

# Co-processador da Transformada para o Codificador de Vídeo AV1

Apresentação Final

---

Miguel Inocêncio

Mestrado Integrado em Engenharia Eletrónica e de Telecomunicações

18/12/2019

Universidade de Aveiro

Instituto de Telecomunicações



Introdução

Sistemas de Codificação de Vídeo

Transformadas em Codificação de Vídeo

Arquiteturas Desenvolvidas

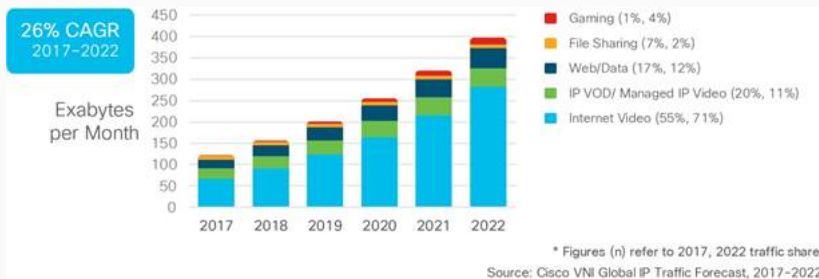
Software

Hardware

Conclusões e Trabalho Futuro

# Introdução

---



**Figura 1:** Previsões da *Cisco* para evolução de tráfego IP

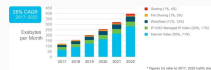


Figura 1: Previsões da Cisco para evolução de tráfego IP

Consumo de vídeo tem vindo a aumentar exponencialmente

A Cisco prevê que para 2022 82% do tráfego IP esteja dedicado à visualização de vídeo



**Figura 2:** Exemplo de dados em vídeo HD

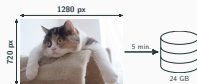


Figura 2: Exemplo de dados em vídeo HD

Enorme quantidade de dados gerados com a captura ou criação de vídeo

Vídeo HD a 30 frames por segundo num espaço RGB de 8 bits por cor ocuparia 24GB em 5 minutos

Para as resoluções que se desejam hoje em dia este problema seria ainda mais grave

Necessidade de reduzir quantidade de informação precisa para reproduzir um vídeo



**Remoção de informação de sequência de  
imagens, mantendo a capacidade de  
reprodução**



# Apresentação Final

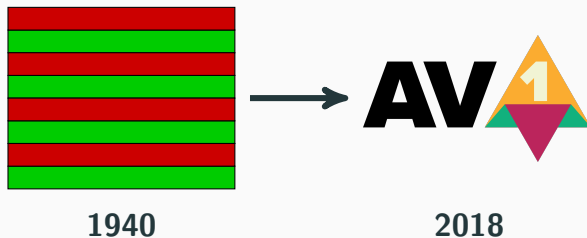
└─ Introdução

└─ Codificação de Vídeo



Remoção de informação de sequência de  
imagens, mantendo a capacidade de  
reprodução

Levou ao conceito de codificação de vídeo



**Figura 3:** Exemplo de *interlaced scanning* e logo do AV1

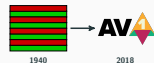


Figura 3: Exemplo de interlaced scanning e logo do AV1

Em prática desde os anos 40 com o interlaced scanning das televisões de raios catódicos

Evolução do vídeo levou à evolução dos métodos de compressão (aumento da complexidade)

Alliance for Open Media Video One ou AV1 apresenta grandes taxas de compressão, a custo de elevada complexidade

Necessidade de software otimizado e arquiteturas de hardware eficientes

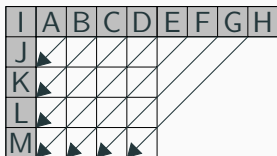
# Sistemas de Codificação de Vídeo

---

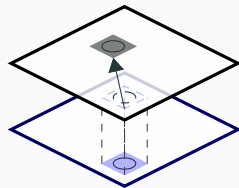
Operação feita por codec

Composto por codificador e decodificador

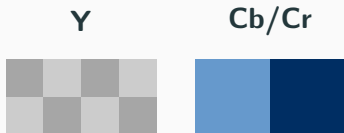
Tem como princípio base a remoção de dados previsíveis, ou redundantes



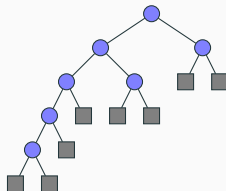
**Figura 4:** Espaciais



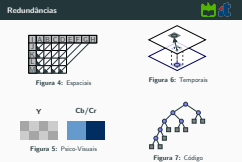
**Figura 6:** Temporais



**Figura 5:** Psico-Visuais



**Figura 7:** Código



Apesar da evolução, os princípios de base continuam os mesmos  
4 tipos de redundâncias, a maioria causadas pela interpretação do  
olho humano

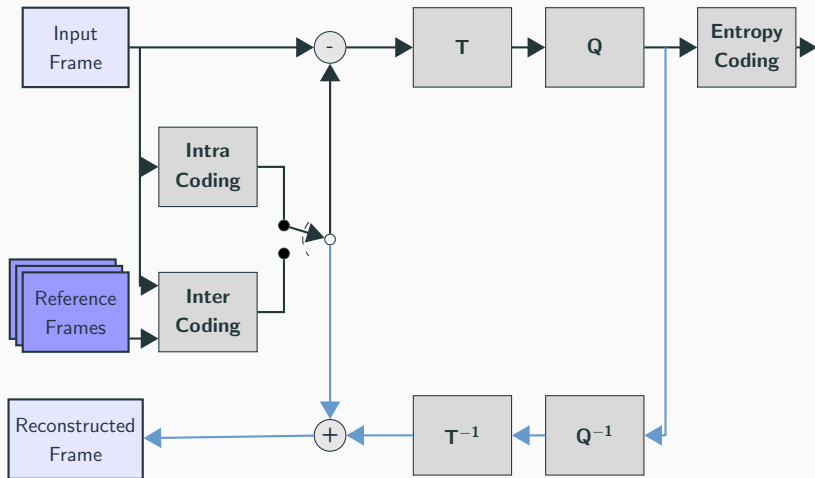
Espaciais referentes à proximidade de pixels próximos

Temporais referentes à semelhança de pixels em imagens consecutivas

Psicovisuais referentes à percepção mais baixa da cor ou de detalhes

Código, não sendo referente à imagem ou percepção, mas à representação dos símbolos em domínio digital

