

# 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

2019-12-16

Apresentação Final

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



- Aradeço a presença de todos
- Começo por uma breve introdução do contexto, nomeadamente o vídeo



Introdução

Sistemas de Codificação de Vídeo

Transformadas em Codificação de Vídeo

Arquiteturas Desenvolvidas

Software

Hardware

Conclusões e Trabalho Futuro

2019-12-16

Apresentação Final

└ Conteúdos



Introdução

Sistemas de Codificação de Vídeo

Transformadas em Codificação de Vídeo

Arquiteturas Desenvolvidas

Software

Hardware

Conclusões e Trabalho Futuro

2019-12-16

Apresentação Final  
└─ Introdução

Introdução

# Introdução



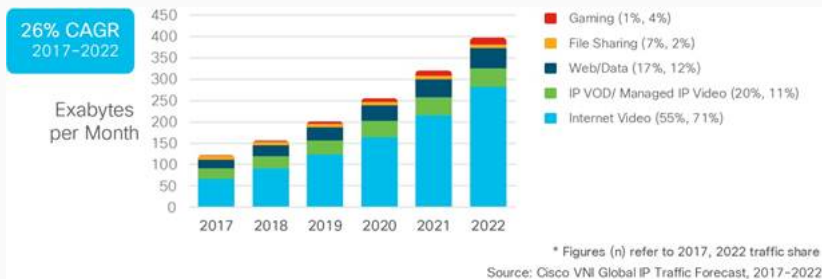


2019-12-16

Apresentação Final

└ Introdução

└ Consumo de Vídeo



**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



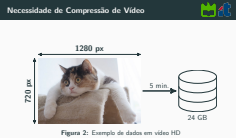
**Figura 2:** Exemplo de dados em vídeo HD

2019-12-16

Apresentação Final

└ Introdução

└ Necessidade de Compressão de Vídeo



- 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**

- Levou ao conceito de codificação de vídeo

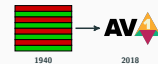
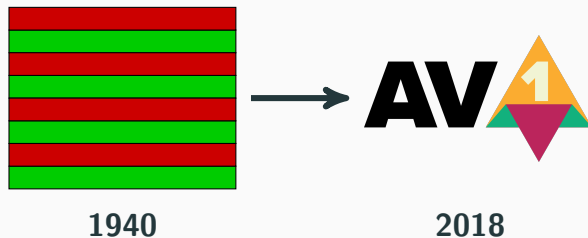


Figura 3: Exemplo de interlaced scanning e logótipo do AV1



**Figura 3:** Exemplo de *interlaced scanning* e logótipo 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

---

2019-12-16

Apresentação Final

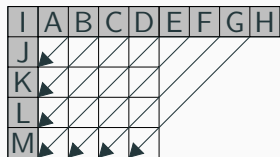
└ Sistemas de Codificação de Vídeo

Sistemas de Codificação de Vídeo

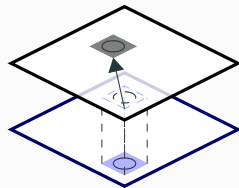
---

- Operação feita por codec
- Composto por codificador e decodificador
- Tem como princípio base a remoção de dados previsíveis, ou redundantes

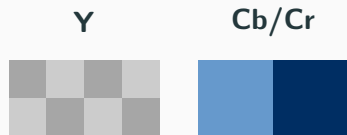




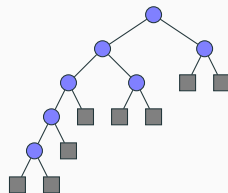
**Figura 4:** Espaciais



**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, exploradas em vários blocos do codificador
- Espaciais referentes à proximidade de pixels próximos, explorada na Predição Intra
- Temporais referentes à semelhança de pixels em imagens consecutiva, explorada na Predição Inter
- Psicovisuais referentes à percepção mais baixa da cor ou de detalhes, explorada com a sub-amostragem de cor e nos estágios de Transformada e quantização
- Código, não sendo referente à imagem ou percepção, mas à representação dos símbolos em domínio digital, explorada na codificação de entropia



Figura 4: Espaciais



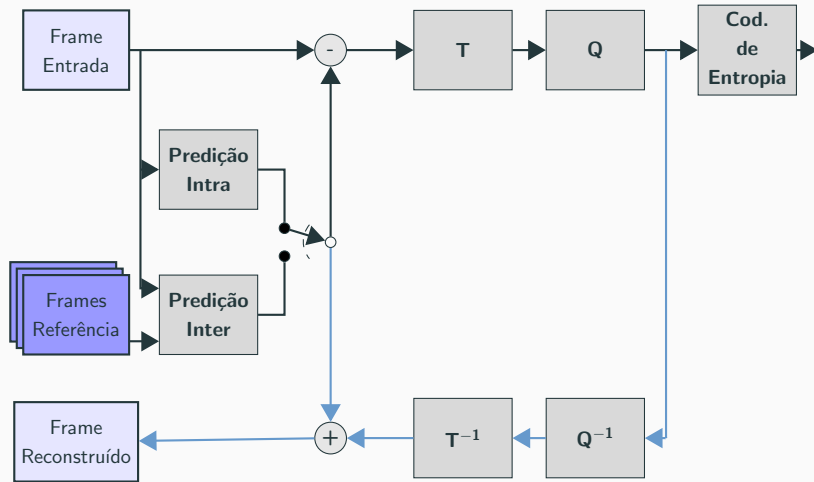
Figura 6: Temporais



Figura 5: Psico-Visuais



Figura 7: Código



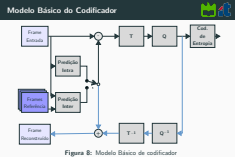
**Figura 8:** Modelo Básico de codificador

