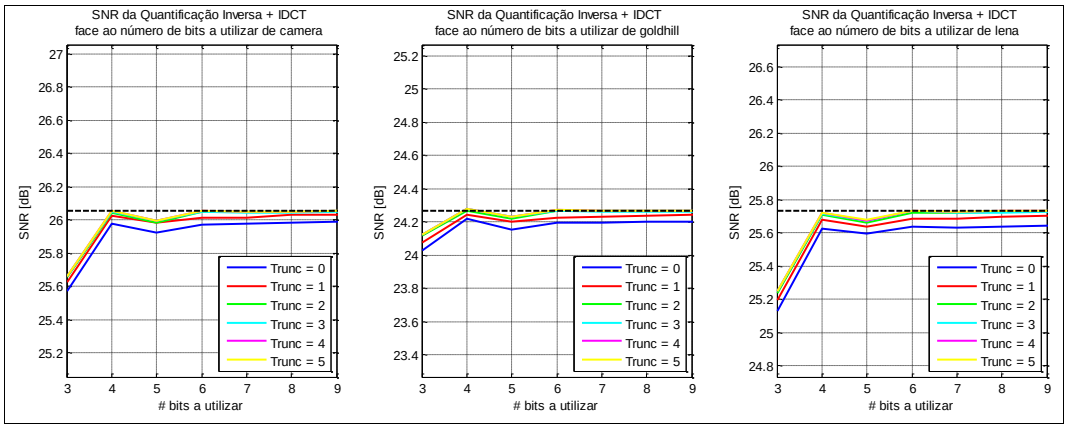


Figura 29 – SNR dos blocos da Quantificação e IDCT desenvolvidos, em função do número de bits a utilizar nos co-senos e variando a truncatura do bloco da IDCT para imagens monocromáticas.

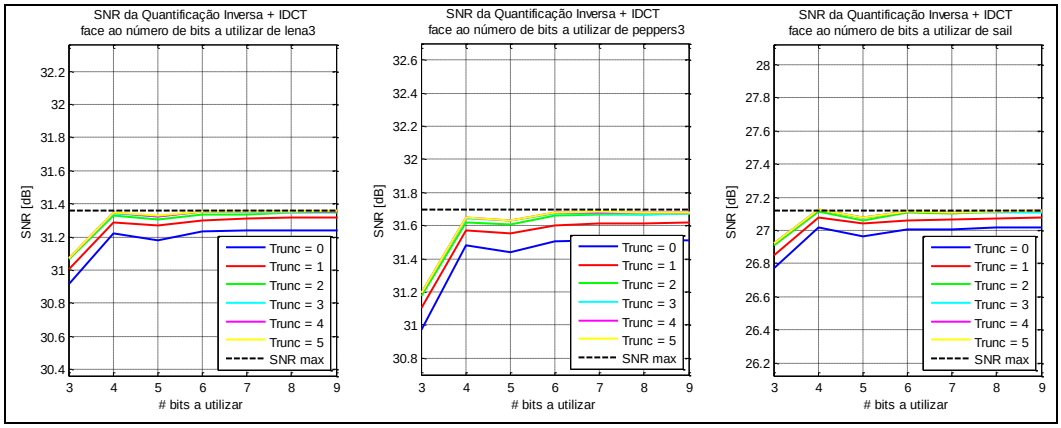


Figura 30 – SNR dos blocos da Quantificação e IDCT desenvolvidos, em função do número de bits a utilizar nos co-senos e variando a truncatura do bloco da IDCT para imagens a cor.

Os valores da SNR obtidos para o par Quantificação Inversa e IDCT são bastante próximos dos valores da SNR máxima (SNR max). Na Tabela 19 e na Tabela 20 encontram-se discriminados os valores das SNR para o número de bits a utilizar nos co-senos e a truncatura a aplicar aos produtos do bloco da IDCT. Através desta, conclui-se que ao utilizar 8 bits e truncatura completa, a diferença entre os valores da SNR dos blocos desenvolvidos face aos valores da SNR máxima (SNR max) é mínima (inferior a 0,1dB). Assim, a escolha recai sobre estes valores.

camera		Bits co-senos			*goldhill*	Bits co-senos			*lena*	Bits co-senos		
		7	8	9		7	8	9		7	8	9
Truncatura	0	25,978	25,987	25,993	0	24,198	24,203	24,204	0	25,633	25,636	25,643
	1	26,016	26,031	26,034	1	24,228	24,238	24,241	1	25,686	25,698	25,702
	2	26,048	26,049	26,056	2	24,264	24,264	24,266	2	25,719	25,727	25,729
	3	26,046	26,047	26,050	3	24,261	24,261	24,263	3	25,721	25,722	25,727
	4	26,048	26,052	26,056	4	24,265	24,266	24,269	4	25,724	25,731	25,734
	Máx	26,057			Máx	24,267			Máx	25,732		

Tabela 19 – Valores das diferentes SNR, face ao número de bits e truncatura a utilizar no bloco da IDCT, com representação de 3 casas decimais para imagens monocromáticas.

lena3		Bits Quantificação			*peppers3*	Bits Quantificação			*sail*	Bits Quantificação		
		7	8	9		7	8	9		7	8	9
Truncatura	0	31,238	31,239	31,240	0	31,513	31,509	31,509	0	27,008	27,016	27,017
	1	31,311	31,317	31,318	1	31,616	31,616	31,617	1	27,064	27,074	27,076
	2	31,338	31,346	31,347	2	31,666	31,670	31,672	2	27,103	27,111	27,114
	3	31,350	31,348	31,350	3	31,675	31,670	31,675	3	27,105	27,108	27,110
	4	31,352	31,354	31,356	4	31,681	31,680	31,680	4	27,109	27,114	27,117
	Máx	31,362			Máx	31,695			Máx	27,123		

Tabela 20 – Valores das diferentes SNR, face ao número de bits e truncatura a utilizar no bloco da IDCT, com representação de 3 casas decimais para imagens a cor.

3.5. Sistema desenvolvido

Nas secções anteriores foram apresentados os resultados dos vários estudos realizados com vista à obtenção de um sistema que implementasse um CODEC JPEG optimizado. Os resultados destes estudos permitem enumerar algumas condições que poderão ser consideradas para a implementação do CODEC em FPGA. As condições focam-se na truncatura de valores após o cálculo dos produtos e no número de bits a utilizar para codificar a parte decimal de cada elemento pré-calculado, i.e., valores dos co-senos e das tabelas de Quantificação e Quantificação Inversa. Os valores considerados para cada elemento encontram-se representados na Tabela 21.

	FDCT	Q	IQ	IDCT
Truncatura	0	-	1	0
Nº bits	6	11	7	8

Tabela 21 – Considerações de cada bloco a implementar, referentes ao número de bits para representação da parte decimal e truncatura.

Com estas considerações e através da Tabela 22 e da Tabela 23 verifica-se que existe uma ligeira diferença entre os valores das SNR obtidas no sistema proposto face ao sistema implementado usando o MATLAB. No que diz respeito às imagens naturais, verifica-se que a diferença entre o sistema implementado no MATLAB e o sistema proposto é inferior a 0,2dB. No entanto existem casos em que a SNR é ligeiramente superior, tal como no caso das imagens "*camera*" ou "*slope*". Esta melhoria na SNR deve-se ao efeito descrito na secção **Erro! A origem da referência não foi encontrada.**, em que o erro provocado pelos arredondamentos resulta numa melhoria da SNR, i.e. o erro deixa de ser destrutivo e pode

ser encarado como construtivo. Para as imagens não naturais, o sistema desenvolvido apresenta igualmente uma variação negativa de aproximadamente 0,5dB para o pior caso. Nas restantes, verifica-se uma variação positiva no máximo de 1dB. Assim, com a realização do sistema, a diferença para uma imagem, utilizando o sistema desenvolvido em MATLAB ou o sistema proposto, é imperceptível ao SVH, tal como ilustrado, na Figura 31 e na Figura 32.

Imagem	SNR		Imagem	SNR	
	Sistema MATLAB	Sistema desenvolvido		Sistema MATLAB	Sistema desenvolvido
Bird	32,67317	32,55084	**Horiz**	28,00729	27,59418
Bridge	22,91603	22,86822	**Lena**	25,73171	25,66964
Camera	26,05653	26,08536	**Montage**	28,36343	28,46581
Circles	26,87683	27,31311	**Slope**	34,65596	34,86972
Crosses	14,75260	15,74534	**Squares**	Inf	Inf
Goldhill	24,26650	24,20604	**Text**	21,49284	21,74175

Tabela 22 – SNR máxima de cada imagem para representação o sistema ideal e o CODEC desenvolvido em MATLAB (representação até à 5.ª casa decimal) para imagens monocromáticas.

Imagem	SNR	
	Sistema MATLAB	Sistema desenvolvido
Lena3	31,36205	31,21293
Monarch	30,33477	30,17002
Peppers 3	31,69453	31,48328
Sail	27,12250	26,97582
Tulips	29,72421	29,52004

Tabela 23 – SNR máxima de cada imagem para representação o sistema ideal e o CODEC desenvolvido em MATLAB (representação até à 5.ª casa decimal) para imagens a cores.

Figura 31 – Comparação da qualidade entre a imagem original (esquerda), imagem JPEG (centro) e imagem após o sistema desenvolvido (direita) para a imagem monocromática *lena*.

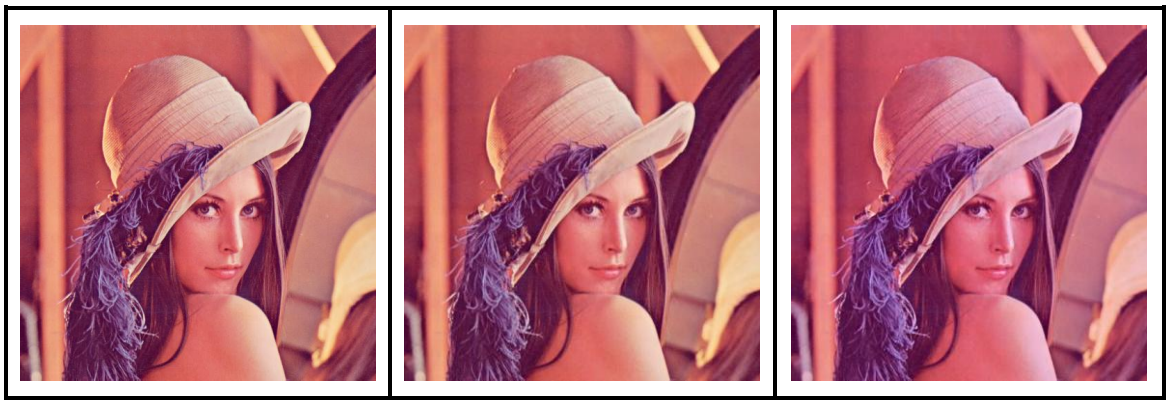


Figura 32 – Comparação da qualidade entre a imagem original (esquerda), imagem JPEG (centro) e imagem após o sistema desenvolvido (direita) para a imagem a cores *lena3*.

4. Implementação

O capítulo anterior focou-se no estudo dos blocos que constituem o CODEC JPEG. A partir desta análise foi possível verificar quantos bits são necessários para representar a parte decimal dos dados e qual a precisão da truncatura a aplicar em cada elemento após a multiplicação, de modo a que a qualidade da imagem seja semelhante à obtida através da implementação em MATLAB. Neste capítulo será estudada a implementação dos blocos estudados, numa FPGA Virtex-4 XC4VSX25 [40], cuja frequência máxima de relógio do sistema onde se encontra inserida é de 100 MHz..

As FPGAs são compostas por blocos de entrada e saída, também conhecidos por *Input/Output Block* (IOB), por elementos lógicos, denominados por *Configuration Logical Blocks* (CLB), e por malhas de integração. Os IOB são as interfaces externas entre os CLB e os circuitos externos à FPGA. Por sua vez, os CLB, são elementos que contêm circuitos lógicos. As malhas de interligação são responsáveis pelas ligações entre os IOB e o CLB e entre cada CLB. Um exemplo da estrutura das FPGAs encontra-se ilustrada na Figura 33.

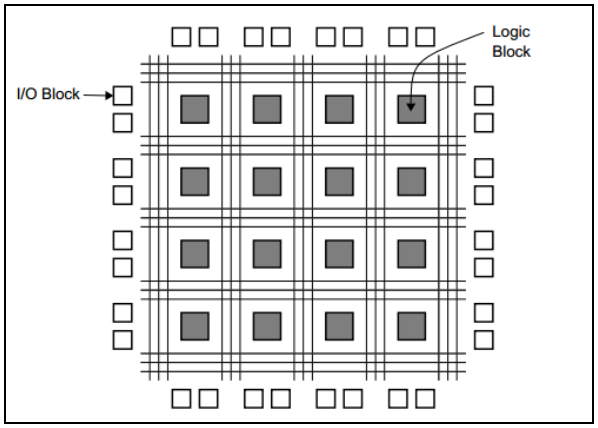


Figura 33 – Estrutura de uma FPGA (retirado de [38]).

Um CLB é composto por quatro slices, em que cada uma dessas slices contém: duas LookUp Tables (LUT) de quatro entradas, bloco lógico que contém células de armazenamento que são utilizadas para implementar funções lógicas, em que cada célula é capaz de armazenar um único valor lógico: zero ou um; dois flip-flops de quatro entradas, edge-triggered do tipo D (circuito síncrono de armazenamento) ou latches (circuito assíncrono de armazenamento); e unidades aritméticas que proporcionam operações de soma e subtracção optimizadas [40, 41]. Por sua vez, a FPGA utilizada dispõe elementos dedicados, DSP48, que realizam operações de multiplicação, multiplicação e adição, somador de três entradas, entre outros [42].