Estas redundâncias são exploradas em vários estágios de um codificador de vídeo



**Figura 8:** Modelo Básico de codificador



## Apresentação Final

## └ Sistemas de Codificação de Vídeo

## └ Modelo Básico do Codificador

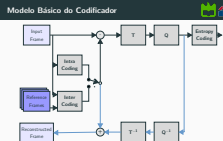


Figura 8: Modelo Básico de codificador

Processo começa com frame de entrada que é dividido em blocos

Estágio de predição Intra ou Inter

Bloco previsto subtraído por original

Transformada é o foco do trabalho

Avalia componentes de frequência

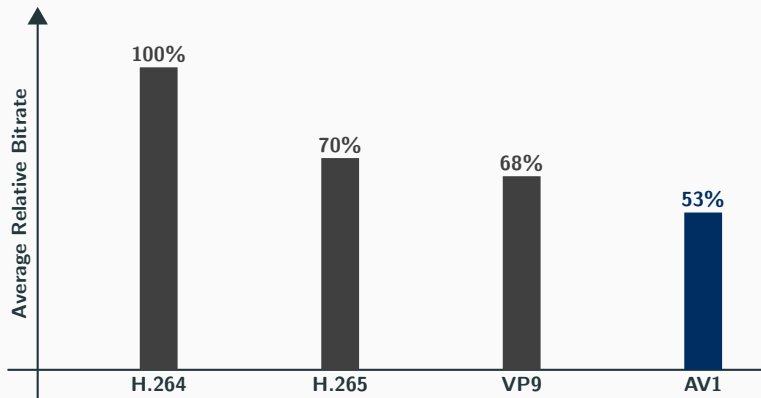
Quantização avalia coeficientes de maior relevância para reconstrução de imagem

codificador de entropia organiza símbolos segundo códigos de comprimento variável

Loop de feedback para restaurar imagem do decodificador para uso nos estágios de predição

Unidade de controlo escolhe quais as ferramentas de codificação a usar

Decodificador faz operação inversa



**Figura 9:** Poupanças de Bitrate relativas ao H.264

**Tabela 1:** Tempos de Codificação

Codec	Tempo de Codificação (s)	
	2018	2019
AV1	226 080	736
H.265		289
VP9		226
H.264		18



## └ Sistemas de Codificação de Vídeo

### └ Performance do AV1

Tabela 1: Tempos de Codificação

Codec	Tempo de Codificação (s)	
	2018	2019
AV1	226 080	736
H.265		289
VP9		226
H.264		18

Processo complexo

Agravado pela complexidade do codificador

Quanto mais opções de codificação, melhor a performance de compressão, mas maior o tempo de operação

AV1 apresenta grandes poupanças em bit savings a custo de elevados tempos de operação

Melhorias até 30% em relação ao HEVC ou VP9 (formatos de codificação recentes)

Demora até 3 vezes mais para atingir a mesma distorção

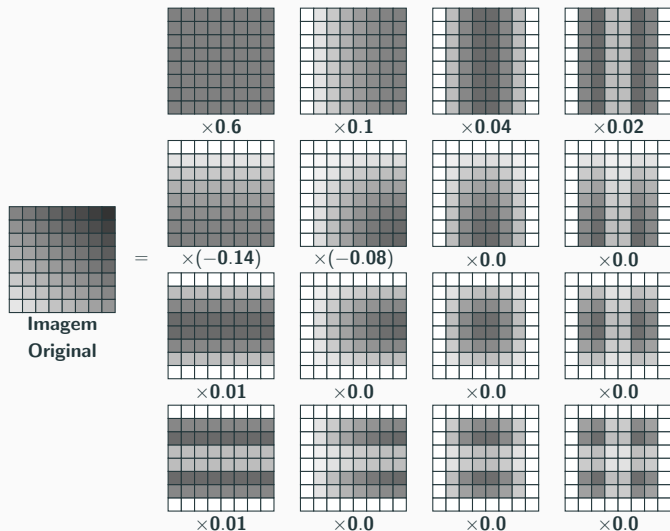
Melhoria ao longo dos anos com otimização do software

# Transformadas em Codificação de Vídeo

---

Avanço para o estudo do software de referência  
objetivo de perceber funcionamento interno do estágio da Transformada

Objetivo do estágio é decomposição em componentes de frequência

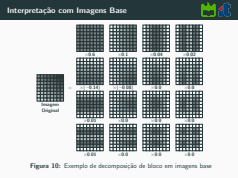


**Figura 10:** Exemplo de decomposição de bloco em imagens base

# Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Interpretação com Imagens Base



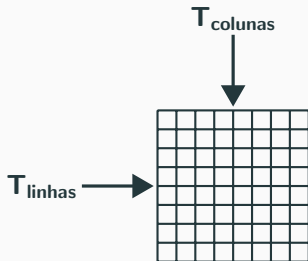
Interpreção em imagens de base: um bloco original pode ser visto como a soma de diversos blocos com diferentes componentes de frequência horizontal e/ou vertical

Transformada vista como calculo da correlação entre imagem original e imagens base

Conjunto de imagens base depende da transformada utilizada e do tamanho do bloco a transformar

AV1 suporta blocos entre 4 e 64, incluindo tamanhos rectangulares





**Figura 11:** Separabilidade de transformadas 2D

Discrete Cosine Transform (DCT)

Identity (IDTX)

Asymmetric Discrete Sine Transform (ADST)

*Flip* - Asymmetric Discrete Sine Transform (Flip-ADST)

└ Transformadas em Codificação de Vídeo

└ Transformadas em Codificação de Vídeo



Figura 11: Separabilidade de transformadas 2D

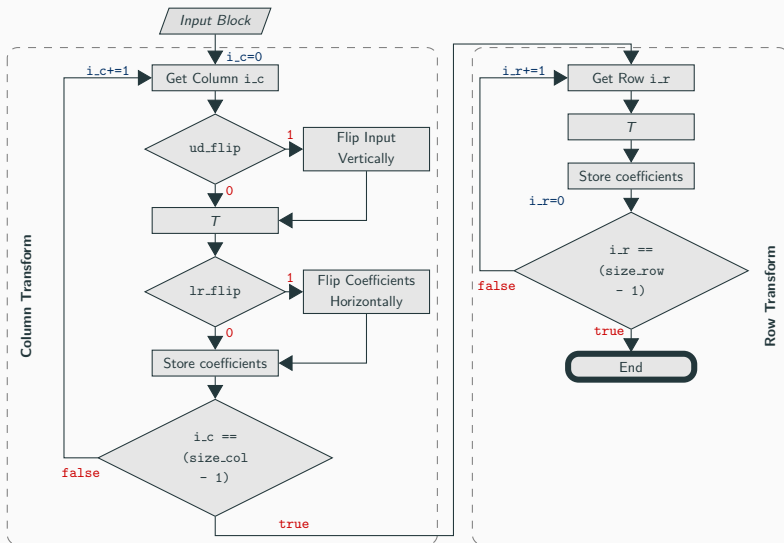
Discrete Cosine Transform (DCT)  
Identity (IDTX)  
Asymmetric Discrete Sine Transform (ADST)  
Flip - Asymmetric Discrete Sine Transform (Flip-ADST)