2019-12-16

## Apresentação Final

### Sistemas de Codificação de Vídeo

#### Modelo Básico do 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

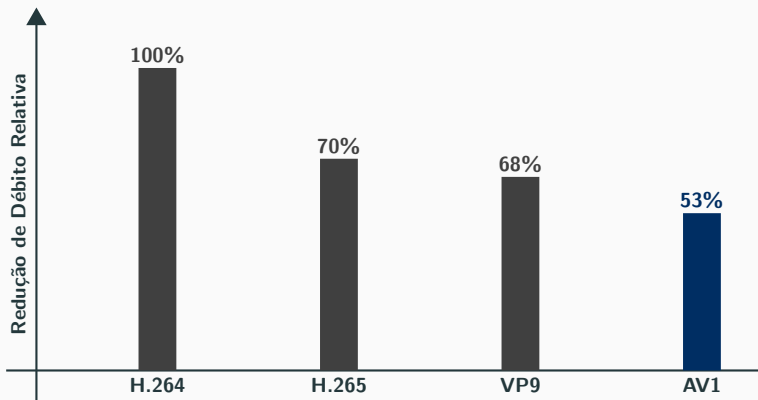


2019-12-16

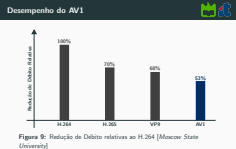
## Apresentação Final

### Sistemas de Codificação de Vídeo

#### Desempenho do AV1



**Figura 9:** Redução de Débito relativas ao H.264 [*Moscow State University*]



- Processo complexo, agravado em codificadores mais recentes
- 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)



Tabela 1: Tempos de Codificação

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

**Tabela 1:** Tempos de Codificação

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

- Demora até 3 vezes mais em encodes da mesma qualidade
- Necessidade de implementações em hardware
- Motivação para o desenvolvimento desta dissertação

# Transformadas em Codificação de Vídeo

---

2019-12-16

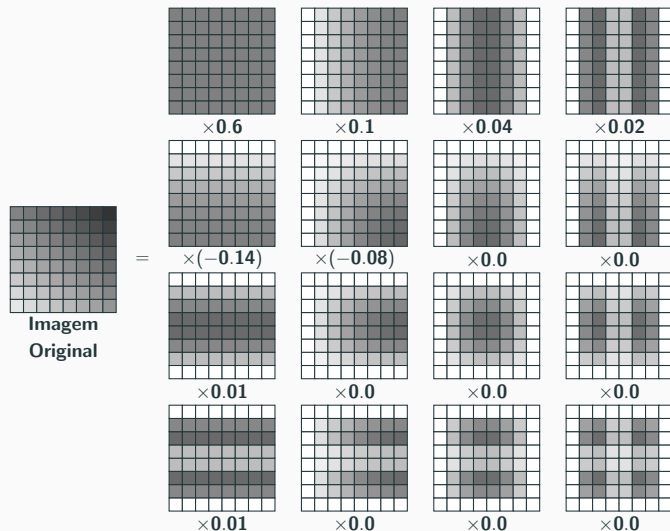
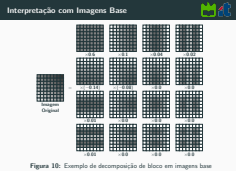
Apresentação Final

└ Transformadas em Codificação de Vídeo

Transformadas em Codificação de  
Vídeo

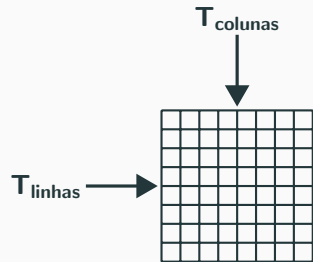
---

- 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

- 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

- Transformada Discreta do Cosseno (DCT)
- Identidade (IDTX)
- Transformada Discreta Assimétrica do Seno (ADST)
- *Flip* - ADST

2019-12-16

## Apresentação Final

### Transformadas em Codificação de Vídeo

### Transformadas em Codificação de Vídeo



Figura 11: Separabilidade de transformadas 2D

- Transformada Discreta do Cosseno (DCT)
- Identidade (IDTX)
- Transformada Discreta Assimétrica do Seno (ADST)
- *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



