Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Sistemas e Computação

Dissertação de Mestrado

Gerador de Vídeo Pseudo-Aleatório Sintético na Verificação Funcional de Circuitos Decodificadores de Vídeo MPEG-4

George Sobral Silveira

Elmar U. K. Melcher
(Orientador)

Campina Grande
Junho - 2008

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Sistemas e Computação

Gerador de Vídeo Pseudo-Aleatório Sintético na Verificação Funcional de Circuitos Decodificadores de Vídeo MPEG-4

George Sobral Silveira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Elmar U. K. Melcher
(Orientador)

Campina Grande, Paraíba, Brasil.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S587g
2008 Silveira, George Sobral.
Gerador de vídeo pseudo-aleatório sintético na verificação funcional de circuitos decodificadores de vídeo MPEG-4 / George Sobral Silveira. ─ Campina Grande, 2008.
52f. : il.

Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientador: Dr. Elmar U. K. Melcher.

1. Verificação Funcional. 2. Decodificador de Vídeo. 3. MPEG-4. 4. Cobertura Funcional. I. Título.

CDU – 004.3'142.23 (043)

Folha de Assinaturas

"GERADOR DE VÍDEO PSEUDO-ALEATÓRIO SINTÉTICO NA VERIFICAÇÃO FUNCIONAL DE CIRCUITOS DECODIFICADORES DE VÍDEO MPEG-4"

GEORGE SOBRAL SILVEIRA

DISSERTAÇÃO APROVADA EM 20.06.2008

PROF. ELMAR UWE KURT MELCHER, Dr.
Orientador

PROFa. JOSEANA MACÊDO FECHINE, D.Sc
Examinadora

PROF. LEONARDO VIDAL BATISTA, D.Sc
Examinador

CAMPINA GRANDE - PB

Resumo

O advento das novas tecnologias de VLSI *(Very Large Scale Integration)* e metodologias de projetos de SoC (*System on Chip*), provocou um crescimento explosivo na complexidade de circuitos eletrônicos modernos. Como resultado, a verificação funcional se tornou o principal “gargalo” no fluxo de projetos digitais. A verificação funcional possui dois problemas principais: escolha de bons estímulos e medida de cobertura. As indústrias vêm lançando nos últimos anos diversos dispositivos eletrônicos, capazes de manipular vídeos digitais. Os codificadores e decodificadores de vídeo vêm se tornando cada vez mais comuns entre os consumidores, popularizando o conceito da compressão de vídeo. Para realização da verificação funcional de um decodificador de vídeo, é necessário que o estímulo utilizado no processo esteja no formato de um vídeo digital codificado, este vídeo deve possuir as características necessárias para estimular adequadamente o circuito digital do decodificador. Esta dissertação de mestrado busca auxiliar na solução de problemas ocorridos na verificação funcional de IP-Cores de decodificadores de vídeos. Durante esse trabalho, foi desenvolvido um gerador de vídeo chamado de RandMovie, capaz de produzir vídeos sintéticos aleatórios, a parti do qual o projetista será capaz de gerar vídeos com estímulos direcionados para verificação funcional com cobertura.

Abstract

The advent of the new VLSI technology and SoC design methodologies has brought about an explosive growth to the complexity of modern electronic circuits. As a result, the functional verification has become the main bottleneck in any digital design flow. The functional verification has two major problems: choose good stimuli and measure coverage. The industries are producing in the last years several electronic devices, capable of manipulating digital movie. The movie encoders and decoders are becoming more and more common among the consumers, popularizing the concept of the movie compression. In order to implement the functional verification of a movie decoder, the stimuli should be in a coded digital movie format. This movie should have the necessary characteristics to stimulate the decoder digital circuit properly. This work presents a solution for the problems that appeared during the functional verification of movie decoders IP-Cores. A movie generator called RandMovie has been developed; it is capable of producing random synthetic movies. With this movie generator the engineer will be capable of generating movies to be used as stimuli in the functional verification with coverage.

Agradecimentos

Agradeço primeiramente a Deus, pela minha vida e por mais esta conquista.

Meus agradecimentos especiais aos meus pais, Geraldo e Nevinha a quem eu devo tudo e dedico esse trabalho, aos meus irmãos (ã) Gilson e Júnior e Giovanna, as minhas cunhadas, sobrinhas e sobrinho.

Quero agradecer com muito carinho a minha namorada Ravely “Maguinha”, obrigado por estar ao meu lado todos os dias dessa jornada, pela sua enorme paciência durante os longos relatórios que tive que escrever aos domingos.