Bloco de duas dimensões implica transformação a duas dimensões  
Transformada pode ser feita em duas operações separáveis para linhas e colunas ou vice-versa

Operações denominadas por kernels da Transformada

AV1 suporta 3 tipos: Identidade, DCT e ADST que pode ser calculada direta ou inversamente

Kernels podem ser utilizados independentemente na vertical ou horizontal



**Figura 12:** Sequência de operações da Transformada no *libaom*

## Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Transformada no AV1

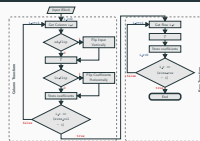


Figura 12: Sequência de operações da Transformada no AV1

Diagrama de operação das transformadas no AV1

Começa com a transformada vertical (colunas)

Blocos de flip usados quando se pretende fazer a transformação com Flip-ADST

Quando todas as colunas foram transformadas, segue para a transformação linha a linha

Kernels representados pelos blocos T

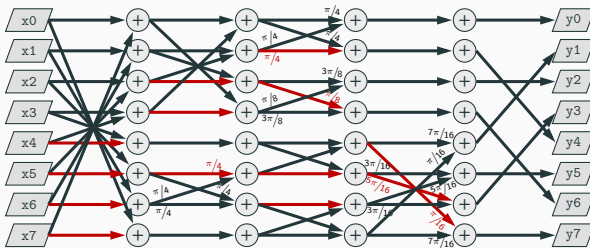


Figura 13: DCT no libaom

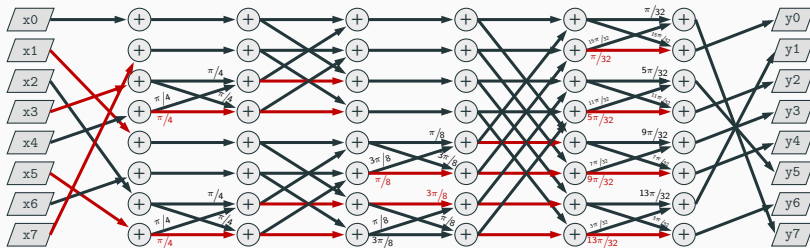
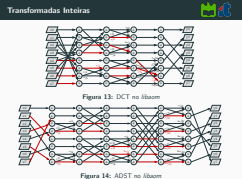


Figura 14: ADST no libaom

## └ Transformadas em Codificação de Vídeo

## └ Transformadas Inteiras



Coeficientes calculados com operações Inteiras de modo a simplificar calculo e evitar discrepâncias entre codificador e decodificador

Métodos eficazes de calcular a DCT tem sido desenvolvidos

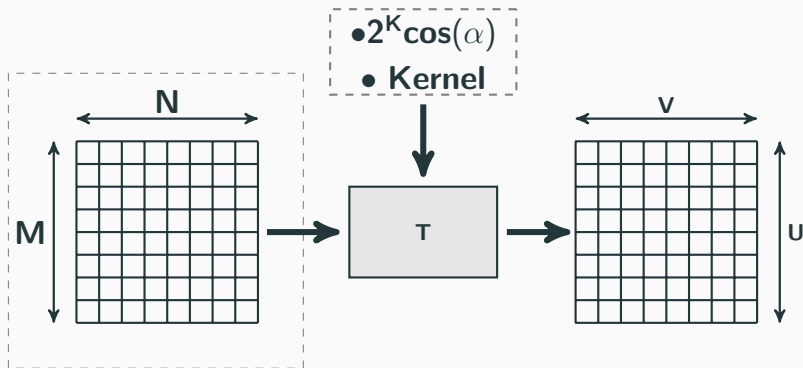
AV1 aplica transformadas sequenciais com estágios de rotação

Princípio aplicado também na ADST

Simplifica implementações em hardware

Cada coeficiente intermédio é calculado como função de dois dos calculados anteriormente e aproximações inteiras do cosseno

Software de referencia permite aproximações com 10 a 16 bits

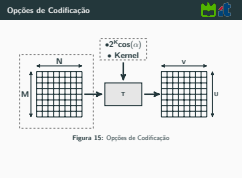


**Figura 15:** Opções de Codificação

## Apresentação Final

## └ Transformadas em Codificação de Vídeo

## └ Opções de Codificação



Estágio da transformada controlado por:

Tamanho do bloco de transformação

kernel utilizado horizontal e verticalmente

$n^{\circ}$  de bits utilizado nas aproximações de cossenos

Testes para saber quais as opções mais utilizadas:

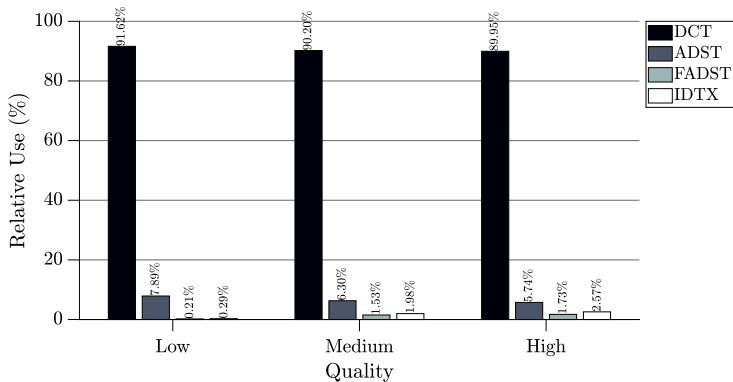
12 vídeos de resoluções entre CIF e UHD

Testar variação das opções com a qualidade do vídeo

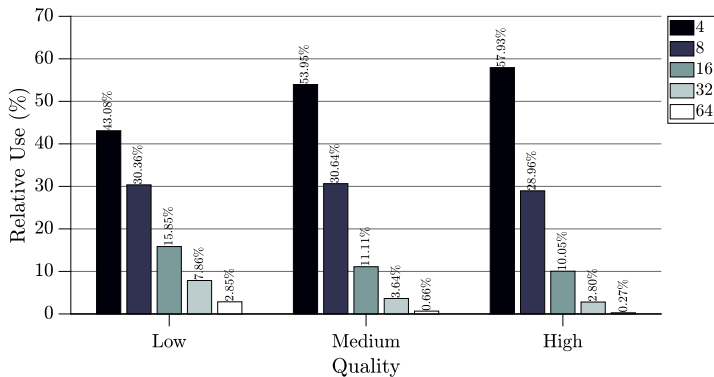
encode de qualidade constante com 3 objetivos distintos de qualidade