2019-12-16

## Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Transformada no AV1

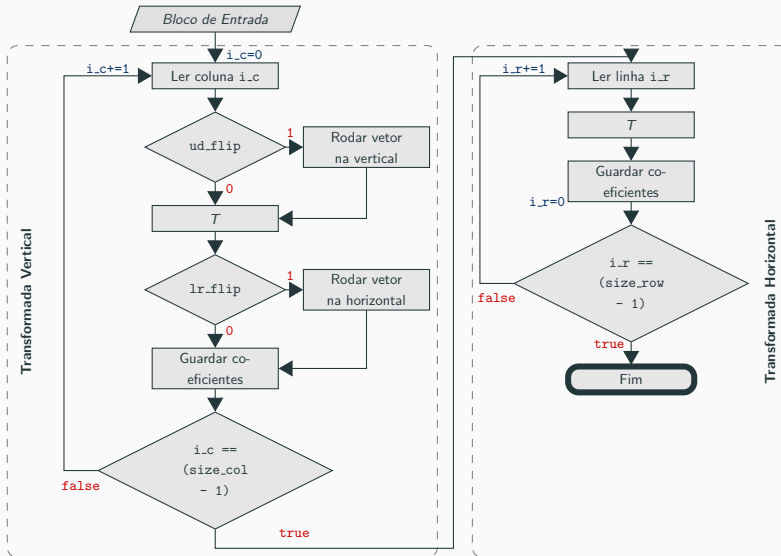
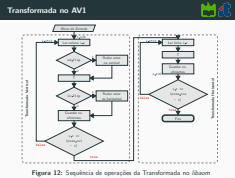


Figura 12: Sequência de operações da Transformada no *libaom*

- 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



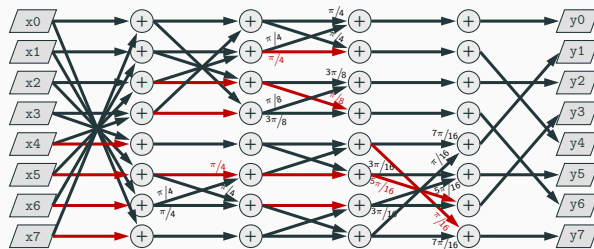


Figura 13: DCT no libaom

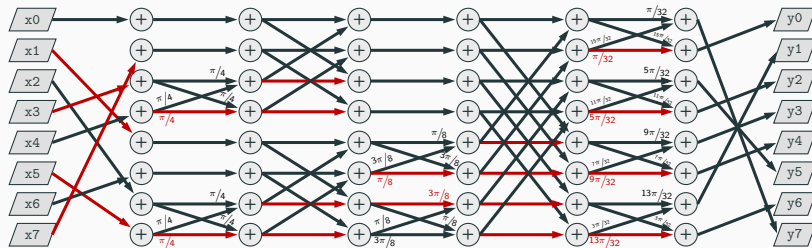


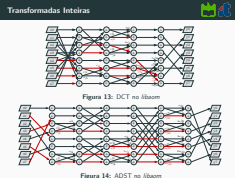
Figura 14: ADST no libaom

2019-12-16

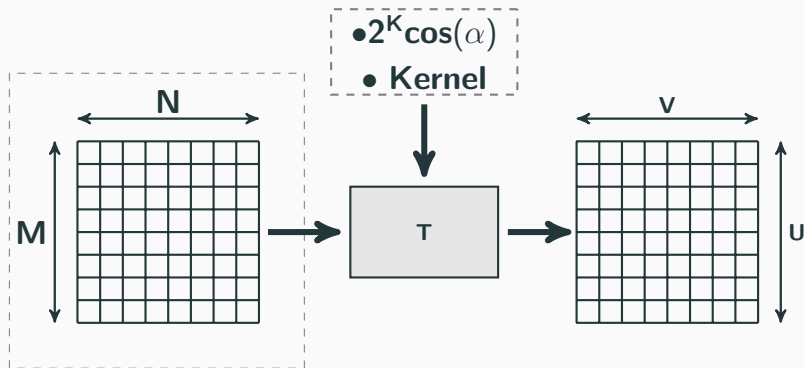
Apresentação Final

└ 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



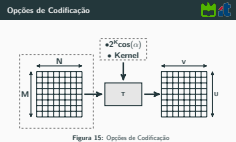
**Figura 15:** Opções de Codificação

2019-12-16

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
  - nº 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

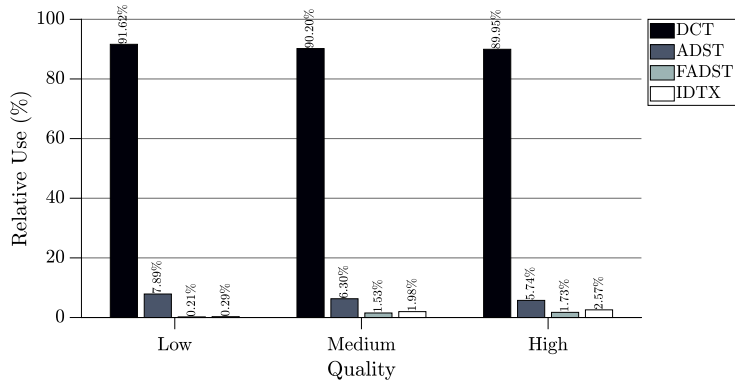


Figura 16: Kernel Utilizado

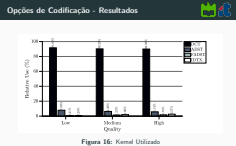
2019-12-16

Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Opções de Codificação - Resultados