Antes de realizar a implementação é necessário definir as características de cada sistema a desenvolver, codificador e descodificador, cuja representação se encontra na Figura 34. As amostras de entrada e saída de ambos são compostas por 8 bits, sendo a entrada do codificador e a saída do descodificador representadas em código binário natural, e a saída do codificador e entrada do descodificador representadas em complemento para dois. Ambos os sistemas têm de ser capazes de notificar quando já se encontram a processar os

64 coeficientes (sinal *busy* com valor lógico '1') e quando o processamento se encontra completo (sinal *coef* com valor lógico '1'). Do mesmo modo que o sistema notifica o exterior, é também necessário notificar o sistema quando existem novas amostras prontas a processar (sinal *start* com valor lógico '1'). A leitura e escrita de amostras, de entrada e saída, são realizadas, mediante os sinais de controlo *busy*, *start* ou *coef*, a cada transição ascendente de relógio (sinal *clk*). Como os sistemas possibilitam o processamento de blocos de crominância e luminância, tem que existir um sinal que notifique qual dos dois irá ser processado, sinal *YUV*. Quando este apresenta valor lógico '0', as amostras a processar são relativas à luminância, sendo o caso contrário relativo à crominância. Por último terá de existir a possibilidade de reiniciar o sistema, sinal *reset*, repondo os seus valores iniciais.

Os sinais de entrada e saída dos blocos a desenvolver encontram-se representados na Figura 34.

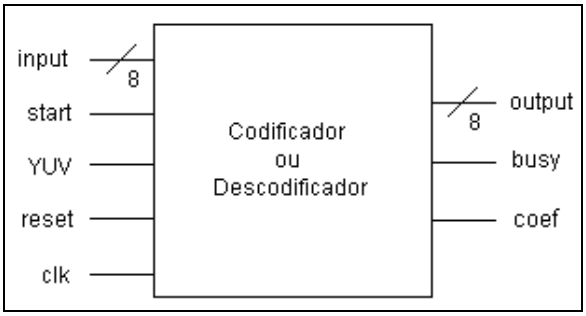


Figura 34 – Bloco do codificador/descodificador a desenvolver.

Nas secções que se seguem será apresentada a estrutura utilizada para cada bloco constituinte do codificador e descodificador, com base nas conclusões retiradas da secção 3. Para implementar cada um dos blocos foi utilizada a ferramenta da Xilinx ISE Design Suite 13.1 [36]. O comportamento de cada bloco e o comportamento do sistema a implementar foram verificados recorrendo à ferramenta de simulação ISim 13.1 da Xilinx [37].

4.1. Forward Discrete Cosine Transform *e sua inversa*

Tal como foi referido na secção 1.1, existem estruturas de hardware para rápida execução das operações relativas às transformadas. Muitas dessas estruturas focam-se na realização do cálculo da transformada a duas dimensões (2D) fazendo uso da sua propriedade da separabilidade. Com esta característica calcula-se a transformada a 2D com recurso à aplicação da transformada a uma dimensão (1D) às linhas e posteriormente às colunas, ou vice-versa, tal como apresentado na secção 2.1.1. Com este tipo de consideração reutiliza-se a estrutura responsável pelo cálculo da DCT 1D, permitindo assim a redução do hardware necessário para a implementação da DCT 2D. Para realizar esta operação existem duas estruturas conhecidas: *Multiply-Accumulate* (MAC) [33] e a estrutura em *butterfly* proposta por Lee [23].

Na estrutura MAC, representada na Figura 35, as amostras são introduzidas sequencialmente (*input*) e, a cada iteração, é aplicado o produto entre a constante dos co-senos, previamente calculados e inseridos numa memória (ROM), com a amostra de

entrada. Após a multiplicação, é adicionado o valor do produto ao valor presente no registo. Após *N* iterações, o valor do registo, que contém o coeficiente calculado, é colocado na saída (*output*). Para o caso apresentado neste trabalho, *N* tem valor 8.

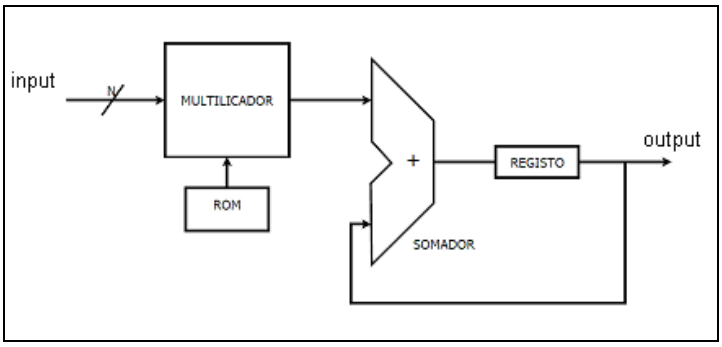


Figura 35 – Exemplo da estrutura MAC.

Por sua vez, a estrutura apresentada por Lee, representada na Figura 36, é uma estrutura paralelo-paralelo, i.e., o cálculo de cada coeficiente da DCT a uma dimensão é realizado em simultâneo. Com este requisito, são necessárias 8 amostras na entrada do bloco (x_i) para que seja iniciado o cálculo da DCT 1D. Na saída, são colocados os coeficientes da DCT 1D, após 8 iterações, tal como na estrutura MAC.

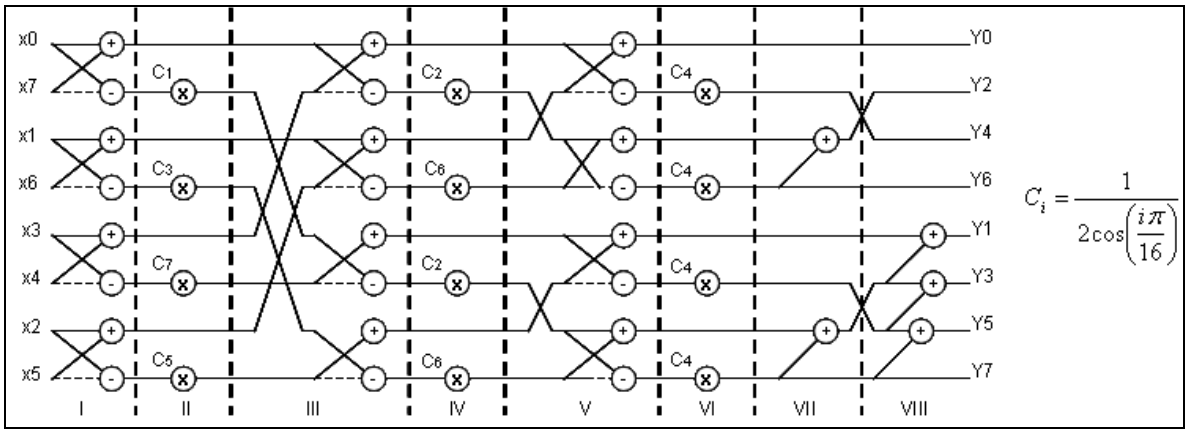


Figura 36 – Exemplo da estrutura da DCT 1D apresentada por Lee.

Analisando a Figura 36 verifica-se que a estrutura é constituída por 8 estágios (de I a VIII) desde a entrada até à saída do sistema. Em cada estágio é realizada uma operação de soma ou subtracção (as somas encontram-se representadas pela intersecção de dois traços contínuos enquanto que as subtracções encontram-se representadas pela intersecção de um traço contínuo com um tracejado horizontal). Nos estágios II, IV e VI são realizadas multiplicações com as constantes C_i. No total são necessárias 12 multiplicações e 29 somas e subtracções[6] para o cálculo dos coeficientes. No entanto, com a introdução desta estrutura, os erros obtidos diferem face aos resultados obtidos na secção 3. Apesar das diferenças nos valores das SNR, verifica-se que a variação máxima entre o CODEC a implementar em hardware e o sistema implementado em MATLAB (CODEC JPEG) é de

[6] as somas e subtrações apresentam o mesmo peso computacional

apenas 0,5dB para as imagens naturais, tal como se observa através da Tabela 24 e da Tabela 25.

Imagem	SNR						
	Codificador desenvolvido	Codificador a implementar	Descodificador desenvolvido	Descodificador a implementar	CODEC desenvolvido	CODEC a implementar	CODEC JPEG
Bird	32,65668	32,64181	32,53942	32,21323	32,55084	32,15507	32,67311
Bridge	22,90237	22,89630	22,87438	22,75746	22,86822	22,72490	22,91607
Camera	26,03520	26,02633	26,09479	25,97426	26,08536	25,94266	26,05652
Circles	26,84632	26,83278	27,30770	27,24532	27,31311	27,28280	26,87661
Crosses	14,69813	14,67650	15,76843	15,71198	15,74534	15,69678	14,7526
Goldhill	24,25438	24,25097	24,20252	24,05566	24,20604	24,03286	24,26651
Horiz	28,00729	27,68125	27,33137	26,67960	27,59418	26,60706	27,97132
Lena	25,71068	25,70402	25,67916	25,48933	25,66964	25,44740	25,73156
Montage	28,32683	28,31200	28,43952	28,26580	28,46581	28,25354	28,36344
Slope	34,60380	34,57110	34,95661	34,63970	34,86972	34,53585	34,65583
Squares	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Text	21,43583	21,42519	21,81389	21,77466	21,74175	21,71583	21,49287

Tabela 24 – SNR máxima de cada imagem monocromática obtida através do codificador e descodificador desenvolvidos em MATLAB e do codificador e descodificador a implementar (representação até à 5.ª casa decimal).

Imagem	SNR						
	Codificador desenvolvido	Codificador a implementar	Descodificador desenvolvido	Descodificador a implementar	CODEC desenvolvido	CODEC a implementar	CODEC JPEG
Lena3	30,99397	31,34316	31,23887	31,00983	31,21293	31,24031	31,36205
Monarch	30,08857	30,30847	30,21040	29,94521	30,17002	30,32698	30,33477
Peppers 3	31,15054	31,65786	31,50917	31,22097	31,48328	31,73632	31,69453
Sail	27,04280	27,10568	27,01583	26,77518	26,97582	26,97430	27,12250
Tulips	29,49416	29,70132	29,55740	29,22305	29,52004	29,87638	29,72421

Tabela 25 – SNR máxima de cada imagem a cores obtida através do codificador e descodificador desenvolvidos em MATLAB e do codificador e descodificador a implementar (representação até à 5.ª casa decimal).

Devido à necessidade de realizar duas DCT 1D para se obter a FDCT ou a IDCT, foi realizada uma análise para verificar qual a estrutura que permite obter um melhor compromisso no hardware utilizado face ao tempo dispendido para o cálculo da DCT 2D. A estrutura proposta por Lee necessita de 8 amostras para iniciar o cálculo da DCT 1D, tal como foi referido anteriormente. Após 8 iterações de relógio coloca à saída os 8 coeficientes. Para a estrutura MAC ter igual desempenho, no cálculo da DCT 1D, é necessário replicar esta estrutura 8 vezes, levando a um aumento do número de acumuladores e multiplicadores a utilizar. No entanto, o número de recursos a utilizar é inferior face à estrutura de Lee (8 acumuladores e 8 multiplicadores face aos 29 acumuladores e 12 multiplicadores).

No que ao tempo de processamento dos coeficientes diz respeito, a estrutura proposta por Lee, por apresentar uma estrutura que permite o processamento em *pipeline*, necessita de 16 ciclos de relógio para calcular os 64 coeficientes das 8 DCT 1D (assumindo que os conjuntos de 8 amostras são colocadas sequencialmente à entrada). Para a estrutura MAC replicada, por apresentar uma característica série-paralelo, são necessárias 64 iterações para calcular 64 coeficientes. Face ao tempo dispendido e à reduzida diferença entre os recursos de hardware necessários para implementar a DCT 1D, a escolha para realizar o cálculo da

FDCT recai sobre a estrutura proposta por Lee. Nesta solução, os 64 coeficientes da FDCT são obtidos após 33 iterações de relógio (16 por cada DCT 1D e 1 iteração dispendida para a transposição linha/coluna).

De modo a realizar a transposição linha/coluna é utilizada uma estrutura de registos, em que, tanto a escrita como a leitura nos registos, é realizada em paralelo. No total existem 8 colunas, cada uma delas composta por 8 registos, perfazendo no total 64 registos que guardam os 64 coeficientes do macrobloco, tal como se observa na Figura 37.

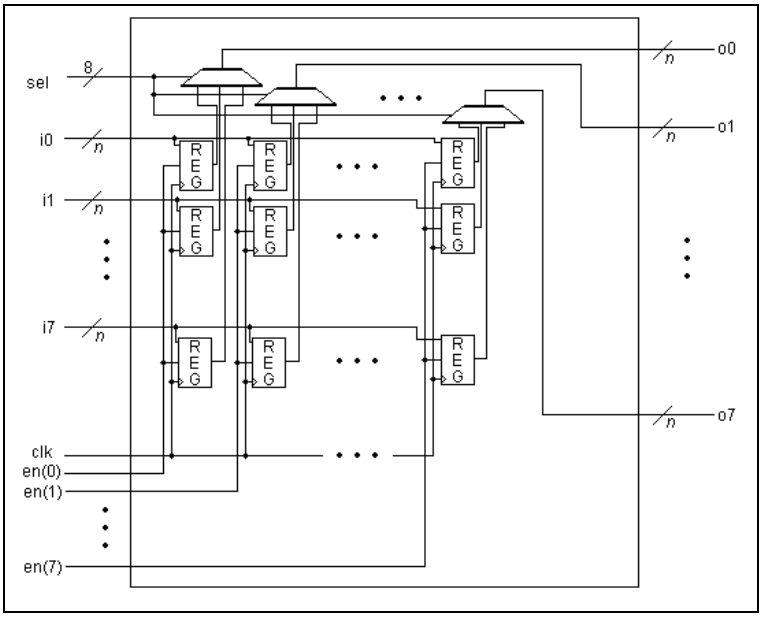


Figura 37 – Arquitectura da estrutura de rotação utilizada para o cálculo da DCT 2D.