Obrigado ao meu orientador professor Elmar, vulgo “Alemão” entre seus alunos, pelo incentivo, dedicação, amizade e pelo envolvimento em todo o processo da minha dissertação, e obrigado a professora Joseana, pela participação no meu mestrado.

Aos integrantes do CAFA-CLUB, Wilson, Henrique, Isaac, Zurita, Osman, Fagner, Matheus e aos demais integrantes do LAD, Karinona, Fabrício, Sergio, Helder, Jorgeluiz, Dani, Ana Karina, Maria, Anderson e Uyara, a todos vocês, muito obrigado pelas contribuições durante a execução do meu trabalho, realizadas dentro do LAD, bares e botequins, lugares que proporcionaram ótimos momentos de descontração. Um agradecimento extra a Karinona, pela ajuda prestada durante todo o trabalho e pela parceria na publicação dos resultados obtidos neste trabalho.

Agradeço aos integrantes do QUEEN, em especial aos meus amigos Leandro e Raphael que me auxiliaram em diversas questões extraordinárias discutidas no Parque do Povo.

Agradeço aos meus amigos Custódio e Wagner. Quero agradecer em especial ao meu grande amigo Tibério, pela sua amizade e apoio em diversas fases da minha vida.

A Aninha, da COPIN, sempre disposta a ajudar e resolver todos os problemas que ocorreram durante o mestrado.

As pessoas que não tiveram o nome citado aqui, mas que de alguma forma também contribuíram para esse trabalho, meu muito obrigado.

Obrigado!

Índice

Lista de Abreviaturas X
Lista de Figuras..... XII
Lista de Tabelas..... XIV

CAPÍTULO 1 – Introdução 1
1.1 Objetivos 4
1.1.1 Objetivos Específicos..... 5
1.2 Estrutura da Dissertação..... 6

CAPÍTULO 2 – Fundamentação Teórica 7
2.1 Verificação Funcional..... 7
2.2 Metodologia VeriSC 8
2.2.1 *Testbench* 8
2.3 Cobertura 10
2.3.1 Biblioteca BVE-COVER 11
2.3.1.1 BVE_COVER_Bucket..... 11
2.4 Vídeos Digitais 12
2.4.1 Compressão de vídeos 12
2.4.1.1 Redundância espacial 13
2.4.1.2 Redundância Temporal 15
2.4.2 Descompressão de vídeos..... 18
2.5 Considerações Gerais..... 19

CAPÍTULO 3 - Caracterização do Problema 21
3.1 Circuito Decodificador de Vídeo..... 21
3.2 Contextualização do Problema 24

3.3 Principais Causas do Problema 25
3.4 Trabalhos Relacionados 26
3.5 Considerações Gerais 28

CAPÍTULO 4 - Descrição do "RandMovie" 29
4.1 Arquitetura do RandMovie 29
4.2 Estratégia para Porta DCDCT do BS 30
4.2 Estratégia para Porta DVM 32
4.2 Combinação das Estratégias 35
4.4 Implementação do RandMovie 37
4.5 Considerações Gerais 37

CAPÍTULO 5 - Apresentação e Análise dos Resultados 39
5.1 Estudo de Caso: Aplicação do RandMovie ao Testbench do MPEG-4 39
5.2 Avaliação dos Resultados 40
5.2.1 Aplicação da Aleatoriedade Direcionada 40
5.2.2 Taxas de Cobertura nas Portas do BS 42
5.2.3 Custo computacional do RandMovie 45
5.2.4 Exposição de Falha no Circuito 46
5.2.5 Considerações Gerais 47

CAPÍTULO 6 – Considerações Finais 48
6.1 Sugestões para Trabalhos Futuros 49

REFERÊNCIAS BIBLIOGRÁFICAS 50

Lista de Abreviaturas

ACDCPI	AC e DC Predição Intra
ARM	*Advanced RISC Machine*
BS	*Bitstream*
BVE-COVER	*Brazil-IP Verification Extension – Coverage*
CBP	Copiador de Blocos Predição
CI	Circuito Integrado
CMOS	*Complementary Metal Oxide Semiconductor*
CODEC	*Compression / Decompression*
DCT	*Discrete Cosine Transform*
DCDCT	Decodificação de Coeficiente para *Discrete Cosine Transform*
DUV	*Design Under Verification*
DVM	Decodificação de Vetor de Movimento
EM	Estimação de Movimento
FIFO	*First In First Out*
FPGA	*Field Programmable Gate Array*
IBM	*International Business Machines*
IDCT	Inversa da DCT
IEC	*International Electrotechnical Commission*
IP-Core	*Intellectual Property*
ISO	*International Organization for Standardization*
JPEG	*Joint Photographic Experts Group*
MPEG	*Moving Picture Experts Group*
QI	Quantização Inversa
RGB	*Red Green Blue*
RISC	*Reduced Instruction Set Computer*
RTL	*Register Transfer Level*
SDRAM	*Static Dinamic Random Access Memory*
SI	*Scan* Inverso
SoC	*System on Chip*
SUM	Somador
TLM	*Transaction Level Modelling*