- DCT é a mais utilizada
- Outros kernels só são utilizados em objetivos de qualidade mais elevados



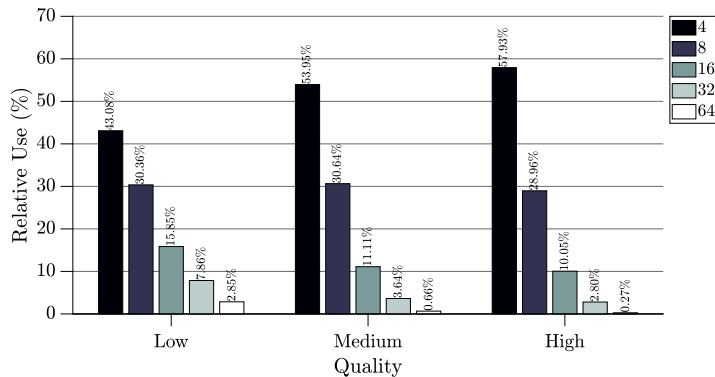


2019-12-16

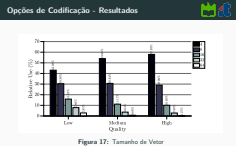
Apresentação Final

└ Transformadas em Codificação de Vídeo

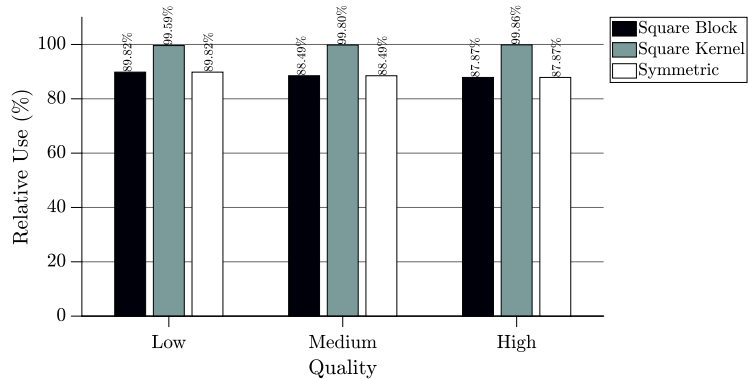
└ Opções de Codificação - Resultados



**Figura 17: Tamanho de Vetor**



- A dimensão mais comum é 4, e o seu uso aumenta com a qualidade pretendida
- Esperado, pois para uma dada area de imahem ser transformada com vetores de 1 por 4, tem que ser feitas mais trnaformações do que para vetores de dimensoes superiores



**Figura 18:** Kernel Simétrico

2019-12-16

Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Opções de Codificação - Resultados

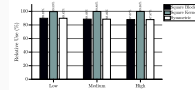


Figura 18: Kernel Simétrico

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



2019-12-16

Apresentação Final

└ Transformadas em Codificação de Vídeo

└ Opções de Codificação - Resultados

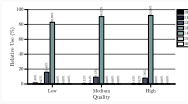
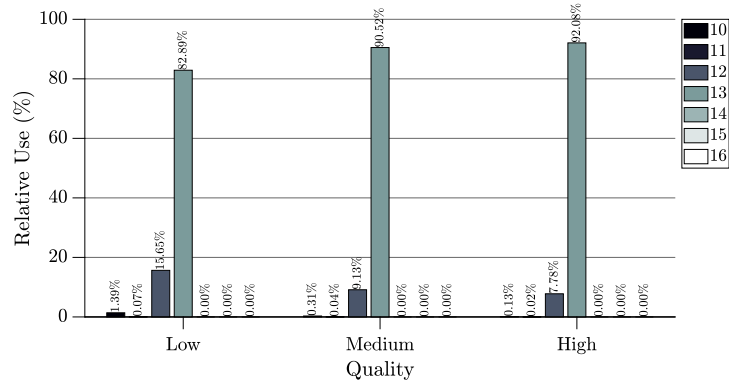
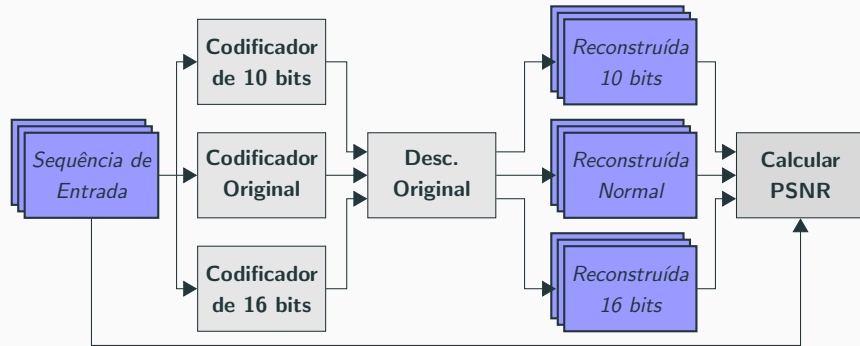