Através da Figura 37 observa-se que a escrita nos registos é controlada pelo sinal *en* que consiste num vector de 8 bits, em que cada bit controla a coluna de registos a ser seleccionada. Cada coluna conta com um *multiplexer* que, através de um sinal de controlo, *sel*, define qual o registo cujo conteúdo será disponibilizado na saída do bloco.

4.1.1. FDCT

Como foi descrito anteriormente, o bloco da FDCT é realizado à custa de uma estrutura re-alimentada que calcula a DCT 1D primeiro às linhas e posteriormente às colunas. Apesar da primeira DCT 1D necessitar apenas de 8 bits por amostra à entrada, o mesmo não acontece com a segunda DCT 1D. Para se poder reutilizar a estrutura da FDCT é necessário adaptar a entrada aos valores de saída da primeira DCT 1D. Após verificar em MATLAB quais os valores máximos obtidos após a primeira DCT 1D (foram utilizadas as imagens de teste presentes no Anexo A), constata-se que a estrutura tem que ter 11 bits por símbolo de modo a ser possível a sua reutilização. Consequentemente, os registos presentes na estrutura de transposição linha/coluna são constituídos por 11 bits. No que à saída diz respeito, esta conta com 14 bits por coeficiente.

O comportamento do bloco depende não só das amostras, mas também de alguns sinais que se tornam imprescindíveis para ler ou escrever os dados a processar ou já processados,

respectivamente. O sinal de entrada *ready* tem como função notificar o sistema sempre que à entrada estejam presentes novas amostras. Por sua vez, o sinal *busy* informa que o sistema se encontra a processar as 64 amostras. O sinal *coefs* tem como objectivo comunicar a presença de coeficientes à saída do bloco. A caracterização das entradas e saídas do bloco da FDCT encontra-se ilustrada na Figura 38.

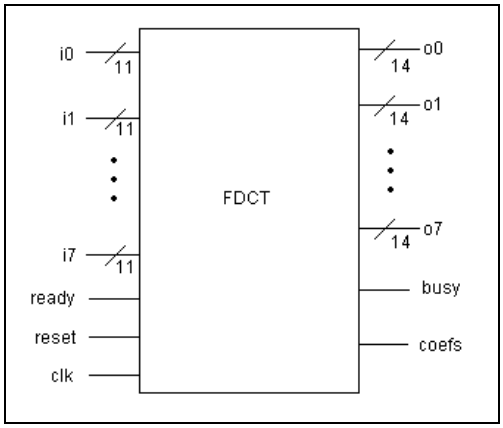


Figura 38 – Bloco da FDCT.

Tal como foi referido anteriormente, a estrutura da DCT 1D é composta por 8 estágios. Como tal, é necessário averiguar o número de bits que são utilizados em cada estágio. Recorrendo novamente às imagens do Anexo A, obteve-se os valores presentes na Tabela 26 através da ferramenta MATLAB.

coeficiente / estágio	1	2	3	4	5	6	7	8
1	12	12	12	12	12	12	12	12
2	12	12	12	12	12	12	14	12
3	13	13	13	13	14	13	14	12
4	13	13	12	13	14	13	13	12
5	14	13	14	13	14	14	13	13
6	14	12	14	12	14	13	13	12
7	14	12	13	12	14	13	14	12
8	14	14	13	12	12	12	12	12

Tabela 26 – Número de bits usados para representar a parte inteira, no cálculo da FDCT.

De forma a uniformizar o número de bits requeridos em cada estágio para todos os coeficientes, é aplicado o valor máximo de bits obtido por estágio. Assim, o primeiro estágio usa 12 bits por símbolo, enquanto que os restantes usam 14 bits por símbolo.

O cálculo dos coeficientes é manipulado através de um sinal, *enable*, que indica qual dos estágios será processado. Caso este sinal apresente valor lógico '0' não será realizada qualquer operação (soma ou multiplicação), mantendo-se o valor do registo.

Com a estrutura da FDCT e transposição de linhas/colunas definidas é necessário integrar cada elemento num único bloco. A arquitectura do bloco da FDCT encontra-se representada na Figura 39.

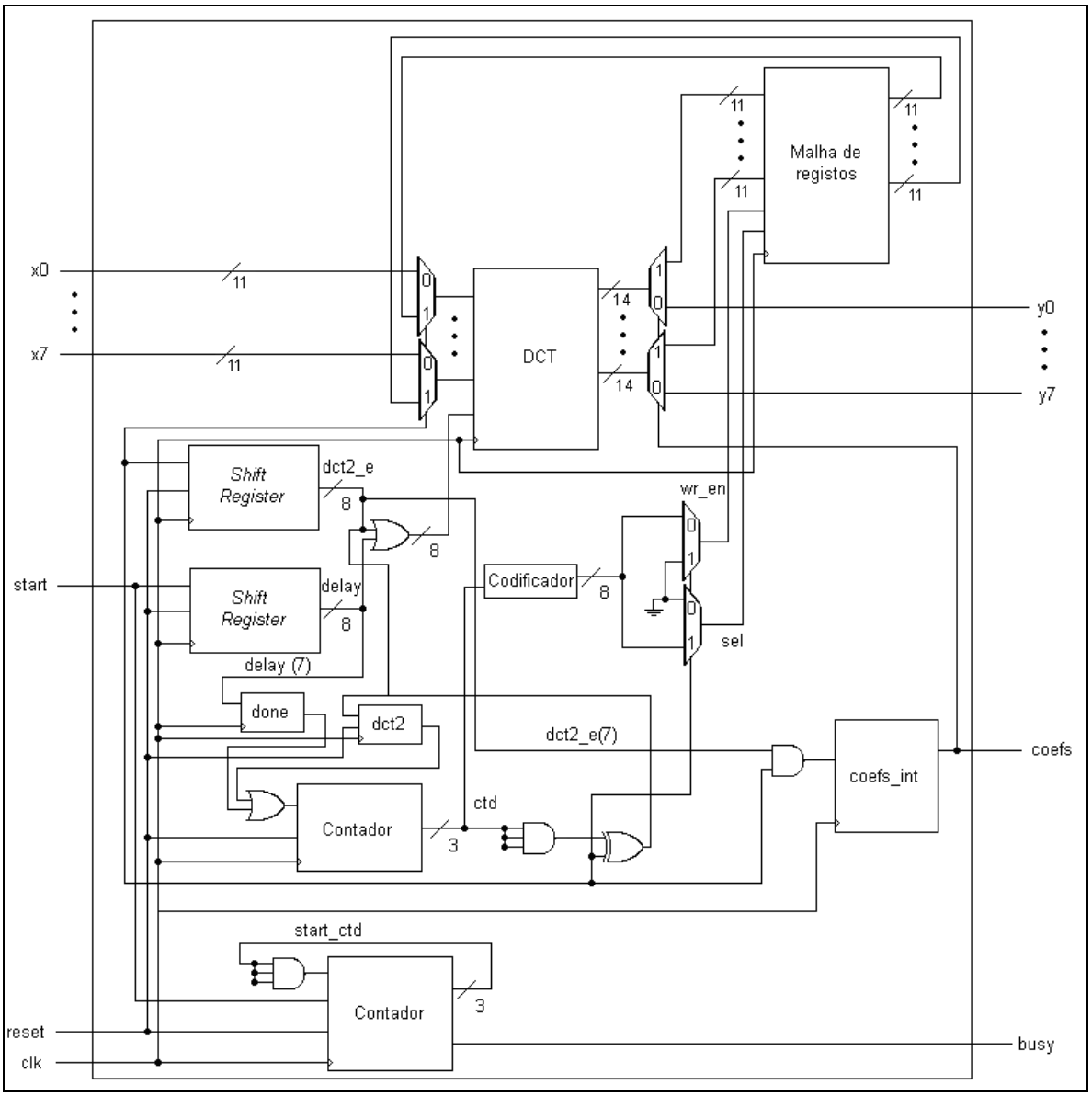


Figura 39 – Arquitectura do bloco da FDCT.

Como são necessários 64 coeficientes para calcular a FDCT, é essencial a presença de um contador que indique quantas amostras já se encontram a ser processadas (*start_ctd*). Este, ao detectar o valor lógico '1' do sinal de entrada *ready* (sinal de entrada da estrutura do bloco FDCT) incrementa a saída em uma unidade. Após 8 iterações (8 amostras por iteração) o sinal *busy* assume valor lógico '1', indicando a presença de 64 amostras a ser processadas pelo bloco. O sinal *busy* reassume o valor lógico '0' quando a estrutura se encontrar disponível para processar novas amostras.

Como a estrutura da DCT 1D é reutilizada, é necessário definir um sinal (*dct2*) que indique qual a operação que está a ser realizada, se a primeira DCT 1D ou a segunda. Dependendo do valor lógico deste (é alterado assim que é realizada cada DCT 1D), as 8 entradas da estrutura da DCT variam. Caso este apresente valor lógico '0' as entradas da

estrutura correspondem às entradas do bloco, caso contrário é a saída da estrutura de registos (matriz de transposição) que corresponde às entradas da estrutura da DCT.

Para controlar quando deve ser realizada a transposição recorre-se a um *shift register* (*delay*). Este *shift register*, que tem como entrada o sinal *ready*, permite um atraso desse mesmo sinal, de modo a que seja possível controlar a primeira DCT. Para a segunda DCT é utilizado um outro *shift register* que tem como entrada o sinal *dct2*. A cada iteração de relógio o valor dos *shift registers* é deslocado em uma posição e o bit menos significativo assume o valor presente no sinal *ready* ou *dct2*.

De modo a controlar a escrita na malha de registos, é utilizado o sinal *done*, que corresponde ao bit mais significativo do sinal *delay*. Se a cada iteração de relógio este sinal apresentar valor lógico '1', é incrementado um contador. De referir que este sinal apenas está activo quando é realizada a primeira DCT 1D. Por sua vez, o sinal proveniente do contador é descodificado para um sinal de 8 bits, que controla a coluna de escrita de ficheiros ou a leitura da linha da malha de registos, dependendo do valor lógico do sinal *dct2* (se '0' corresponde à escrita, se '1' à leitura). Como a operação da segunda DCT 1D é sequencial, i.e. as 64 amostras são colocadas à entrada da estrutura da DCT consecutivamente, o contador não necessita de qualquer sinal de controlo para ser incrementado, dependendo apenas das iterações de relógio. Assim que o contador atinge o valor máximo, i.e., 8 iterações o valor do sinal *dct2* é alterado (de '1' para '0' ou de '0' para '1'). Dependendo do valor de *dct2* são forçados valores noutros sinais de controlo. Caso apresente valor lógico '1', o sinal *busy* transita para '0', tal como o bit de menor peso do *shift register* utilizado para a segunda DCT 1D (*dct2_e*), e o sinal de saída *coefs* assume valor '1', indicando que o sistema está a colocar na saída os coeficientes calculados. Por sua vez, quando o sinal *dct2* apresente valor lógico '0', o bit de menor peso do *shift register* *dct2_e* assume valor lógico '0'. Os fluxogramas deste bloco de controlo encontram-se representados na Figura 40.

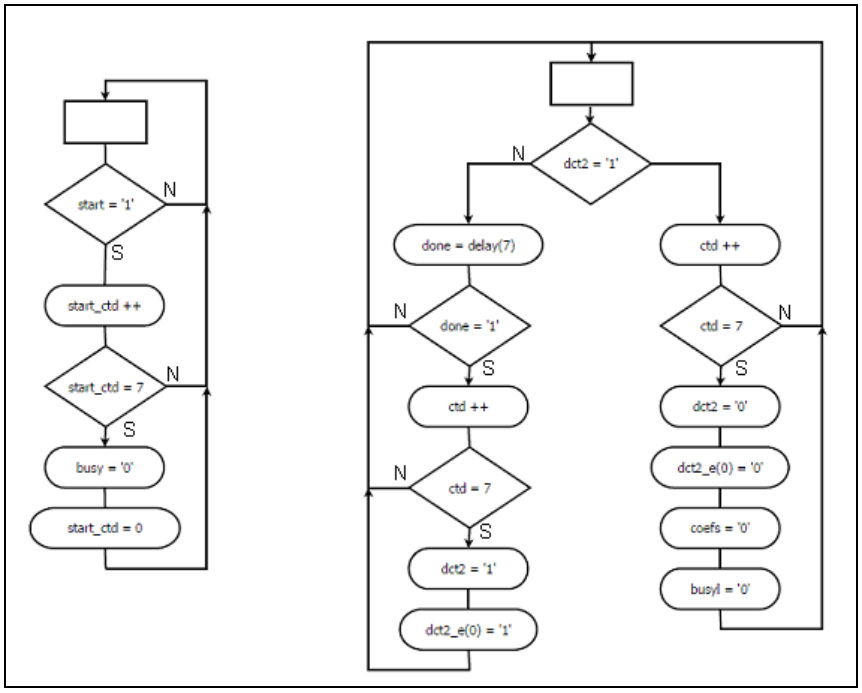


Figura 40 – Fluxogramas do bloco da FDCT.

Com a caracterização e implementação do sistema realizada, constata-se, através de simulações, que o bloco da FDCT tem o comportamento desejado, tal como se ilustra na Figura 41.