Tempo passado no estágio da transformada

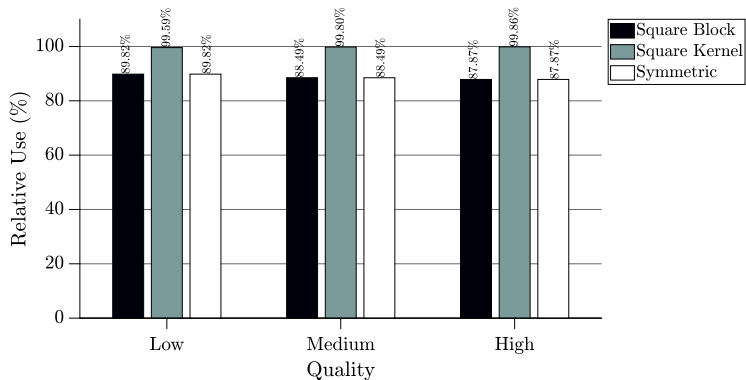




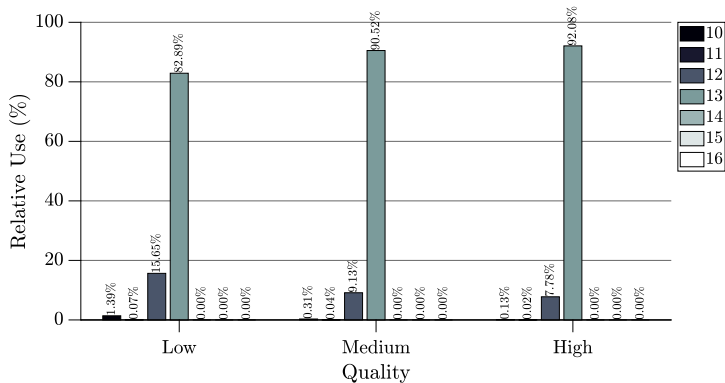
**Figura 16:** Kernel Utilizado



**Figura 17: Tamanho de Vetor**



**Figura 18:** Kernel Simétrico



**Figura 19:** Número de Bits Utilizados nas Aproximações do Cosseno

# Apresentação Final

## └ Transformadas em Codificação de Vídeo

### └ Opções de Codificação - Resultados

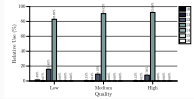


Figura 19: Número de Bits Utilizados nas Aproximações do Cosseno

DCT é a mais utilizada

Outros kernels só são utilizados em objetivos de qualidade mais elevados

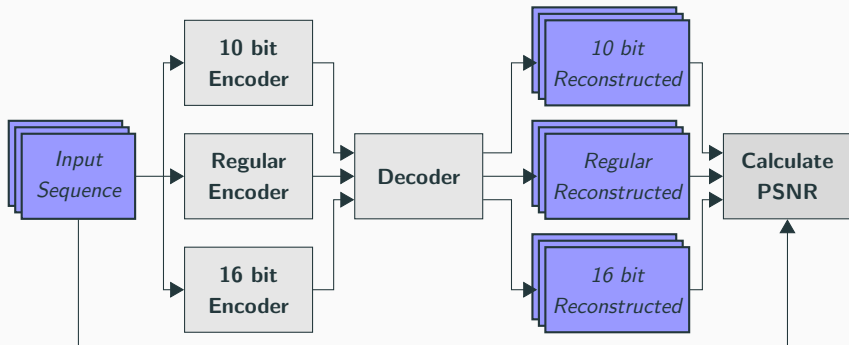
A dimensão mais comum é 4, e o seu uso aumenta com a qualidade pretendida

Esperado, pois para uma dada área de imagem ser transformada com vetores de 1 por 4, tem que ser feitas mais transformações do que para vetores de dimensões superiores

É possível verificar a percentagem de ocorrências para kernels simétricos (tamanho e kernel de colunas igual a tamanho e kernel de linhas)

A maior parte das transformações usa 13 bits para as representações dos cossenos

À medida que a qualidade aumenta, também aumenta o número de

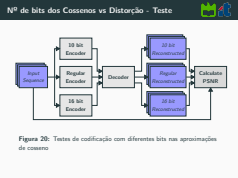


**Figura 20:** Testes de codificação com diferentes bits nas aproximações de cosseno

## Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Nº de bits dos Cossenos vs Distorção - Teste



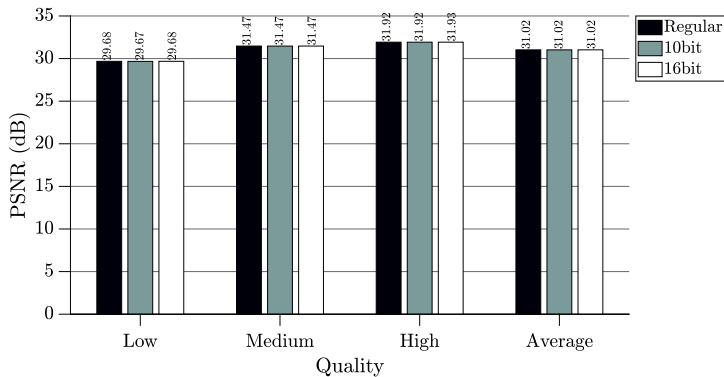
Poderá levar a pensar que o número de bits do cosseno influencia a qualidade da imagem

Testes feitos para avaliar esta hipótese

Testes de codificação forçando o número de bits utilizados nas aproximações dos cossenos, no codificador

Descodificação feita com o descodificador de origem, que usa sempre 12 bits

Cálculo da Relação Sinal Ruído de Pico e comparar com os tres codificadores, para os tres objetivos de qualidade distintos