Figura 19: Número de Bits Utilizados nas Aproximações do Cosseno



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

- A maior parte das transformações usa 13 bits para as representações dos cossenos
- À medida que a qualidade aumenta, também aumenta o número de bits no cosseno



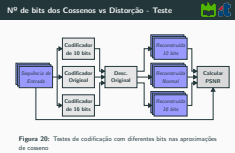
**Figura 20:** Testes de codificação com diferentes bits nas aproximações de cosseno

2019-12-16

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
- Decodificação feita com o decodificador 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

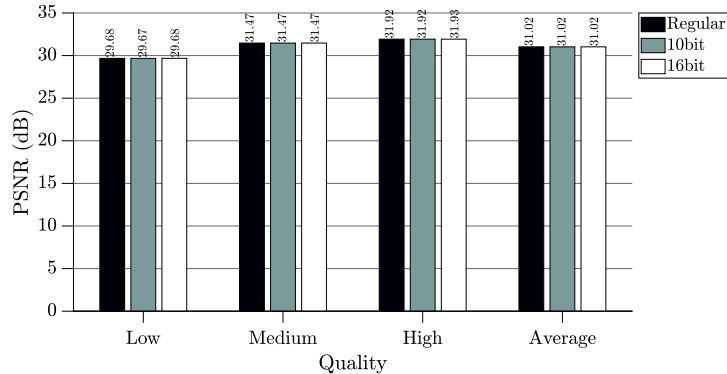


2019-12-16

Apresentação Final

Transformadas em Codificação de Vídeo

Nº de bits dos Cossenos vs Distorção - Resultados



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

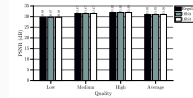


Figura 21: Comparação Distorção com Número de Bits usados no Cosseno

- 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

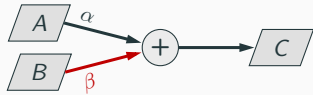


# Arquiteturas Desenvolvidas

---

## 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

$$M_{original} = 728 B$$

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

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

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

↓

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

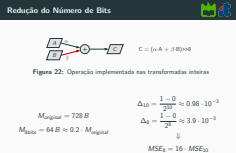
2019-12-16

Apresentação Final

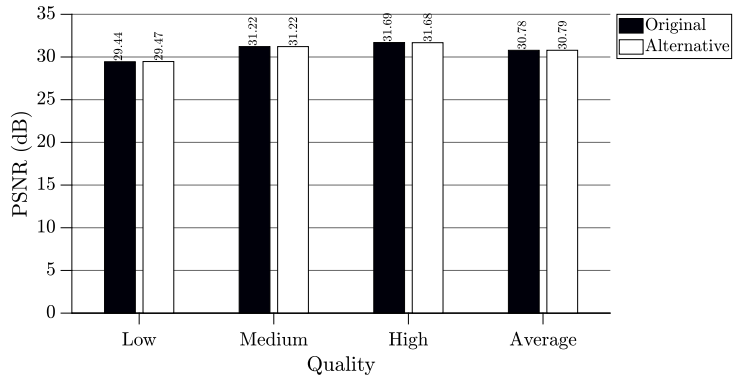