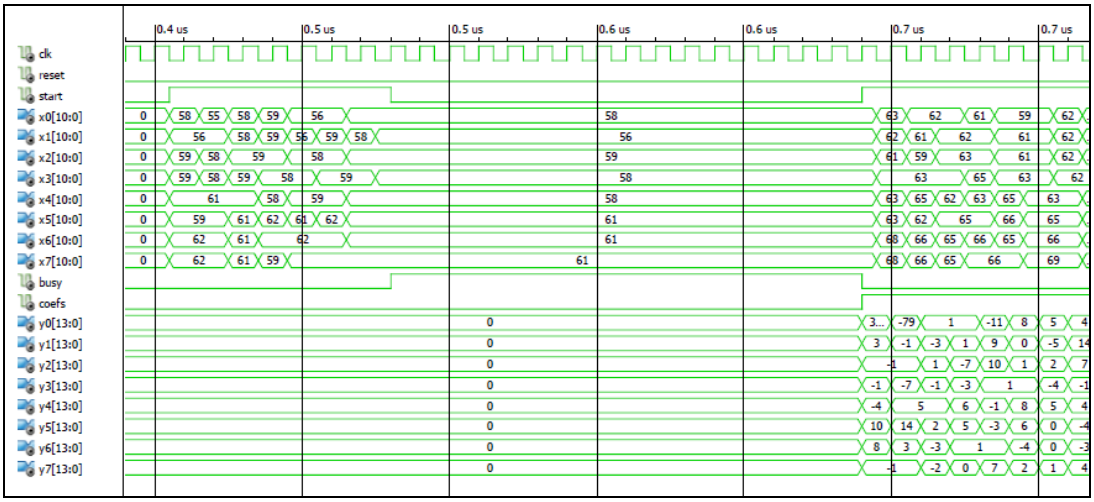


Figura 41 – Simulação realizada sobre o bloco da FDCT.

À entrada foram colocadas 8 amostras em paralelo e, com recurso ao sinal *start*, as amostras são inseridas no bloco. Após 8 iterações do sinal *start*, o sinal *busy* assume valor lógico '1', ignorando a entrada de novas amostras. Após 17 iterações (8 para o cálculo da primeira DCT, 1 para a transposição e 8 para o cálculo da segunda DCT para a primeira linha), é colocada na saída os coeficientes, activado o sinal *coefs* e desactivado o sinal *busy*.

4.1.2. IDCT

No que diz respeito ao bloco da IDCT, o processo é bastante semelhante ao que foi aplicado na FDCT. A estrutura apresentada na Figura 36 é igualmente utilizada para o bloco da IDCT, sendo apenas necessário inverter o sentido do processamento dos dados na estrutura, i.e., a entrada da estrutura é sua saída e a saída a sua entrada. São utilizados os mesmos sinais de entrada e saída, sendo apenas diferente o número de bits para as amostras (10 bits) e coeficientes (9 bits). Os restantes sinais, bem como a lógica aplicada no bloco da FDCT, mantêm-se inalterados. A caracterização das entradas e saídas do bloco da IDCT encontra-se ilustrada Figura 42.

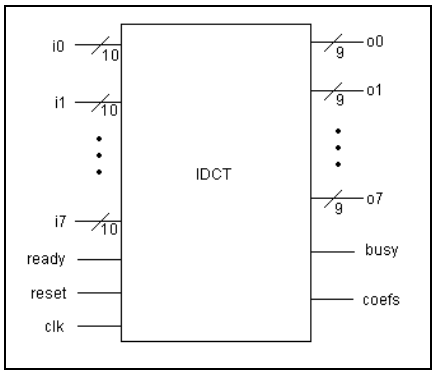


Figura 42 – Bloco da IDCT.

Em relação aos estágios, também o número de bits considerados para representar os vários símbolos difere face à FDCT, tal como se verifica através da Tabela 27. De referir que nesta estrutura foi também generalizado o número de bits por estágio, sendo os símbolos do quarto e sexto estágios representados com 10 bits e os restantes com 9. Toda a lógica de controlo associada ao bloco da FDCT foi também aplicada no bloco da IDCT (ver Figura 40).

coeficiente / estágio	1	2	3	4	5	6	7	8
1	9	9	9	9	9	9	9	9
2	9	9	9	9	9	9	9	9
3	9	8	9	8	9	8	9	9
4	9	9	9	8	9	9	10	9
5	9	9	8	9	9	9	9	9
6	9	9	9	9	10	9	7	9
7	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9

Tabela 27 – Número máximo de bits para a parte real, no cálculo da IDCT.

Tal como para a FDCT, após caracterizar e implementar o sistema realizado, constata-se que o bloco da IDCT tem um comportamento semelhante ao da FDCT, tal como se observa na Figura 43.

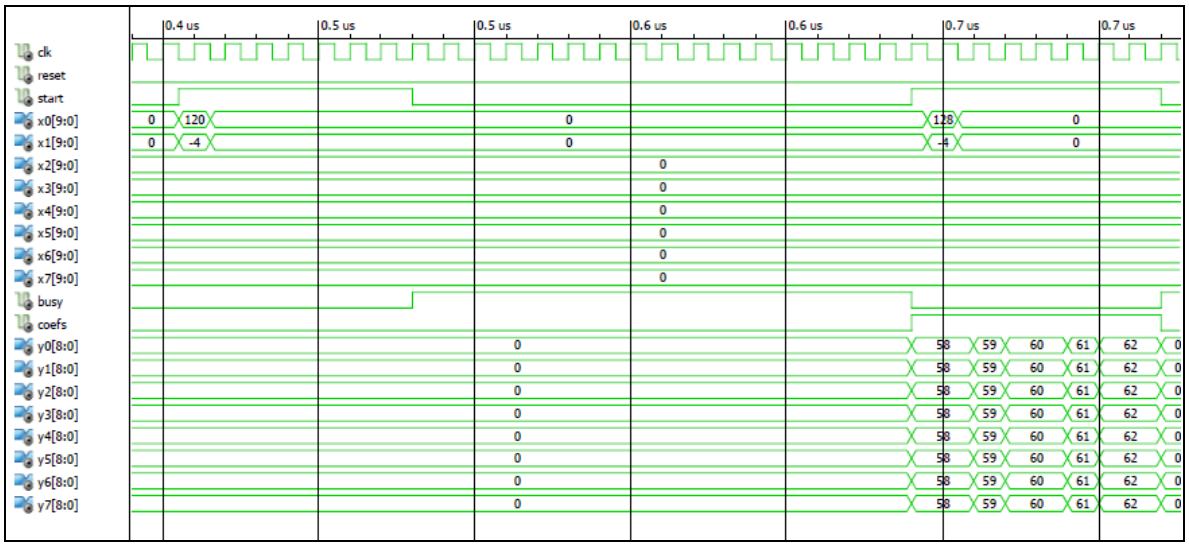


Figura 43 – Simulação realizada sobre o bloco da IDCT.

Tal como no bloco da FDCT, foram colocados à entrada 8 amostras em paralelo. As amostras são inseridas no bloco com recurso ao sinal *start* e, após 8 iterações do sinal *start*, o sinal *busy* assume valor lógico '1', ignorando a entrada de novas amostras. Após 17 iterações (8 para o cálculo da primeira IDCT, 1 para a transposição e 8 para o cálculo da segunda IDCT para a primeira linha), são colocados os coeficientes na saída do sistema, activado o sinal *coefs* e desactivado o sinal *busy*.

De salientar que na Figura 43 a entrada do bloco apresenta o dobro do seu valor real. Esta representação deve-se ao facto da utilização de uma casa decimal para representar o símbolo (a parte inteira encontra-se representada nos 9 bits de maior peso, sendo o bit menos significativo utilizado para a representação decimal).

4.2. Quantificação e Quantificação Inversa

Tal como referido nas secções 3.2 e 3.4, nos blocos da Quantificação e Quantificação Inversa são realizadas multiplicações sobre os coeficientes da FDCT ou dos valores codificados, respectivamente Assim, a arquitectura proposta para estes blocos é constituída por um bloco multiplicador e por uma ROM (*Read-Only Memory*), tal como exemplificado na Figura 44.

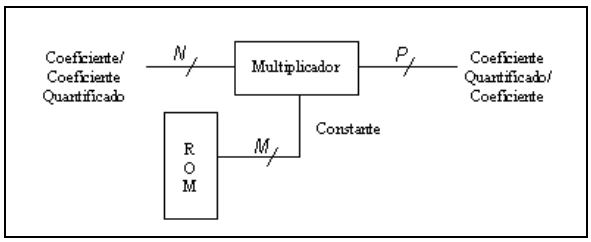


Figura 44 – Arquitectura inicial dos blocos de Quantificação e Quantificação Inversa.

4.2.1. Quantificação

O bloco da Quantificação tem como entrada os coeficientes obtidos através do bloco da FDCT. Assim, os símbolos contam com uma representação a 14 bits (*N* na Figura 44), em código de complemento para 2, e apenas parte inteira.

Recorrendo às conclusões retiradas na secção 3.2, verifica-se que os símbolos presentes na ROM são representados com 12 bits, 1 bit para a parte inteira e 11 bits para a parte decimal (*M* na Figura 44).

Após verificar a representação simbólica dos mesmos, observa-se a presença de um número fixo de bits a '0' à esquerda. Como o sistema conta com uma implementação de vírgula fixa (ver secção 3), reduzi-se o tamanho dos símbolos, anulando a maioria dos bits que apresentam valor nulo à esquerda.. Assim, o tamanho dos símbolos é reduzido de 12 para 7 bits (é sempre garantido o bit da parte inteira, sendo os restantes utilizados para a parte decimal).

Com esta alteração é necessário adaptar também o produto resultante da multiplicação. Se num primeiro caso o produto era constituído por 26 bits, neste novo caso, o produto conta apenas com 21 bits. No entanto, os 11 bits menos significativos contêm a informação referente à parte decimal e, como tal, podem ser ignorados, devido à operação de arredondamento para o inteiro mais próximo. Apesar de na saída o produto apresentar um símbolo com 10 bits, alguns destes podem ser descartados sem que daí resulte perda de informação. Para a codificação de entropia, é considerado para este trabalho a utilização de 8 bits para representar os coeficientes quantificados (ver secção 2.3). Assim, constata-se que os 2 bits mais significativos podem ser ignorados, sendo a saída constituída por símbolos com 8 bits. Do símbolo obtido no produto composto por 21 bits, apenas são considerados 8 bits (*P* na Figura 44).

Tal como referido no início da secção 4, as amostras são alimentadas sequencialmente no sistema. Com esta consideração e, sabendo que o bloco da FDCT necessita de um intervalo de tempo considerável entre a entrada das amostras e o cálculo dos coeficientes, a realização deste bloco recorre a apenas um multiplicador. No que à ROM diz respeito, é apenas utilizado um circuito que armazena os valores de todas as constantes. Nos primeiros 64 endereços encontram-se as constantes referentes à luminância e nos restantes 64 as constantes referentes às crominâncias.

Uma das características da saída do bloco da FDCT é a disposição dos coeficientes calculados em paralelo. Com esta particularidade é necessário que o bloco da Quantificação disponibilize uma estrutura que permita o armazenamento temporário dos coeficientes gerados para que possam ser posteriormente processados. A saída desta estrutura apresenta, sequencialmente, os coeficientes previamente guardados, permitindo que seja utilizado apenas um multiplicador, tal como pretendido. A arquitectura da estrutura é composta por um total de 64 registos de 14 bits e um *multiplexer* de 64 entradas, tal como se verifica na Figura 45. A escrita nos registos é realizada em paralelo, através de um sinal que indica a coluna que será escrita. A leitura é realizada com recurso ao *multiplexer* de 64 entradas, que coloca na saída o registo seleccionado por um sinal constituído por 6 bits.

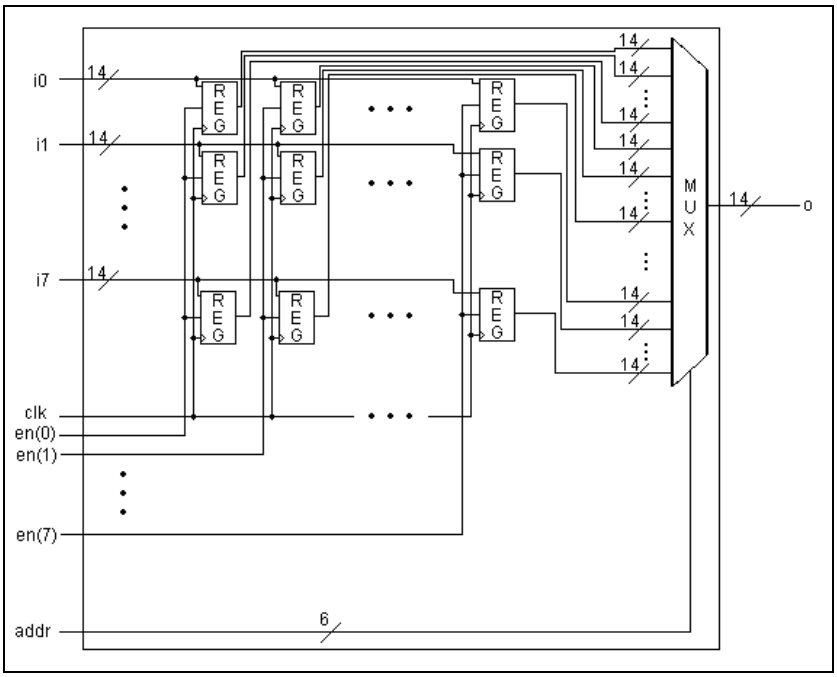


Figura 45 – Arquitectura da estrutura de armazenamento do bloco da Quantificação.

Assim, à arquitectura inicial do bloco da Quantificação é adicionada a estrutura de registos. A escrita das amostras de entrada na malha de registos é realizada com recurso a um *shift register* e um descodificador. Para cada impulso de relógio, *clk*, é verificado o nível lógico do sinal de entrada *start*. Caso este apresente valor lógico '1', o registo presente no shift register é deslocado em uma casa para a esquerda e no bit de menor peso é colocado o valor lógico '1'. Caso contrário, nada é realizado. O descodificador recebe o sinal proveniente do shift register e activa na saída o bit correspondente à sequência de bits com valor lógico '1' (só pode haver uma sequência e tem que começar no bit menos significativo), tal como exemplificado na Tabela 28.