VLC	*Variable Length Codes*
VLSI	*Very Large Scale Integration*
VSA	Vídeo Sintético Aleatório
YCbCr	*Y is Luma, Chroma blue, Chroma red*

Lista de Figuras

Figura 1.1: Etapas para o desenvolvimento de hardware. 1
Figura 1.2: Quadro do vídeo sintético aleatório. 4
Figura 2.1: Estrutura do *testbench*. 9
Figura 2.2: Redundâncias de informações temporal e espacial. 13
Figura 2.3: Exemplos de regiões homogêneas. 14
Figura 2.4: Imagem após processo de suavização da textura. 14
Figura 2.5: Concentração de energia após DCT. 15
Figura 2.5: Quadro n do filme *Star Wars*. 16
Figura 2.6: Quadro n+1 do filme *Star Wars*. 16
Figura 2.7: Busca do bloco semelhante no quadro de referência. 17
Figura 2.8: Geração do vetor de movimento. 17
Figura 2.9: Decodificação do quadro. 18
Figura 2.10: Esquemático de decodificador de vídeo. 19
Figura 3.1: Esquemático do decodificador de vídeo MPEG-4. 21
Figura 3.2: Esquemático *Bitstream*. 22
Figura 3.3: Processo de quantização. 26
Figura 4.1: Arquitetura do RandMovie. 30
Figura 4.2: Padrões de Base, blocos 8x8. 31
Figura 4.3: Quadro gerado. 31
Figura 4.4: Janela de busca e macroblocos semelhantes. 32
Figura 4.5: Janela de busca. 33
Figura 4.6: Quadro particionado em quadrantes 48x48 *pixels*. 33
Figura 4.7: Regiões do quadrante. 34
Figura 4.8: Inserção de macroblocos e janela de busca. 34
Figura 4.9: Compensação de movimento. 35
Figura 4.10: Combinação das estratégias. 36
Figura 5.1: RandMovie e decodificador aplicado ao *testbench*. 40
Figura 5.2: Arquivo de *log* de uma simulação. 41
Figura 5.3: Cobertura nas portas de saída do BS utilizando VSA, Vídeos Naturais e o RandMovie. 43
Figura 5.4: Cobertura entre VSA, Vídeos Naturais e o RandMovie. 44

Figura 5.5: Cobertura entre VSA, Vídeos Naturais e o RandMovie. 45
Figura 5.6: Custo computacional para codificação do vídeo. 46
Figura 5.7: Diagrama de blocos do MPEG-4 com indicação do erro. 46

Lista de Tabelas

Tabela 1.1: Cobertura individual dos blocos do MPEG-4. 3
Tabela 3.1: Cobertura do módulo BS do MPEG-4. 24
Tabela 3.2: Cobertura das portas de saída de BS. 24
Tabela 5.1: Portas do BS e suas variáveis. 42

CAPÍTULO 1 – Introdução

O projeto lógico de um CI (Circuito Integrado), possui um fluxo de criação composto por diversas fases. Não se tem um consenso entre os trabalhos da área sobre os elementos conceituais de cada fase. No entanto, para que o leitor possa se orientar com relação ao processo de desenvolvimento de hardware, as fases podem ser descritas da seguinte forma: especificação do hardware, especificação da verificação funcional, implementação do *testbench*, implementação do DUV (*Design Under Verification*), verificação funcional, síntese, simulação pós-síntese, prototipação e SoC (*System on Chip*). Na Figura 1.1 está apresentado o diagrama do fluxo de desenvolvimento de hardware.