└ Arquiteturas Desenvolvidas

└ Software

└ Redução do Número de Bits



- 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

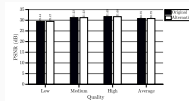


Figura 23: Comparação da distorção

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

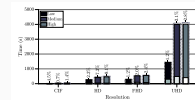


Figura 24: Tempo de Codificação

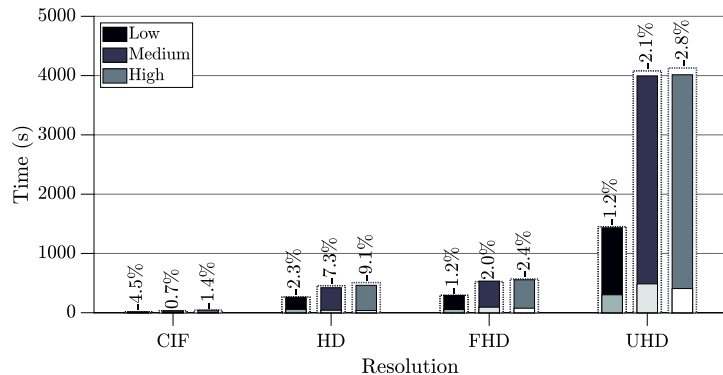


Figura 24: Tempo de Codificação

- 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

# Arquiteturas Desenvolvidas

---

## Hardware

2019-12-16

Apresentação Final

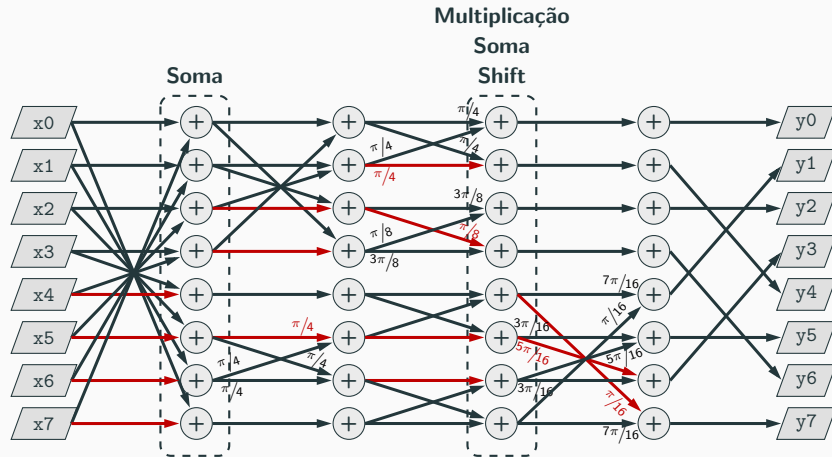
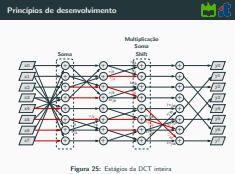
└─Arquiteturas Desenvolvidas

└─Hardware

Arquiteturas Desenvolvidas

Hardware

- Com o estudo do software de referencia feito
- Avanço para arquiteturas em hardware



**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

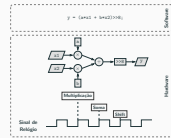
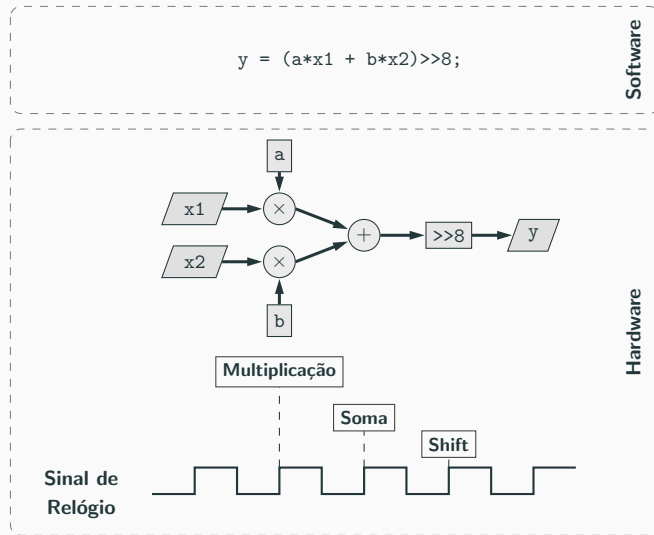
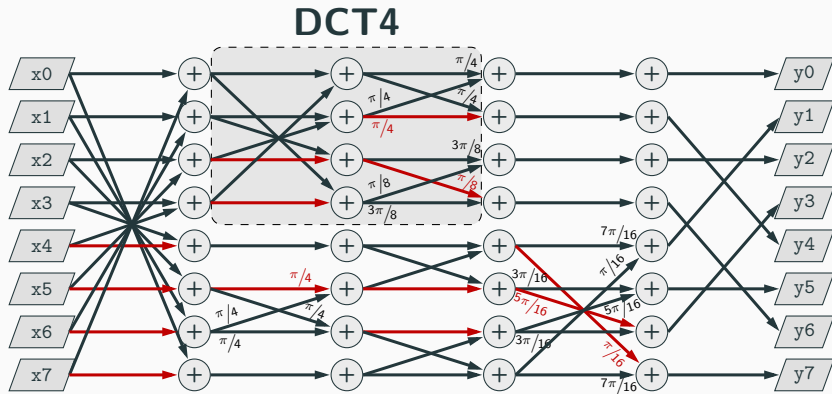


Figura 26: Implementação de operação de software em hardware



**Figura 26:** Implementação de operação de software em hardware

- 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



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

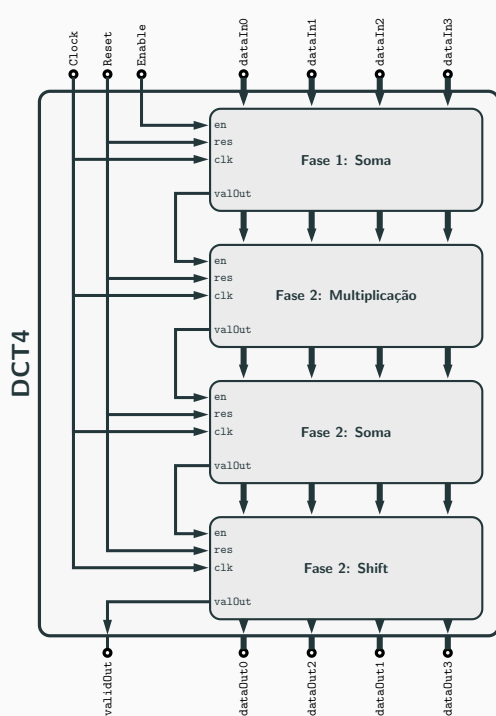
- Operações das DCTs mais pequenas são contidas nas DCTs de maiores dimensões
- Comportamento replica-se para os restantes tamanhos
- Permite repetição de hardware, simplificando o desenvolvimento