Entrada do descodificador	Saída do descodificador
00000001	00000001
00000011	00000010
00000111	00000100
00001111	00001000
00011111	00010000
00111111	00100000
01111111	01000000
11111111	10000000

Tabela 28 – Descodificação do sinal proveniente do *shift register* do bloco de Quantificação.

Assim que são escritas as primeiras amostras, o sistema encontra-se pronto para realizar o cálculo dos coeficientes quantificados. Para tal é utilizado um registo, *delay*, que impõe o atraso de uma iteração (um ciclo de relógio, *clk*) ao sinal *start*. Este atraso permite que o valor presente na ROM seja descarregado para um registo, de modo a realizar o produto entre este e a amostra. Após a multiplicação é activado o sinal que indica a possibilidade de leitura desse mesmo registo (sinal *wr*). O endereçamento de leitura dos registos da malha é realizado através dos 6 bits de menor peso do sinal proveniente de um contador (*addr*). Este sinal é igualmente utilizado no endereçamento da ROM, em conjunto com o sinal de

entrada *YUV*. O contador incrementa o sinal *addr* a cada iteração de relógio caso o registo *delay* apresente valor lógico '1'. O 7º bit do sinal *addr* apresenta um comportamento semelhante ao sinal *reset*. Assim que este assuma valor lógico '1', o contador, o registo do *shift register*, o registo *delay* e o resultado da multiplicação assumem os valores iniciais de sistema. O diagrama de blocos do circuito implementado encontra-se representado na Figura 46 e os fluxogramas encontram-se representados na Figura 47

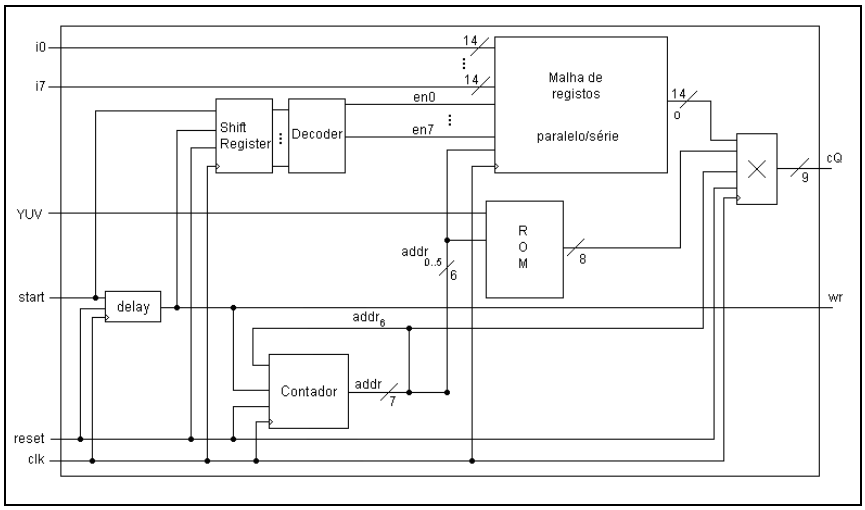


Figura 46 – Arquitectura do bloco da Quantificação.

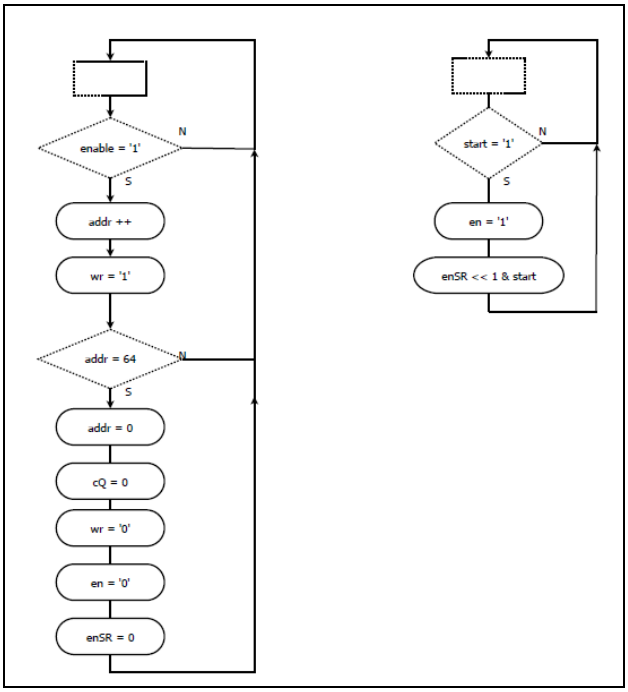


Figura 47 – Fluxogramas do bloco da Quantificação.

A Figura 48 ilustra a simulação realizada de modo a validar o comportamento do bloco da Quantificação.

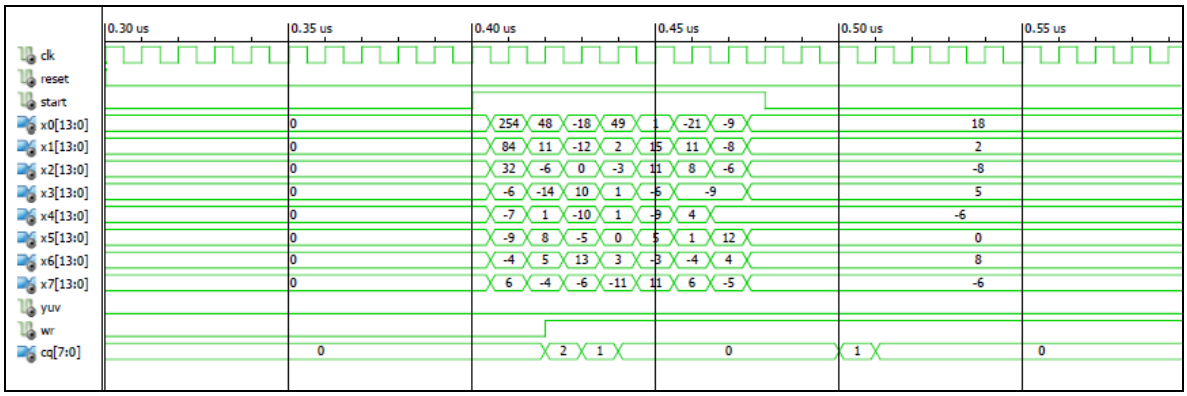


Figura 48 – Simulação realizada sobre o bloco da Quantificação.

À entrada do bloco de Quantificação foram colocados 8 coeficientes em paralelo (obtidos através da simulação do bloco da FDCT) e, com recurso ao sinal *start* (sinal *coefs* do bloco FDCT), as amostras são inseridas no bloco. Assim que é inserido o primeiro conjunto de coeficientes é realizada a multiplicação e colocada na saída na iteração seguinte positiva de relógio e activado o sinal *wr*. A cada iteração positiva de relógio é colocado o novo coeficiente quantificado à saída.

4.2.2. Quantificação Inversa

O bloco da Quantificação Inversa tem como entrada os coeficientes quantificados. Como foi constatado no bloco da Quantificação, os símbolos contam com uma representação simbólica de 8 bits com sinal e apenas parte inteira.

As constantes são representadas através de símbolos com 13 bits, em que 6 pertencem à parte inteira e 7 à parte decimal (*M* da Figura 44), tal como descrito na secção 3.4. Neste bloco não se realiza qualquer optimização no que diz respeito às constantes, tal como foi realizado no bloco da Quantificação, devido à presença de parte inteira na representação das constantes.

O produto obtido entre os coeficientes quantificados e o valor presente na ROM é pois representado com um máximo de 21 bits. De modo a seguir as especificações descritas na secção 3.4, os 7 bits menos significativos do resultado podem ser ignorados por representarem a parte decimal do produto. Dos restantes 14 bits, os 9 bits menos significativos são suficientes para representar os valores dos coeficientes.

Tal como referido no início da secção 4, os valores dos coeficientes quantificados são lidos sequencialmente. Com esta consideração a realização deste bloco recorre a apenas um multiplicador. O número de endereços da ROM a utilizar é semelhante à utilizada no bloco da Quantificação. No total, esta ROM dispõe de 128 endereços, onde nos primeiros 64 endereços constam as constantes referentes à luminância e, nos restantes 64 as constantes referentes à crominância.

Como a IDCT necessita de 8 coeficientes para iniciar o cálculo, é necessário implementar uma estrutura que seja capaz de armazenar os dados para posteriormente serem processados. Esta estrutura é composta por 8 registos de 10 bits cada.

Tal como no bloco da Quantificação, é necessário integrar a estrutura de registos com o bloco de Quantificação Inversa. De modo a endereçar a memória são utilizados os 6 bits menos significativos do sinal *addr*, em conjunto com o sinal *YUV*. O sinal *addr* é proveniente de um contador, que incrementa o seu valor sempre que a cada impulso de relógio, *clk*, o sinal *enable* apresente valor lógico '1'. O contador e o valor do produto são reiniciados quando o 7º bit de *addr* ou o sinal *reset* apresentam valor lógico '1'. O fluxograma do módulo de controlo do bloco da Quantificação Inversa encontra-se representado na Figura 49.

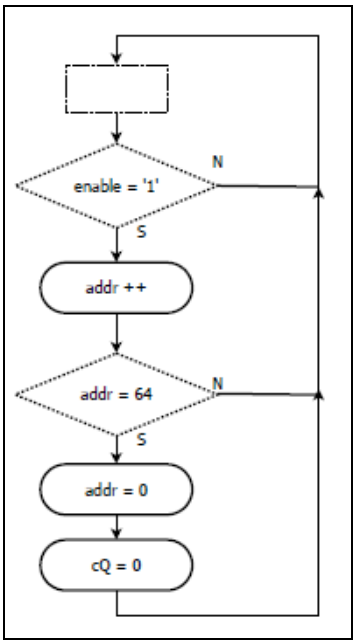


Figura 49 – Fluxograma do bloco da Quantificação Inversa.

A escrita nos registos é realizada com recurso a um *multiplexer*. Este é controlado pelos 3 bits menos significativos do sinal *addr*. Após a escrita nos 8 registos é activado o sinal *ready*. Este sinal notifica o bloco da IDCT que existem coeficiente a serem processados. Assim que são calculados os 64 coeficientes, o sinal *busy* é activo. Este sinal corresponde ao 7º bit de *addr*.

O diagrama de blocos do circuito implementado encontra-se representado na Figura 50.

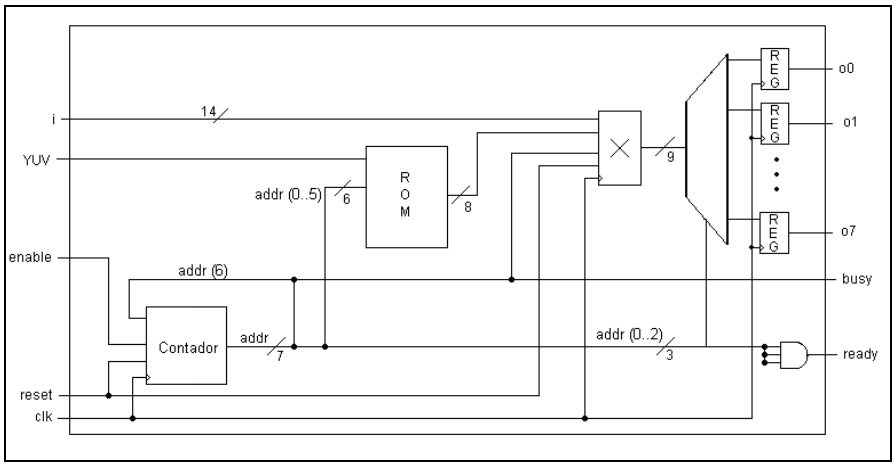


Figura 50 – Arquitectura do bloco da Quantificação Inversa.

Com a caracterização e implementação do sistema realizada, constata-se que, através de simulações, o bloco da Quantificação Inversa tem o comportamento desejado, tal como se verifica na Figura 51. De referir que a saída é representada por 10 bits (9 bits para a parte inteira e 1 bit para a parte decimal) e, como tal, encontra-se representado com o dobro do seu valor, i.e., sem o formato de vírgula fixa.

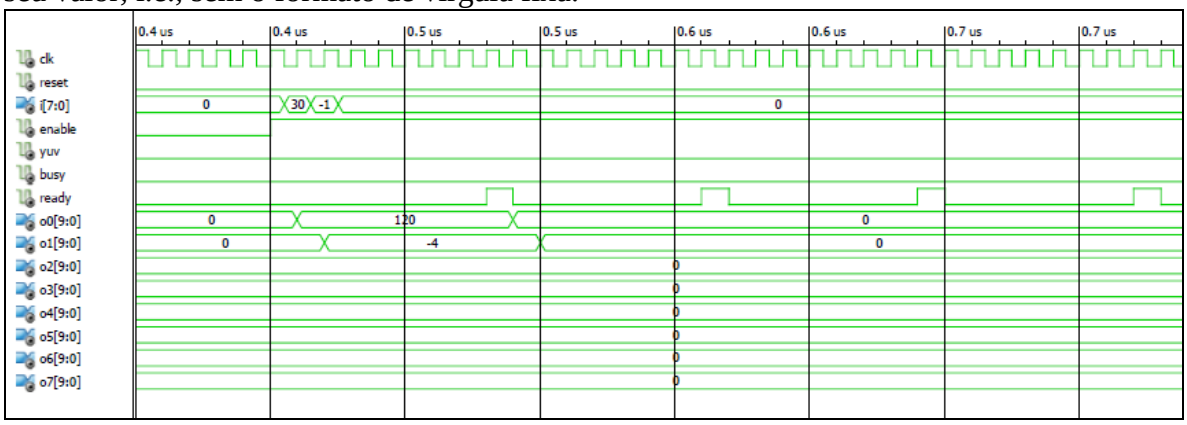


Figura 51 – Simulação realizada sobre o bloco da Quantificação Inversa.