**Figura 21:** Comparação Distorção com Número de Bits usados no Cosseno



## Apresentação Final

└ Transformadas em Codificação de Vídeo

└ N<sup>o</sup> de bits dos Cossenos vs Distorção - Resultados

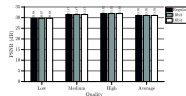


Figura 21: Comparação Distorção com Número de Bits usados no Cosseno

PSNR independente do número de bits utilizado no cosseno, para qualquer objetivo de qualidade

Princípio de base para otimização do software de referencia

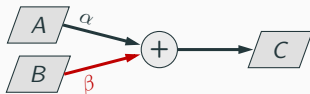
Outros aspetos podem ser explorados para a otimização do software de referencia

# Arquiteturas Desenvolvidas

---

Software

Foco na DCT pois seria a que causaria mais impacto na performance do encoder



$$C = (\alpha \cdot A + \beta \cdot B) \gg 8$$

**Figura 22:** Operação implementada nas transformadas inteiras

$$M_{original} = 728 B$$

$$M_{8bits} = 64 B \approx 0.2 \cdot M_{original}$$

$$\Delta_{10} = \frac{1 - 0}{2^{10}} \approx 0.98 \cdot 10^{-3}$$

$$\Delta_8 = \frac{1 - 0}{2^8} \approx 3.9 \cdot 10^{-3}$$

$\Downarrow$

$$MSE_8 = 16 \cdot MSE_{10}$$

## Apresentação Final

└ Arquiteturas Desenvolvidas

└ Software

└ Redução do Número de Bits



$$C = (\alpha A + \beta B) \gg 8$$

Figura 22: Operação implementada nas transformadas inteiras

$$\begin{aligned}
 M_{\text{original}} &= 728 B \\
 M_{\text{bits}} &= 64 B \approx 0.2 \cdot M_{\text{original}} \\
 \Delta_{10} &= \frac{1-0}{2^{10}} \approx 0.98 \cdot 10^{-3} \\
 \Delta_8 &= \frac{1-0}{2^8} \approx 3.9 \cdot 10^{-3} \\
 &\quad \Downarrow \\
 \text{MSE}_8 &= 16 \cdot \text{MSE}_{10}
 \end{aligned}$$

Testar a hipótese da redução de complexidade pela redução do número de bits dos cossenos

Multiplicações mais simples

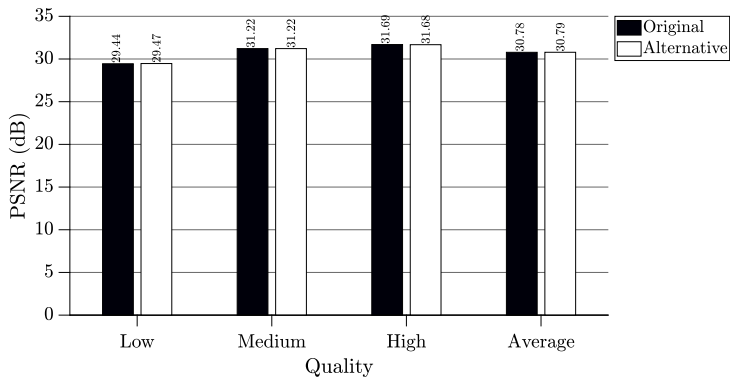
Número de Shifts reduzido

Teste com 8 bits

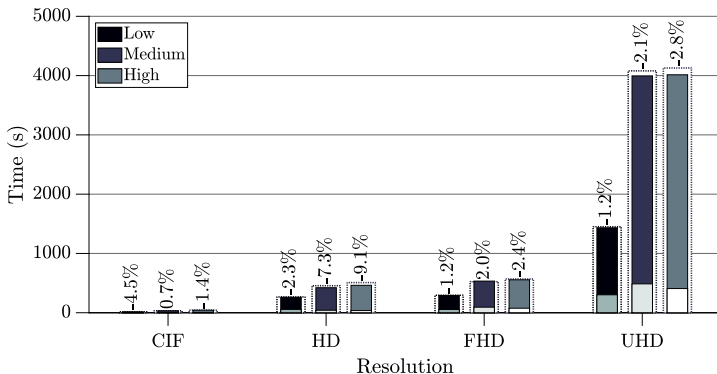
levaria à redução em 81% da memória utilizada para armazenamento destas aproximações

Possibilidade de degradação da qualidade de imagem pois o Erro de Quantização Quadrático médio aumentaria em 16 vezes

Repetição dos testes de distorção usando encoder com 8 bits e encoder original



**Figura 23:** Comparação da distorção



**Figura 24:** Tempo de Codificação

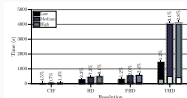


Figura 24: Tempo de Codificação

Qualidade mantém-se em ambas codificações

Redução de 3% no tempo de codificação:

Tracejado: Tempo de codificação com codificador original

Barras escuras: Tempo de codificação do codificador modificado

Barras claras: tempo da Transformada

Percentagem: Diferença percentual do tempo entre codificador original e codificador modificado

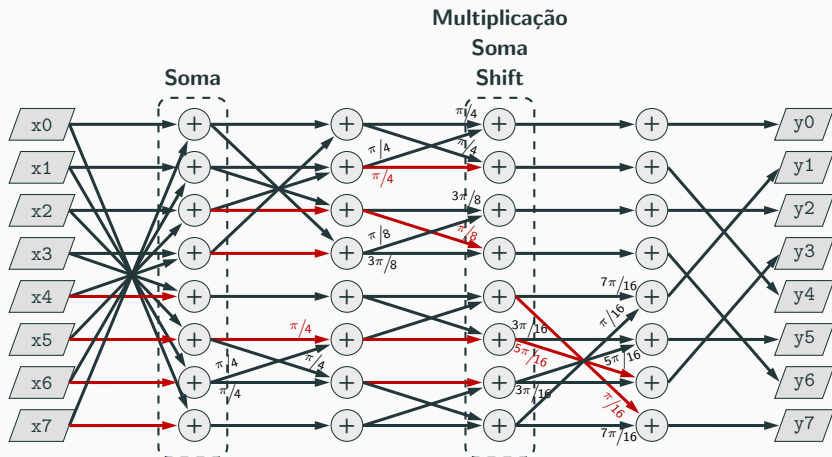
Avanço para arquiteturas em hardware



# Arquiteturas Desenvolvidas

---

Hardware



**Figura 25:** Estágios da DCT inteira

## Apresentação Final

- └ Arquitecturas Desenvolvidas
  - └ Hardware
    - └ Princípios de desenvolvimento