2019-12-16

## Apresentação Final

- Arquiteturas Desenvolvidas
  - Hardware



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

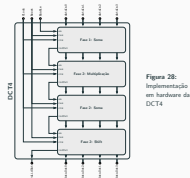
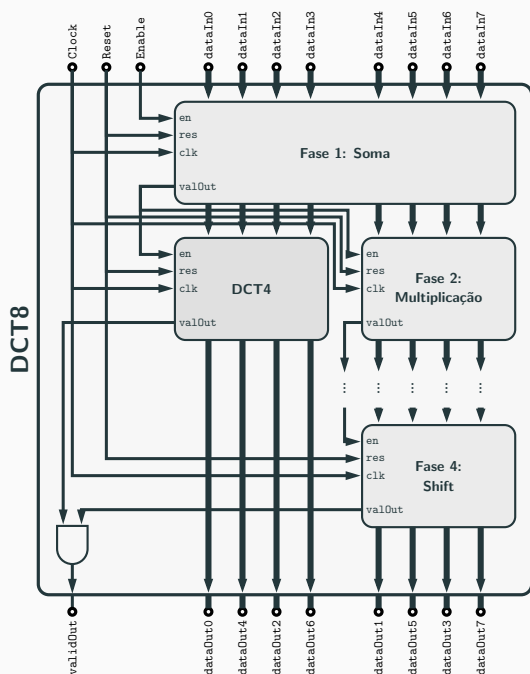


Figura 28:  
Implementação  
em hardware da  
DCT4

- 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

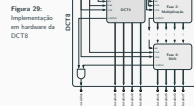


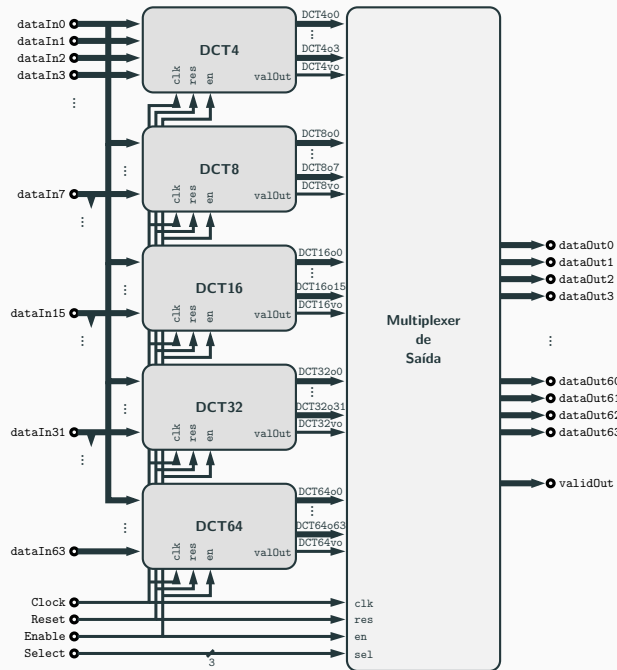
2019-12-16

## Apresentação Final

- └ Arquitecturas Desenvolvidas
  - └ Hardware

- Segue o mesmo molde da arquitetura anterior, mas DCT4 é incluída internamente
- validOut dependente do ultimo estágio interno, bem como da DCT4



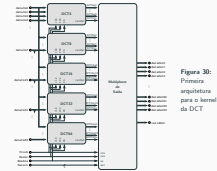


**Figura 30:**  
Primeira  
arquitetura  
para o kernel  
da DCT

2019-12-16

## Apresentação Final

- Arquiteturas Desenvolvidas
- Hardware



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

- 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

Tamanho da DCT	Utilização Lógica	
	Slice LUTs	Slice Registers
4	1125	636
8	2428	2087
16	7103	5702
32	19148	14257
64	45996	34146
Wrapper	75805	58370

2019-12-16

Apresentação Final

└─Arquiteturas Desenvolvidas

└─Hardware

└─Primeira arquitetura - Resultados



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

Tamanho da DCT	Utilização Lógica	
	Slice LUTs	Slice Registers
4	1125	636
8	2428	2087
16	7103	5702
32	19148	14257
64	45996	34146
Wrapper	75805	58370

- Tamanhos superiores ocupam muita área
- Arquitetura de tamanho elevado, devido à repetição de hardware



2019-12-16

Apresentação Final

└ Arquiteturas Desenvolvidas

└ Hardware

└ Primeira arquitetura - Resultados

Primeira arquitetura - Resultados

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

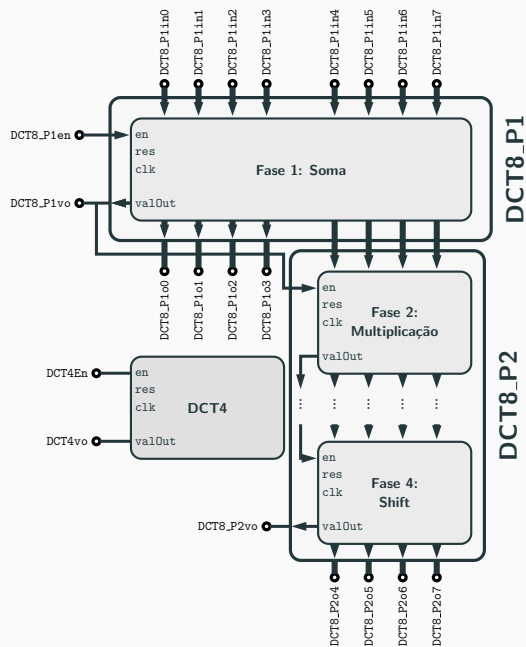
Resolução	Frequência (MHz)
1280 × 720	83
1920 × 1080	187
3840 × 2160	746
7680 × 4320	2986