Da Figura 51 constata-se que o bloco de Quantificação recebe um coeficiente sempre que, a cada iteração de relógio, esteja presente o valor lógico ‘1’ no sinal *enable*. Após 8 iterações é activado o sinal *ready* que indica quando estão presentes na saída, e em paralelo, os coeficientes a serem processados pelo bloco da IDCT. Após 64 iterações o sinal *busy* fica activo por um ciclo de relógio de modo a reiniciar todo o processo.

4.3. Codificador – Descodificador

Após a implementação dos blocos constituintes do codificador e descodificador, é necessário integrar cada um na estrutura final para que esta siga as especificações definidas no início desta secção.

4.3.1. Codificador

Segundo as especificações do codificador, as amostras de entrada são introduzidas no sistema em série. Como o bloco da FDCT necessita de 8 amostras em paralelo para iniciar o cálculo de uma operação de transformada, é necessário implementar uma estrutura que seja capaz de armazenar temporariamente os dados para posteriormente serem processados. Esta estrutura, que se encontra representada na Figura 52, é composta por 8 registos de 11 bits cada, à semelhança do que foi utilizado no bloco da Quantificação Inversa.

No funcionamento do sistema, o sinal *start* indica a presença de nova amostra. Sobre essa amostra é realizada uma operação de subtracção por 128 (descrita na secção 2) e seguidamente inicia-se um contador (*ctdBuffer*). O valor obtido através da subtracção é escrito num registo (i_x). Por cada 8 amostras carregadas é activado o sinal *startFDCT* que corresponde ao sinal *ready* do bloco da FDCT, e portanto inicia o cálculo da operação de transformada. O sinal *YUV* é carregado para um registo após a entrada das 64 amostras, de modo a evitar que haja qualquer alteração do mesmo durante a codificação da imagem.

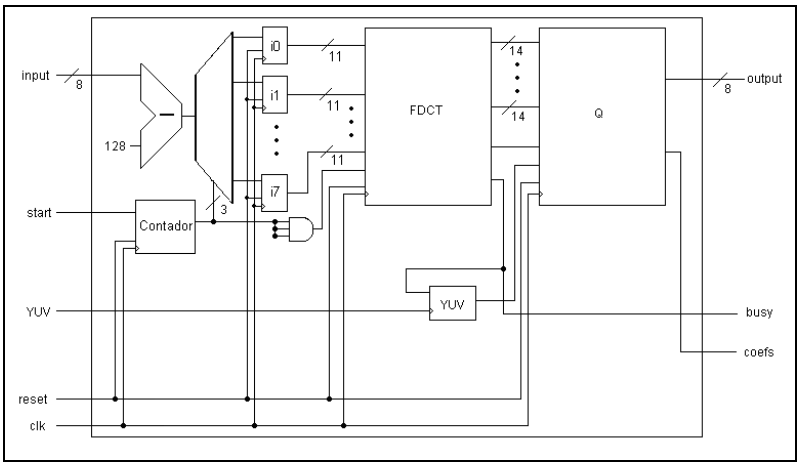


Figura 52 – Arquitectura do bloco de codificação.

Na Figura 53 verifica-se que os *pixels*, em código binário natural, são lidos para o sistema sempre que o sinal *start* apresente valor lógico '1'. Após a leitura de 64 amostras o sinal *busy* fica activo e só retoma o valor lógico '0' quando se encontram calculados os 64 coeficientes quantificados (ver Figura 54). Por sua vez, o sinal *coefs* indica a presença dos coeficientes quantificados à saída.

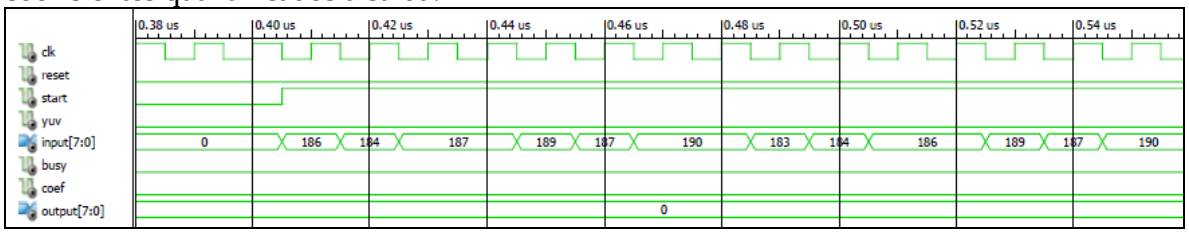


Figura 53 – Simulação realizada sobre o bloco de codificação, referente à entrada de amostras.

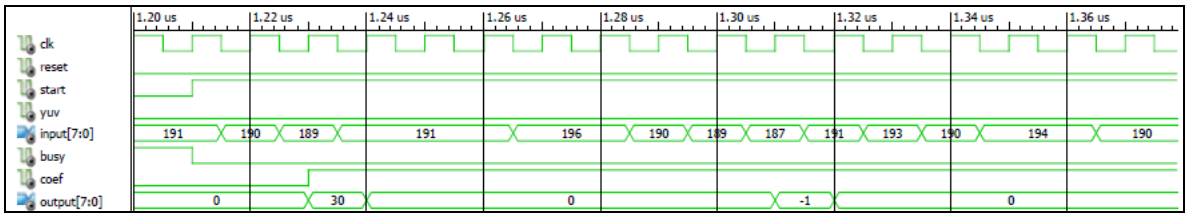


Figura 54 – Simulação realizada sobre o bloco de codificação, referente à saída de coeficientes.

Com a implementação realizada verifica-se que os resultados obtidos através do bloco de codificação coincidem com os obtidos em MATLAB. Constata-se que a frequência máxima do sinal de relógio do bloco de codificação é de 92,345MHz (820,5 µs para codificar uma imagem de 256 linhas por 256 colunas).

O circuito sintetizado consiste em 13 multiplicadores, 1 ROM, 43 acumuladores, 4 contadores, 1 *multiplexer* e 2446 registos. Estes componentes são então sintetizados para os componentes lógicos presentes na FPGA. A taxa de utilização na FPGA encontra-se descriminada na Tabela 29.

Componente lógico	Quantidade	Utilização
Slices	1896	18%
Flip-flops	2472	12%
LUT	1889	9%
DSP48	13	10%

Tabela 29 – Recursos utilizados para a realização do bloco de codificação.

Os multiplicadores são inferidos nos componentes DSP48 e os contadores, *multiplexer* e acumuladores em LUT. Os registos são inferidos em *flip-flops*.

4.3.2. Descodificador

De modo a seguir as especificações referidas anteriormente, foi necessário inserir uma estrutura semelhante à que foi utilizada no bloco da Quantificação (ver Figura 45), a jusante do bloco da IDCT, de forma a disponibilizar a saída de forma sequencial. Esta estrutura é composta por 64 registos de 9 bits e permite que a saída do descodificador seja apresentada sequencialmente face à saída em paralelo da IDCT.

Tal como no codificador, o sinal *start* indica a presença de uma amostra a ser processada pelo descodificador. Estas amostras são primeiramente processadas pelo bloco IQ e posteriormente pelo bloco IDCT. Assim que o cálculo da segunda IDCT a uma dimensão é concluído, os 8 coeficientes são colocados na estrutura de registos. Após a primeira coluna de 8 registos ser escrita, é realizada a operação de soma por 128 a cada registo. No seguimento desta operação, o sinal *coef* fica activo para assinalar que os registos de saída estão prontos para ser lidos. A arquitectura do bloco de descodificação encontra-se representada na Figura 55.

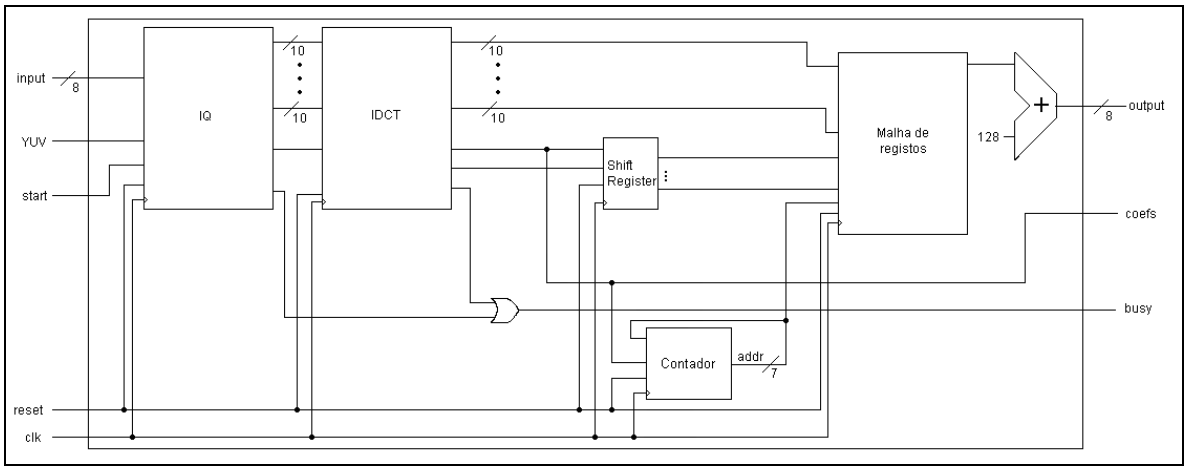


Figura 55 – Arquitectura do bloco de descodificação.

Após a integração de ambos os blocos constata-se, através de simulações, que o bloco responsável pela codificação de imagem tem o comportamento desejado, ilustrado nas Figura 56 e Figura 57. O bloco da descodificação tem comportamento semelhante face ao da codificação. À entrada são colocados os coeficientes quantificados e sempre que o sinal *start* apresente valor lógico '1', estas são lidas para o bloco. Após a leitura das 64 amostras o sinal *busy* fica activo, retomando o valor lógico '0' quando o bloco se encontra preparado para processar novas 64 amostras. O sinal *coefs* com valor lógico '1' indica a presença dos *pixels* da matriz da imagem descodificada na saída do circuito.

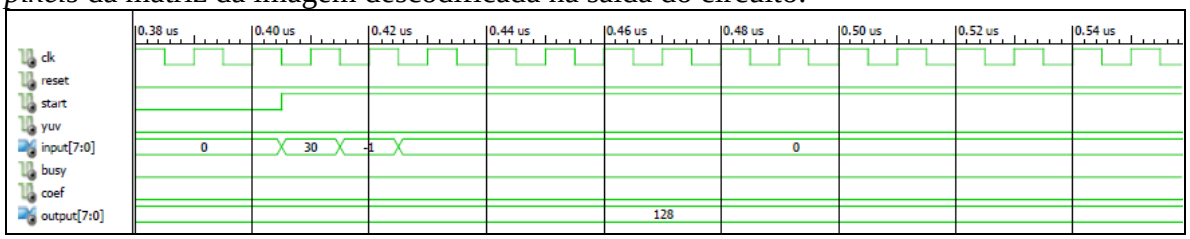


Figura 56 – Simulação realizada sobre o bloco de descodificação, referente à entrada de amostras.

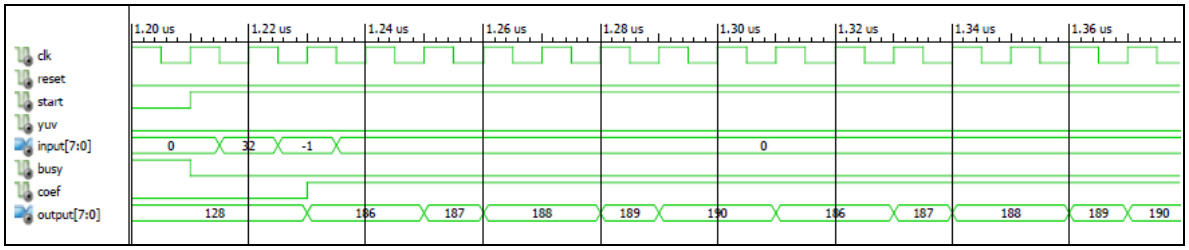


Figura 57 – Simulação realizada sobre o bloco de descodificação, referente à saída de coeficientes.

Com a implementação realizada e, após verificar que os resultados obtidos através do bloco de descodificação coincidem com os obtidos em MATLAB, verifica-se que a frequência máxima do sinal de relógio do bloco de codificação é de 97,516MHz (830,5 µs para codificar uma imagem de 256 linhas por 256 colunas). Após a síntese deste bloco verifica-se que o mesmo consiste em 13 multiplicadores, 1 ROM, 51 acumuladores, 4 contadores, 1 *multiplexer* e 1739 registos. Estes componentes são então sintetizados para os

componentes lógicos presentes na FPGA. O número de componentes e a percentagem de utilização na FPGA encontra-se descriminada na Tabela 30.

Componente lógico	Quantidade	Utilização
Slices	1399	13%
Flip-flops	1763	8%
LUT	1551	3%
DSP48	13	10%

Tabela 30 – Recursos utilizados para a realização do bloco de descodificação.

Tal como para o sistema codificador, os multiplicadores são inferidos nos componentes DSP48 e os contadores, *multiplexer* e acumuladores em LUT. Os registos são inferidos em *flip-flops*.

Comparando o bloco de descodificação com o de codificação verifica-se que o número de DSP48 a utilizar é semelhante. Tal como foi referido no início da secção 4, os DSP48 podem ser utilizados para realizar a operação de multiplicação e, tanto para o codificador como para o descodificador, são realizadas 13 multiplicações (12 para a cálculo das transformadas e 1 para as quantificações). A utilização deste elemento prende-se pelo facto de o mesmo ser uma unidade dedicada que realiza a operação de multiplicação. Assim, a utilização deste permite que o número de LUT (unidades lógicas) a utilizar seja inferior face a uma utilização sem este elemento. De modo a forçar a sua síntese é necessário recorrer à macro "*attribute use_dsp48 : string;*".