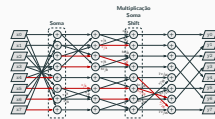
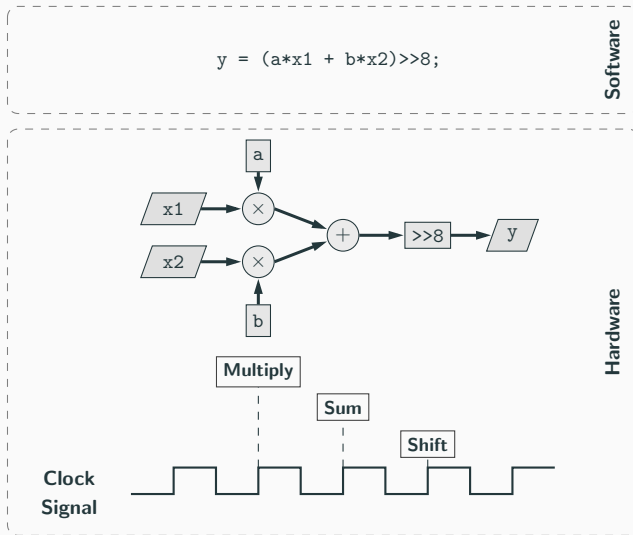


Figura 25: Estágios da DCT inteira

Desenvolvimento foi feito em VHDL no Vivado da Xilinx  
Designs e sínteses feitas com objetivo de implementação em Artix 7  
Cada estágio é feito sequencialmente  
Estágios de somas simples e de Multiplicação, Soma e Shift



**Figura 26:** Deconstrução de operação em software

- └ Arquiteturas Desenvolvidas
  - └ Hardware
    - └ Princípios de desenvolvimento

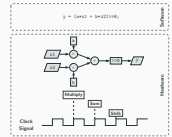
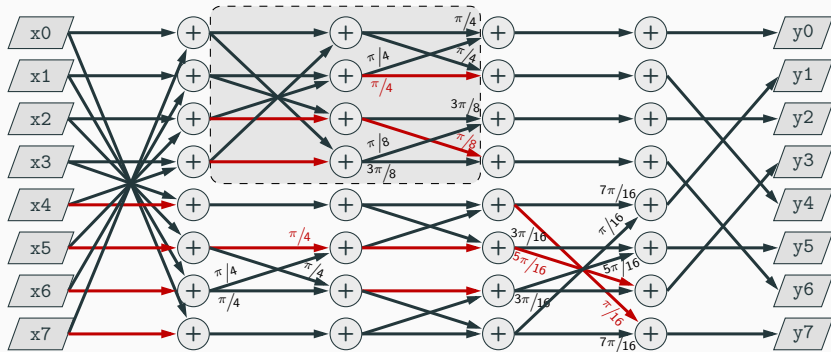


Figura 26: Decomposição de operação em software

em software é uma operação fácil, descrita numa linha de códigos  
em hardware tem que ser decomposta em três estágios diferentes  
controlado pelo flanco ascendente de um sinal de clock

## DCT4



**Figura 27:** Inclusão de DCT4 na DCT8

- └ Arquitecturas Desenvolvidas
  - └ Hardware
    - └ Princípios de desenvolvimento

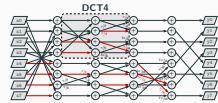
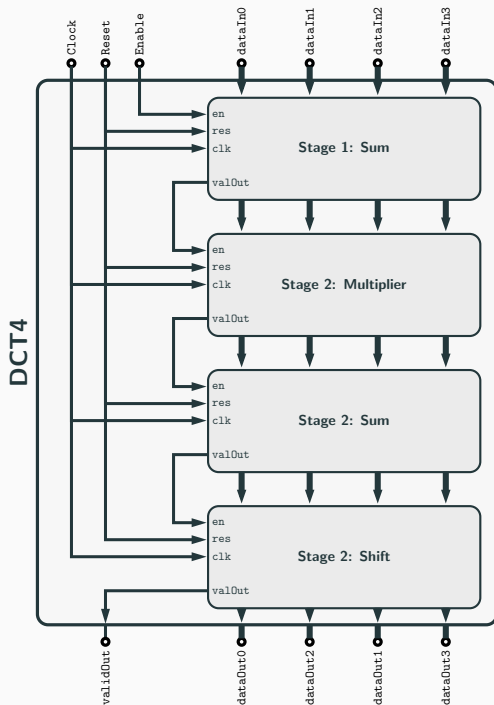


Figura 27: Inclusão de DCT4 na DCT8

Operações das DCTs mais pequenas são contidas nas DCTs de maiores dimensões

Comportamento replica-se para os restantes tamanhos

Permite repetição de hardware, simplificando o desenvolvimento



**Figura 28:**  
Implementação  
em hardware da  
DCT4



## Apresentação Final

└ Arquitecturas Desenvolvidas

└ Hardware

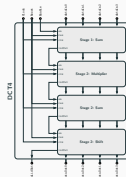


Figura 28:  
Implementação  
em hardware da  
DCT4

Arquitetura mais simples

Ativada por sinal de enable

Estágios internos controlados por sinal de enable

Quando terminam estágio intermédio geram validação da saída

Pipeline de estágios feito com interligação dos sinais de validação de saída de um estágio com o enable do próximo