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

Resolução	Frequência (MHz)
<b>1280 × 720</b>	83
<b>1920 × 1080</b>	187
<b>3840 × 2160</b>	746
<b>7680 × 4320</b>	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

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



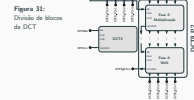
2019-12-16

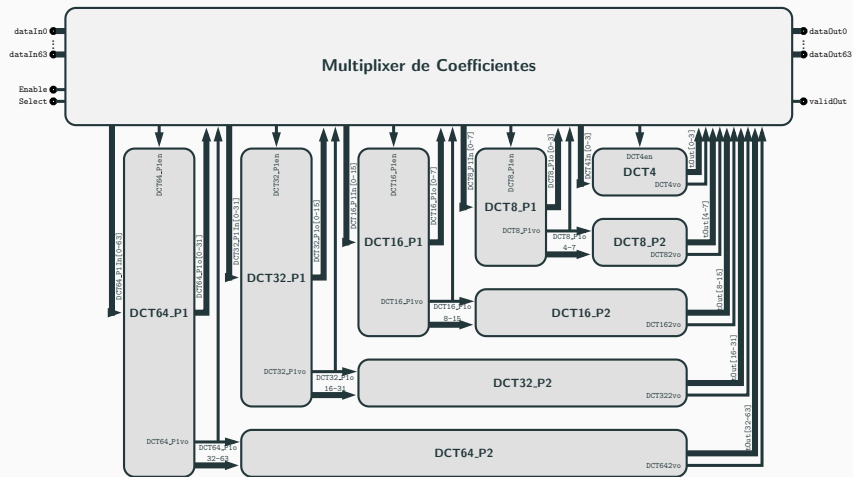
## Apresentação Final

### Arquiteturas Desenvolvidas

#### Hardware

- 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
- arquitetura fica com tamanho semelhante a DCT64





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

2019-12-16

## Apresentação Final

└─Arquiteturas Desenvolvidas

└─Hardware

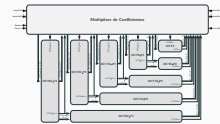


Figura 32: Segunda arquitetura para o kernel da DCT

- 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

Bloco	Utilização Lógica	
	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

2019-12-16

## Apresentação Final

└─Arquiteturas Desenvolvidas

└─Hardware

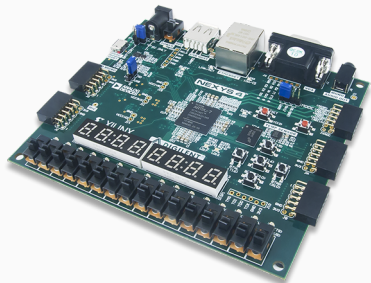
└─Segunda arquitetura - Resultados

- 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

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

Bloco	Utilização Lógica	
	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





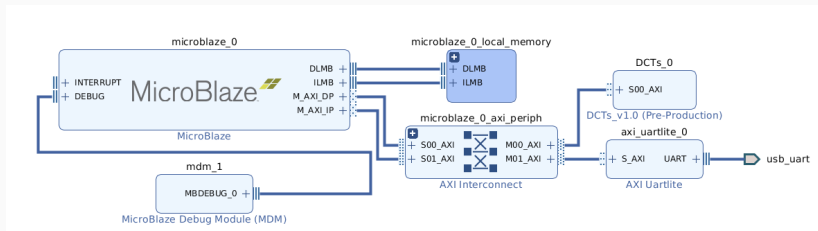
**Figura 33:**  
Placa *Nexys 4*  
da *Digilent*



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

**Figura 34:** Diagrama de blocos 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



**Figura 34:** Diagrama de blocos implementado



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

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

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

Tamanho de Bloco	Resolução			
	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

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

Tamanho de Bloco	Resolução			
	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

- 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

- Objetivos parcialmente atingidos
- Otimização do software de referência com estudo das características de codificação do AV1
- Contrução 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!

2019-12-16

Apresentação Final

└─ Conclusões e Trabalho Futuro

Obrigado!



# Discussão

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

Miguel Oliveira Inocêncio

Professor  
**Armando Pinho**  
Presidente do Júri

Professor  
**Pedro Assunção**  
Arguente Principal

Professor  
**António Navarro**  
Orientador

2019-12-16

Apresentação Final  
└─ Conclusões e Trabalho Futuro

### Discussão

Co-processador da Transformada para o  
Codificador de Vídeo AV1

Miguel Oliveira Inocêncio

Professor  
Armando Pinho  
Presidente do Júri

Professor  
Pedro Assunção  
Arguente Principal

Professor  
António Navarro  
Orientador