Relativamente aos restantes elementos, existem algumas diferenças devido ao número de bits utilizado em cada um dos sistemas. Os elementos mais influentes são as transformadas, como se observa na Tabela 31. Como para a IDCT são utilizados 9 ou 10 bits, por registo e por estágio, o número de elementos a utilizar para a IDCT é inferior ao número de elementos a utilizar na FDCT, onde são utilizados entre 11 a 14 bits, por registo e por estágio. Os restantes elementos são distribuídos pelos blocos de quantificação e para as malhas de registos.

Componente lógico	Quantidade (FDCT)	Quantidade (IDCT)
Slices	1072	859
Flip-flops	1442	1069
LUT	1334	1149

Tabela 31 – Recursos utilizados para a realização dos blocos da FDCT e IDCT.

5. Conclusões

O presente trabalho incidiu sobre o estudo e a implementação parcial de um CODEC JPEG. Durante o decorrer do trabalho foram analisados os blocos constituintes do sistema de codificação e descodificação, bem como as propriedades inerentes a cada um deles. Foi também apresentado um exemplo da aplicação do CODEC de modo a facilitar a percepção do mesmo.

Como resultado do estudo realizado, constatou-se que se reduzi consideravelmente o número de operações a realizar nos blocos da FDCT e IDCT, através de algumas optimizações de cálculo. A separação da DCT a duas dimensões para duas DCT a uma dimensão traduziu-se numa redução de hardware a utilizar, à custa de um aumento do tempo de cálculo da transformada a duas dimensões. O cálculo *à priori* das constantes da FDCT e IDCT e a sua integração com os blocos de Quantificação e Quantificação inversa traduziu-se na redução do número de multiplicações, influenciando apenas a representação destas matrizes.

O número de bits a utilizar para representar cada operação são pontos críticos nos sistemas digitais, onde existe um compromisso de qualidade face aos recursos ocupados. Verificou-se que, utilizando um número limitado de bits, a qualidade da imagem codificada e descodificada pouco difere. Assim, para a FDCT, a parte decimal dos co-senos é representada com 6 bits, e o resultado das operações não contém representação decimal. Para a IDCT apenas se aumenta o número de bits a utilizar para a representação dos co-senos, mantendo-se a truncatura completa. Para a Quantificação são necessários 11 bits para a representação dos valores da matriz de quantificação, enquanto que para a Quantificação Inversa são utilizados 7 bits. A truncatura completa é aplicada às operações da Quantificação e para a Quantificação Inversa é utilizada a truncatura até à primeira casa decimal. Com base no estudo realizado verifica-se, através de simulações, que os resultados obtidos diferem no máximo 0,5dB face ao CODEC ideal obtido através do MATLAB.

No decorrer deste trabalho apenas foram analisadas duas estruturas para o cálculo da DCT e IDCT a uma dimensão: estrutura de Lee e MAC. A estrutura Lee, utilizada para os blocos da FDCT e IDCT, permite que em apenas 33 iterações de relógio sejam calculados os 64 coeficientes de uma matriz de 8x8 amostras, com recurso a 12 multiplicações e 29 somas. Comparando com as operações a realizar para o cálculo da FDCT e IDCT, verifica-se uma redução superior a 1000 vezes face ao cálculo original e sem optimizações, tanto no número de multiplicações como de somas. Esta estrutura apresenta um total de 8 estágios independentes. Cada registo de cada estágio tem no mínimo 11 bits e no máximo 14, para a FDCT, e no mínimo 9 bits e no máximo 10, para a IDCT.

Os blocos de Quantificação e Quantificação Inversa foram implementados com recurso a apenas um multiplicador. Esta implementação deveu-se às especificações de entrada e saída dos blocos de codificação e descodificação (amostragem sequencial). Cada um dos operandos do multiplicador do bloco da Quantificação conta com 12 e 14 bits, constante da matriz de quantificação e coeficiente obtido da FDCT, respectivamente. Para o bloco da Quantificação Inversa são utilizados 8 e 13 bits para a representação dos coeficientes quantificados e para as constantes da matriz de quantificação inversa, respectivamente.

A quantidade de recursos de hardware necessários para a implementação de ambos os blocos é bastante semelhante. No entanto, o bloco de descodificação necessita de uma área menor face ao bloco da codificação, devido ao número de bits necessário para a implementação do bloco da descodificação ser inferior ao número de bits do bloco da codificação.

Com ambos os sistemas implementados (codificação e descodificação) utilizando o modelo de FPGA Virtex-4, codifica-se e descodifica-se uma imagem, constituída por 256 linhas e 256 colunas, em 820,5 μs e 830,5 μs, respectivamente. Como tal, o CODEC desenvolvido permite codificar/descodificar, aproximadamente, 1200 imagens monocromáticas por segundo. Com esta característica, codifica-se imagens com maior resolução ou tirar fotografias mais rapidamente, permitindo obter fotografias em sequências de imagem rápida, por exemplo, desporto.

5.1. Trabalho futuro

De modo a concluir o CODEC é necessário implementar a codificação de entropia, bem como a construção da bitstream de saída. Esta implementação poderá ser realizada com recurso a tabelas de verificação (*lookup tables*) ou através de um processador embebido na FPGA (picoBlaze).

É possível reduzir ou até mesmo eliminar os multiplicadores presentes no cálculo das transformadas. Este cálculo, por recorrer a apenas 7 constantes, pode ser realizado com recurso a *shift register* e acumuladores (estruturas *shift-and-add*). No entanto é necessário avaliar o compromisso entre o tempo dispendido na multiplicação e o tempo dispendido para a codificação e descodificação da imagem. Esta consideração é apenas valida para a estrutura proposta por Lee.

Apesar as estruturas estudadas (MAC e Lee) serem eficientes no cálculo das transformadas (devido ao número de recursos utilizados face ao tempo dispendido para cálculo), podem existir outras estruturas que apresentem características semelhantes e que necessitam de ser avaliadas. Neste contexto, salientam-se as características das estruturas de processamento sistólico com vista ao aumento do ritmo de processamento dos circuitos.

De modo a avaliar se é possível reduzir a perda inferida nas tabelas de Quantificação é necessário realizar um estudo com recurso a novas tabelas. Estas podem ser constituídas por elementos que facilitem a multiplicação e divisão de elementos, como por exemplo, elementos constituídos por potências de 2, através de deslocamentos para a esquerda ou direita, respectivamente.

Por fim, seria interessante implementar e avaliar o comportamento do CODEC na FPGA Virtex-4 de modo a validar os resultados obtidos na secção 4, resultados esses obtidos através da ferramenta de simulação ISim.

Referências

[1] – ISO/IEC 10918-1 and ITU-T Recommendation T.81. Information technology-digital compression and coding of continuous tone still images: Requirements and guidelines, 1993.

[2] – B. Girod, EE368b Image and Video Compression, Human visual perception, Stanford University.

[3] – I. Kuon, R. Tessier and J. Rose, "FPGA Architecture: Survey and Challenges", Foundations and Trends in Electronic Design Automation, ISBN 978-1-60198-126-4, April, 2008.

[4] – J. Miano, "Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP", Addison-Wesley Professional, ISBN 0201604434, August, 1999.

[5] – ISO 12639:1998 - Graphic technology -- Prepress digital data exchange -- Tag image file format for image technology (TIFF/IT).

[6] – http://www.jpeg.org/apps/photo.html, consultado a 21 de Janeiro de 2011.

[7] – http://digitaljournalist.org/issue0602/dunleavy.html, consultado a 04 de Janeiro de 2012.

[8] – RFC 1341- MIME: Multipurpose Internet Mail Extensions, June 1992

[9] – http://www.cabiatl.com/mricro/obsolete/graphics/graphics.html, consultado a 04 de Abril de 2011.

[10] – ISO/IEC 15948:2004 - Information technology -- Computer graphics and image processing -- Portable Network Graphics (PNG): Functional specification.

[11] – http://www.mathworks.com/, consultado a 21 de Janeiro de 2011.

[12] – http://www.egr.msu.edu/waves/people/Ali_files/DCT_TR802.pdf, consultado a 05 de Abril de 2011.

[13] – V. Britanak, K. R. Rao and P. Yip, "Discrete Cosine Transform: Properties, Algorithms, Advantages, Applications", Academic Press Publications, ISBN 978-0-12-373624-6, Boston, 1990.

[14] – W. Chen, C. H. Smith and S. C. Fralick: A Fast Computational Algorithm for the Discrete Cosine Transform, IEEE Trans. on Communications, Vol. COM-25, pp. 1004-1009, 1977.

[15] – V. Britanak, P. Yip, K. Rao, "Discrete Cosine and Sine Transforms General Properties, Fast Algorithms and Integer Approximations", Academic Press, ISBN 978-0123736246, November 6, 2006.

[16] – N. Kehtarnavaz, "Real-Time Digital Signal Processing: Based on the TMS320C6000", Newnes, ISBN 978-0750678308, July, 2004.

[17] – http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf, consultado a 8 de Abril de 2011.

[18] – http://www.xilinx.com/esp/aerospace-defense/index.htm, consultado a 8 de Abril de 2011.

[19] – http://www.xilinx.com/esp/wireless/index.htm, consultado a 8 de Abril de 2011.

[20] – http://www.xilinx.com/esp/broadcast/index.htm, consultado a 8 de Abril de 2011.

[21] – http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm, consultado a 8 de Abril de 2011.

[22] – http://www.mp3-tech.org/, consultado a 8 de Abril de 2011.

[23] – B. G. Lee, "A new algorithm to compute the discrete cosine transform," IEEE Trans. Acoust., Speech and Signal Process., vol. ASSP-32, pp. 1243-1245, Dececember, 1984.

[24] – D. Chelemal and K. Bo, "Fast computational algorithms for the discrete cosine transform," in ls'h Asilomar Conf. on Circuits, Systems, and Computers, (Pacific Grove, CA), November, 1985.

[25] – H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, vol. ASSP-35, pp. 1455-1461, October, 1987.

[26] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages,and Applications", ISBN 978-0125802031, Academic Press, September, 1990.

[27] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression- A survey," Proc. IEEE, vol. 83, pp. 220–246, Feb. 1995.

[28] – Carlach, J. C., Penard, P., Sicre, J. L. "TCAD: a 27 MHZ 8x8 Dicrete Cosine Transform Chip", Centre Commun d'Etudes de Telediffusion et de Telecommunications.

[29] – M. Vetterli and A. Lichtenberg, "A discrete cosine transform chip," IEEE Journal of Selected Areas in Communications,pp. 49-65, January 1986.

[30] – Wallace, G. K. "The JPEG Still Picture Compression Standard", Communications of the ACM, Vol. 34, Issue 4, pp.30-44.

[31] – Pennebaker, W.B., Mitchell, J.L., et. al. "Arithmetic coding articles". IBM J. Res. Dev., vol. 32, no. 6 (Nov. 1988), pp. 717-774.

[32] – Huffman, D.A. "A method for the construction of minimum redundancy codes". In Proceedings IRE, vol. 40, 1962, pp. 1098-1101.

[33] – Z. Mohd Yusof, I. Suleiman, Z. Aspar, "Implementation of two dimensional forward DCT and inverse DCT usingFPGA", TENCON 2000. Proceedings. 02/2000; 3:242-245 vol.3. DOI: 10.1109/TENCON.2000.892266

[34] – Ze-Nian Li, Mark S Drew, "Fundamentals of Multimedia", Prentice-Hall, ISBN 978-8120328174, 2004

[35] – Liptak, Béla G. "Instrument Engineers' Handbook: Process control and optimization, vol. 2", CRC Press, ISBN 978-0849310812, September, 2005

[36] – http://www.xilinx.com/products/design-tools/ise-design-suite/logic-edition.htm, consultado a 10 de Janeiro de 2012.

[37] – ISim User Guide, Xilinx, UG660 (v 11.3) September 16, 2009.

[38] – S. Brown, J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial", Department of Electrical and Computer Engineering, University of Toronto.

[39] – F.Pereira (editor), "Comunicações Audiovisuais: Tecnologias, Normas e Aplicações", IST Press, ISBN 978-972-8469-81-8, Julho 2009.

[40] – Virtex-4 Family Overview, Product Specification, Xilinx, DS112 (v3.1) August 30, 2010.

[41] – Virtex-4 FPGA User Guide, Xilinx, UG070 (v2.6) December 1, 2008.

[42] – XtremeDSP for Virtex-4 FPGAs, Xilinx, UG073 (v2.7) May 15, 2008.