**Figura 29:**  
Implementação  
em hardware da  
DCT8

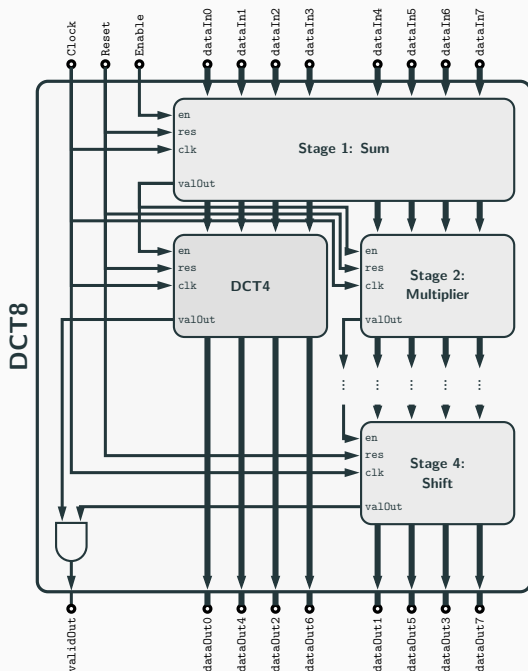
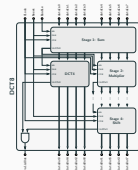
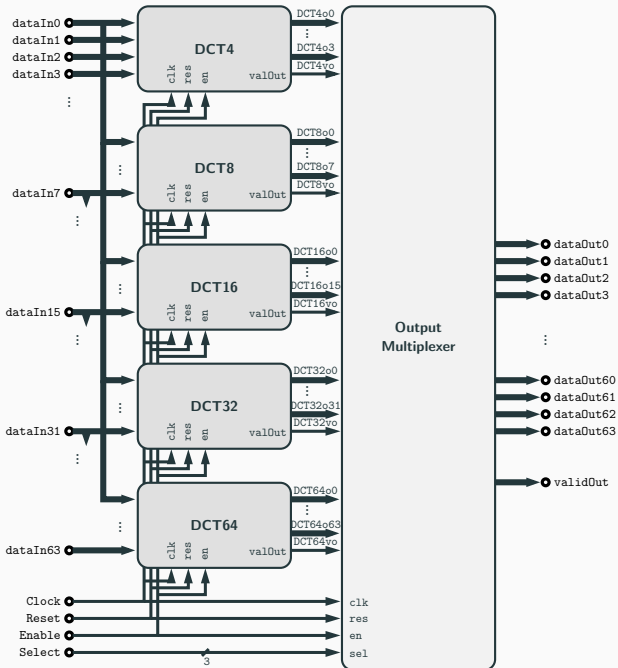


Figura 29:  
Implementação  
em hardware da  
DCT8



Segue o mesmo molde da arquitetura anterior, mas DCT4 é incluída internamente

validOut dependente do ultimo estágio interno, bem como da DCT4

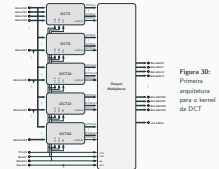


**Figura 30:**  
Primeira  
arquitetura  
para o kernel  
da DCT

## Apresentação Final

## └ Arquitecturas Desenvolvidas

## └ Hardware



As restantes transformadas seguem o mesmo design

Interligando todos com um multiplexer de seleção do output obtém-se a seguinte arquitetura

Permite cálculo dos 5 tamanhos de transformada em paralelo

Repetição de hardware causa que seja uma arquitetura de dimensões elevadas com muita utilização lógica

**Tabela 2:** Resultados de utilização lógica da primeira arquitetura em família Artix 7

<b>DCT Size</b>	<b>Utilization</b>	
	<b>Slice LUTs</b>	<b>Slice Registers</b>
<b>4</b>	1125	636
<b>8</b>	2428	2087
<b>16</b>	7103	5702
<b>32</b>	19148	14257
<b>64</b>	45996	34146
<b>Wrapper</b>	75805	58370

**Tabela 3:** Frequência de operação necessária para codificação em tempo real a 30 imagens por segundo

Resolution	Frequency (MHz)
<b>1280 × 720</b>	83
<b>1920 × 1080</b>	187
<b>3840 × 2160</b>	746
<b>7680 × 4320</b>	2986

# Apresentação Final

## └ Arquiteturas Desenvolvidas

### └ Hardware

#### └ Primeira arquitetura - Resultados



Tabela 3: Frequência de operação necessária para codificação em tempo real a 30 imagens por segundo

Resolution	Frequency (MHz)
1280 × 720	83
1920 × 1080	187
3840 × 2160	746
7680 × 4320	2986

Velocidade do Sistema dependente da transformada mais lenta (DCT64), que demora 22 ciclos de relógio

Considerando que um frame seria codificado com blocos de transformação quadrados, é possível calcular a frequência de operação mínima necessária para codificar vídeo de uma dada resolução com uma frame rate específica

Para resoluções HD e FHD o sistema poderia facilmente atingir a frequência necessária (dependendo da implementação)

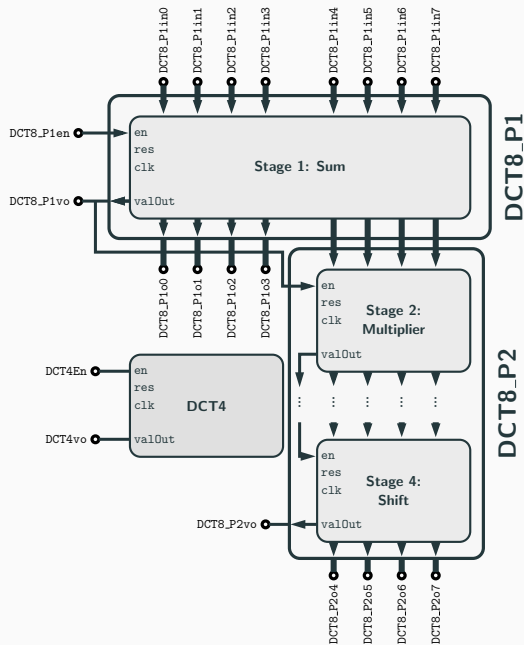
Para resoluções mais altas não se pode dizer o mesmo

Necessidade de arquiteturas com elevado grau de paralelismo

Tamanhos superiores ocupam muita área



**Figura 31:**  
Deconstrução de  
blocos da DCT

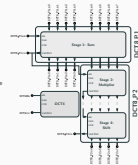


## Apresentação Final

## └ Arquiteturas Desenvolvidas

## └ Hardware

Figura 31:  
Deconstrução de  
bloco de DCT

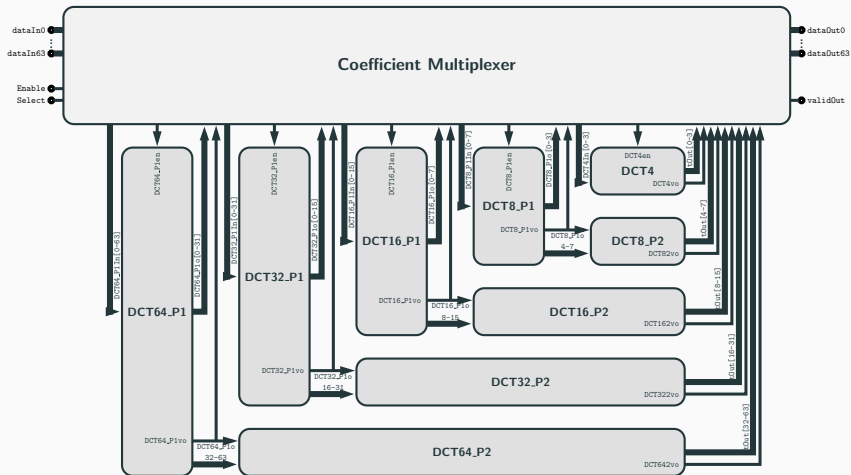


De forma a ocupar menos área, é possível construir uma arquitetura sem repetição de hardware