Anexo A

Neste anexo são apresentadas as imagens utilizadas nas simulações realizadas para o sistema desenvolvido.

a) Imagens monocromáticas:

bird.gif

bridge.gif

camera.gif

circles.gif

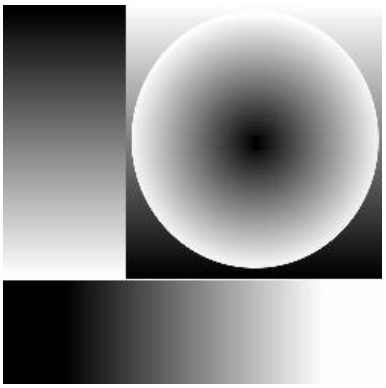
crosses.gif

goldhill.gif

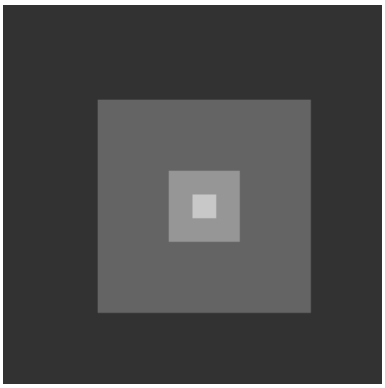
horiz.gif

lena.gif

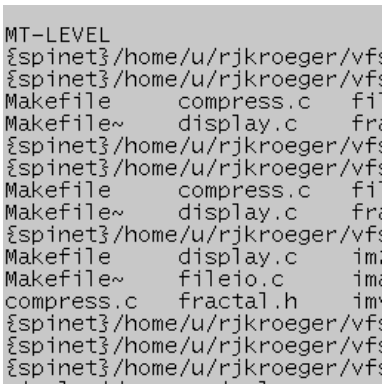
montage.gif



slope.gif

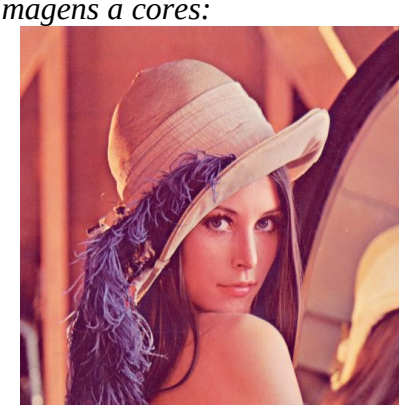


squares.gif



text.gif

b) Imagens a cores:



lena3.tif

monarch.tif

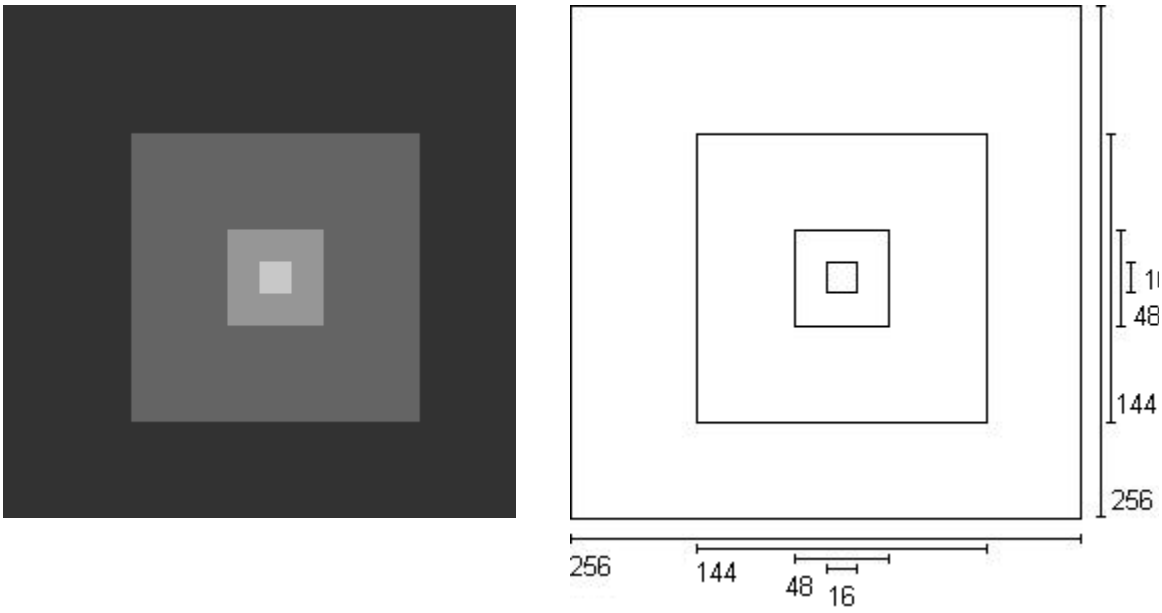
peppers3.tif

sail.tif

tulips.tif

Anexo B

Neste anexo é apresentada a particularidade inerente à imagem '*squares.gif*'.



$16/8 = 2 \Rightarrow$ É divisível por 8. Os elementos desta área contêm o valor 200.

$48/8 = 6 \Rightarrow$ É divisível por 8. Os elementos desta área contêm o valor 150.

$(48-16)/8 = 4 \Rightarrow$ É divisível por 8.

$144/8 = 18 \Rightarrow$ É divisível por 8. Os elementos desta área contêm o valor 100.

$(144-48)/8 = 12 \Rightarrow$ É divisível por 8.

$256/8 = 32 \Rightarrow$ É divisível por 8. Os elementos desta área contêm o valor 50.

$(256-144)/8 = 14 \Rightarrow$ É divisível por 8.

Aplicando a FDCT a uma matriz de cada área, confirma-se que apenas o primeiro elemento da matriz (coeficiente DC) apresenta valor não nulo, tal como se verifica no exemplo seguinte:

$$FDCT\begin{bmatrix} 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \end{bmatrix} = \begin{bmatrix} 12800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dividindo as matrizes de coeficientes pela matriz de Quantificação, através da aplicação de (15), obtém-se os coeficientes com perda. Como apenas os coeficientes DC apresentam valores não nulos, verifica-se que não existe perda inferida pela Quantificação:

$$round\left(\frac{\begin{bmatrix} 12800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}{\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 108 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}}\right) = \begin{bmatrix} 800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Aplicando a parte do descodificador à última matriz obtida, obtém-se o seguinte resultado final:

Para o primeiro coeficiente $\Rightarrow 800 \times 16 = 12800$
Para os restantes $\Rightarrow 0$

$$IDCT\begin{bmatrix} 12800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \end{bmatrix}$$

O mesmo sucede com as restantes matrizes. Este efeito deve-se à inexistência de transições na matriz de entrada. Se um conjunto de coeficientes apresentasse um valor diferente, o resultado final seria distinto.

Por exemplo, para a seguinte matriz,

$$\begin{bmatrix} 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 205 \\ 200 & 200 & 200 & 200 & 200 & 200 & 205 & 205 \\ 200 & 200 & 200 & 200 & 200 & 205 & 205 & 205 \end{bmatrix}$$

o resultado final seria diferente

$$\begin{bmatrix} 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\ 200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \end{bmatrix}$$

o suficiente para que a SNR não tendesse para ∞.

Anexo C

Neste anexo são apresentadas as tabelas típicas para a codificação de Huffman. De salientar uma vez mais que as tabelas apresentadas neste anexo, apesar de estarem referidas na norma, não devem ser consideradas como as tabelas standard. As tabelas C.1 e C.2 dizem respeito à codificação da diferença dos coeficientes DC para a luminância e crominância, respectivamente.

SIZE	Tamanho do símbolo	Símbolo 1
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Tabela C.1 – Tabela da luminância para as diferenças dos coeficientes DC.

SIZE	Tamanho do símbolo	Símbolo 1
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

Tabela C.2 – Tabela da crominância para as diferenças dos coeficientes DC.

As tabelas 3 e 4 são referentes à codificação dos coeficientes AC para a luminância e crominância, respectivamente.

RUNLENGTH/SIZE	Tamanho do símbolo	Símbolo 1
0/0	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	6	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100

Tabela C.3 – Tabela da luminância para os coeficientes AC (parte 1 de 4).

1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101
8/1	9	111111000

Tabela C.3 – Tabela da luminância para os coeficientes AC (parte 2 de 4).

8/2	14	11111111100000
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011
8/9	16	1111111110111100
8/A	16	1111111110111101
9/1	9	111111001
9/2	16	1111111110111110
9/3	16	1111111110111111
9/4	16	1111111111000000
9/5	16	1111111111000001
9/6	16	1111111111000010
9/7	16	1111111111000011
9/8	16	1111111111000100
9/9	16	1111111111000101
9/A	16	1111111111000110
A/1	9	111111010
A/2	16	1111111111000111
A/3	16	1111111111001000
A/4	16	1111111111001001
A/5	16	1111111111001010
A/6	16	1111111111001011
A/7	16	1111111111001100
A/8	16	1111111111001101
A/9	16	1111111111001110
A/A	16	1111111111001111
B/1	10	1111111001
B/2	16	1111111111010000
B/3	16	1111111111010001
B/4	16	1111111111010010
B/5	16	1111111111010011
B/6	16	1111111111010100
B/7	16	1111111111010101
B/8	16	1111111111010110
B/9	16	1111111111010111
B/A	16	1111111111011000
C/1	10	1111111010
C/2	16	1111111111011001
C/3	16	1111111111011010
C/4	16	1111111111011011
C/5	16	1111111111011100
C/6	16	1111111111011101
C/7	16	1111111111011110
C/8	16	1111111111011111
C/9	16	1111111111100000
C/A	16	1111111111100001
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	1111111111110000

Tabela C.3 – Tabela da luminância para os coeficientes AC (parte 3 de 4).

E/7	16	1111111111110001
E/8	16	1111111111110010
E/9	16	1111111111110011
E/A	16	1111111111110100
F/0	11	11111111001
F/1	16	1111111111110101
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111111

Tabela C.3 – Tabela da luminância para os coeficientes AC (parte 4 de 4).

RUNLENGTH/SIZE	Tamanho do símbolo	Símbolo 1
0/0	2	00
0/1	2	01
0/2	3	100
0/3	4	1010
0/4	5	11000
0/5	5	11001
0/6	6	111000
0/7	7	1111000
0/8	9	111110100
0/9	10	1111110110
0/A	12	111111110100
1/1	4	1011
1/2	6	111001
1/3	8	11110110
1/4	9	111110101
1/5	11	11111110110
1/6	12	111111110101
1/7	16	1111111110001000
1/8	16	1111111110001001
1/9	16	1111111110001010
1/A	16	1111111110001011
2/1	5	11010
2/2	8	11110111
2/3	10	1111110111
2/4	12	111111110110
2/5	15	111111111000010
2/6	16	1111111110001100
2/7	16	1111111110001101
2/8	16	1111111110001110
2/9	16	1111111110001111
2/A	16	1111111110010000
3/1	5	11011
3/2	8	11111000
3/3	10	1111111000
3/4	12	111111110111
3/5	16	1111111110010001
3/6	16	1111111110010010
3/7	16	1111111110010011
3/8	16	1111111110010100
3/9	16	1111111110010101
3/A	16	1111111110010110
4/1	5	11101
4/2	9	111110110
4/3	16	1111111110010111
4/4	16	1111111110011000
4/5	16	1111111110011001

4/6	16	1111111110011010

Tabela C.4 – Tabela da crominância para os coeficientes AC (parte 1 de 3).

4/7	16	1111111110011011
4/8	16	1111111110011100
4/9	16	1111111110011101
4/A	16	1111111110011110
5/1	6	111011
5/2	10	1111111001
5/3	16	1111111110011111
5/4	16	1111111110100000
5/5	16	1111111110100001
5/6	16	1111111110100010
5/7	16	1111111110100011
5/8	16	1111111110100100
5/9	16	1111111110100101
5/A	16	1111111110100110
6/1	7	1111001
6/2	11	11111110111
6/3	16	1111111110100111
6/4	16	1111111110101000
6/5	16	1111111110101001
6/6	16	1111111110101010
6/7	16	1111111110101011
6/8	16	1111111110101100
6/9	16	1111111110101101
6/A	16	1111111110101110
7/1	7	1111010
7/2	11	11111111000
7/3	16	1111111110101111
7/4	16	1111111110110000
7/5	16	1111111110110001
7/6	16	1111111110110010
7/7	16	1111111110110011
7/8	16	1111111110110100
7/9	16	1111111110110101
7/A	16	1111111110110110
8/1	8	11111001
8/2	16	1111111110110111
8/3	16	1111111110111000
8/4	16	1111111110111001
8/5	16	1111111110111010
8/6	16	1111111110111011
8/7	16	1111111110111100
8/8	16	1111111110111101
8/9	16	1111111110111110
8/A	16	1111111110111111
9/1	9	111110111
9/2	16	1111111111000000
9/3	16	1111111111000001
9/4	16	1111111111000010
9/5	16	1111111111000011
9/6	16	1111111111000100
9/7	16	1111111111000101
9/8	16	1111111111000110
9/9	16	1111111111000111
9/A	16	1111111111001000
A/1	9	111111000
A/2	16	1111111111001001
A/3	16	1111111111001010
A/4	16	1111111111001011
A/5	16	1111111111001100
A/6	16	1111111111001101
A/7	16	1111111111001110
A/8	16	1111111111001111
A/9	16	1111111111010000
A/A	16	1111111111010001

B/1	9	111111001

Tabela C.4 – Tabela da crominância para os coeficientes AC (parte 2 de 3).

B/2	16	1111111111010010
B/3	16	1111111111010011
B/4	16	1111111111010100
B/5	16	1111111111010101
B/6	16	1111111111010110
B/7	16	1111111111010111
B/8	16	1111111111011000
B/9	16	1111111111011001
B/A	16	1111111111011010
C/1	9	111111010
C/2	16	1111111111011011
C/3	16	1111111111011100
C/4	16	1111111111011101
C/5	16	1111111111011110
C/6	16	1111111111011111
C/7	16	1111111111100000
C/8	16	1111111111100001
C/9	16	1111111111100010
C/A	16	1111111111100011
D/1	11	11111111001
D/2	16	1111111111100100
D/3	16	1111111111100101
D/4	16	1111111111100110
D/5	16	1111111111100111
D/6	16	1111111111101000
D/7	16	1111111111101001
D/8	16	1111111111101010
D/9	16	1111111111101011
D/A	16	1111111111101100
E/1	14	11111111100000
E/2	16	1111111111101101
E/3	16	1111111111101110
E/4	16	1111111111101111
E/5	16	1111111111110000
E/6	16	1111111111110001
E/7	16	1111111111110010
E/8	16	1111111111110011
E/9	16	1111111111110100
E/A	16	1111111111110101
F/0	10	1111111010
F/1	15	111111111000011
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

Tabela C.4 – Tabela da crominância para os coeficientes AC (parte 3 de 3).