Divisão das DCTs em duas partes

P1 com primeiro estágio de soma e rotação

P2 com restantes estágios, e metade dos coeficientes



**Figura 32:** Segunda arquitetura para o kernel da DCT

## Apresentação Final

└ Arquitecturas Desenvolvidas

└ Hardware

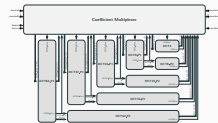


Figura 32: Segunda arquitetura para o kernel da DCT

Aplicando às restantes DCTs é possível obter a seguinte arquitetura

Entradas seguem para uma das fases P1, consoante o sinal de seleção

**Tabela 4:** Resultados de utilização lógica da segunda arquitetura em família Artix 7

Block	Utilization	
	Slice LUTs	Slice Registers
DCT4	1077	507
DCT8_P1	709	257
DCT8_P2	1064	717
DCT16_P1	1285	513
DCT16_P2	3860	2150
DCT32_P1	3064	1025
DCT32_P2	9090	5624
DCT64_P1	6123	2049
DCT64_P2	22344	14000
Wrapper	50039	32352

## Apresentação Final

## └ Arquiteturas Desenvolvidas

## └ Hardware

## └ Segunda arquitetura - Resultados



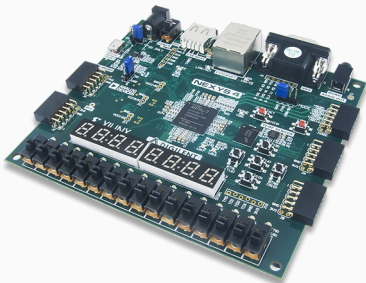
Tabela 4: Resultados de utilização lógica da segunda arquitetura em família Artix 7

Block	Utilization	
	Slice LUTs	Slice Registers
DCT4	2077	507
DCT8,P1	709	257
DCT8,P2	1064	717
DCT16,P1	1285	513
DCT16,P2	3860	2150
DCT32,P1	3064	3025
DCT32,P2	9090	5624
DCT64,P1	6123	2049
DCT64,P2	22344	14000
Wrapper	50039	32352

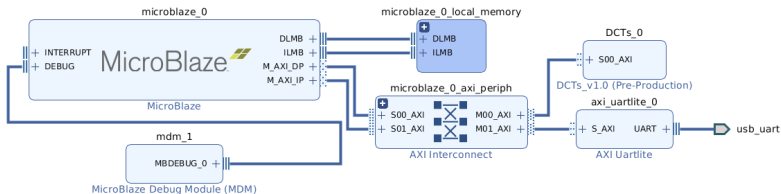
Ocupa dois terços da implementação anterior

Apenas permite calculo de um dos tamanhos pois parte do software está sempre utilizado

Menos adaptada ao uso em encoder complexo, pois não permite paralelização de opções de codificação



**Figura 33:**  
Kit Nexys 4  
da Digilent



**Figura 34:** Diagrama de blocos implementado

## Apresentação Final

- └ Arquiteturas Desenvolvidas
  - └ Hardware
    - └ Implementação Nexys 4



Figura 33:  
Kit Nexys 4  
da Digilent



Figura 34: Diagrama de bloco implementado

Ultimo passo desta dissertação foi a integração da segunda arquitetura num design com Microblaze, num kit de hardware Nexys 4 da Digilent

Micro-Processador ARM usado em ferramentas da Xilinx

Prova de conceito para codificador completo



$$f_{Max} = 101.9 \text{ MHz}$$

$$P = 50 \text{ mW}$$

**Tabela 5:** Frame rate máximo obtido na implementação com Nexys 4

Block Size	Resolution			
	1280 × 720	1920 × 1080	3840 × 2160	7680 × 4320
4 × 4	37	16	4	1
8 × 8	44	20	5	1
16 × 16	63	28	7	2
32 × 32	98	44	11	3
64 × 64	161	71	18	4

## Apresentação Final

└ Arquiteturas Desenvolvidas

└ Hardware

└ Implementação Nexys 4 - Resultados



$$f_{\text{clk}} = 101.0 \text{ MHz}$$

$$P = 50 \text{ mW}$$

Tabela 5: Frame rate máximo obtido na implementação com Nexys 4

Block Size	Resolution			
	1280 × 720	1920 × 1080	3840 × 2160	7680 × 4320
4 × 4	37	16	4	1
8 × 8	44	20	5	1
16 × 16	63	28	7	2
32 × 32	98	44	11	3
64 × 64	101	71	18	4

Vivado estima uma frequência máxima de operação de aproximadamente 102 MHz

50 mW de potência consumida

Frame rates baixos neste hardware, apesar do funcionamento ser o Esperado

Justifica-se a necessidade de implementações em ASIC, capazes de frequências de clock mais elevadas

## **Conclusões e Trabalho Futuro**

---



- ✓ Otimização do Software de referência
  - ✓ Construção de arquiteturas em hardware para o kernel da DCT
- 
- Integração dos restantes kernels
  - Teste com *libaom* em FPGA
  - Síntese para ASIC

- ✓ Otimização do Software de referência
- ✓ Construção de arquiteturas em hardware para o kernel da DCT

- Integração dos restantes kernels
- Teste com libaom em FPGA
- Síntese para ASIC

Objetivos parcialmente atingidos

Otimização do software de referência com estudo das características de codificação do AV1

Construção de arquiteturas de hardware funcionais para a DCT

Pontos em falta para um verdadeiro estágio da transformada em hardware

Integração dos restantes kernels

Teste da arquitetura de hardware com libaom em FPGA com interface com muita largura de banda (PCIe por exemplo)

Síntese para ASIC para obter frequência máxima de operação

**Obrigado!**

# Discussão

**Co-processador da Transformada para o  
Codificador de Vídeo AV1**

**Miguel Oliveira Inocêncio**

**Armando Pinho**  
Presidente de Mesa

**Pedro Assunção**  
Arguente Principal

**António Navarro**  
Orientador