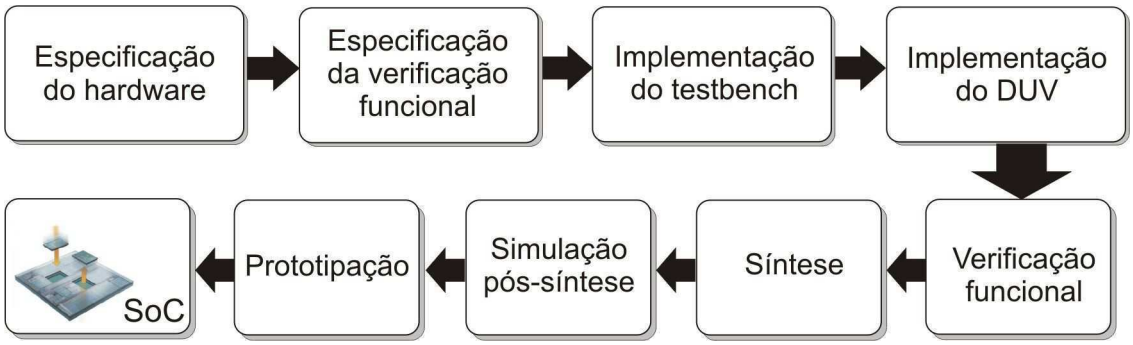

Figura 1.1: Etapas para o desenvolvimento de hardware.

A verificação funcional pode ser definida como: "um processo usado para demonstrar que o objetivo do projeto é preservado em sua implementação" [14]. Esta fase é responsável pelo maior consumo de recursos dentro do fluxo de desenvolvimento de um SoC. Na literatura, é visto que 70% de todos os recursos de um projeto são envolvidos neste processo [3].

Um grande desafio dentro da verificação funcional é a aplicação de estímulos adequados para exercitar as funcionalidades especificadas. Algumas técnicas tentam resolver este problema usando geradores de estímulos aleatórios [4] [5], pois

a aleatoriedade introduz o fator da imprevisibilidade, criando condições relevantes que não foram previstas na especificação do projeto. A criação destes estímulos pode ser automatizada com o desenvolvimento de uma fonte aleatória. Quando geradores de estímulos aleatórios não atingem os resultados almejados, pode-se acrescentar restrições à fonte geradora aumentando, assim, a probabilidade de gerar os estímulos desejados.

Embora os geradores de estímulos aleatórios gerem freqüentemente resultados melhores do que os com valores e ordem pré-determinados, conhecidos como “estímulos direcionados”, eles não podem garantir que todas as funcionalidades especificadas foram exercitadas. Portanto, existe uma dificuldade para identificar quais funcionalidades não foram cobertas.

Em função deste problema, o projetista tem que usar do artifício da cobertura junto com os estímulos. A cobertura, em seu senso mais abrangente, é responsável por medir o progresso da verificação por uma métrica especificada pelo engenheiro responsável pela especificação do circuito.

A compactação de imagens e vídeos é um tema de pesquisa crescente. Nos últimos 20 anos, diversos sistemas e algoritmos foram desenvolvidos para compressão e descompressão. Para encorajar os estudos, foram definidos métodos e padrões para que produtos de fabricantes distintos pudessem comunicar-se efetivamente. Isto conduziu ao desenvolvimento de vários padrões internacionais como JPEG e MPEG [21].

O Padrão MPEG-4 da ISO/IEC, pertencente à família MPEG, vem apresentando sucesso em aplicações gráficas interativas (conteúdo sintético) e aplicações multimídia interativas [19]. Conteúdo sintético são imagens produzidas por computadores.

Foi desenvolvido na UFCG (Universidade Federal de Campina Grande) um *IP-core* de um circuito decodificador de vídeo no formato MPEG-4 *Simple Profile Level 0*. O decodificador foi validado em silício, possuindo as seguintes características: 22.7mm² em tecnologia CMOS (*Complementary Metal Oxide Semiconductor)* de 0.35µm com 4 níveis de metal e freqüência operacional de 25 MHz [2].

Para realizar a verificação do decodificador MPEG-4 foi utilizada a metodologia VeriSC, junto com a biblioteca de cobertura BVE-COVER [7] [8]. A

verificação foi realizada em nível hierárquico, o IP-Core do MPEG-4 foi dividido em módulos, verificados separadamente.

O decodificador de vídeo contém os seguintes módulos:

- BS (*Bitstream*);
- DVM (Decodificação de Vetor de Movimento);
- CBP (Copia de Blocos Predição);
- DCDCT (Decodificação de Coeficiente para DCT);
- SI (*Scan* Inverso);
- ACDCPI (AC e DC Predição para Intra);
- QI (Quantização Inversa);
- IDCT (Inversa da DCT)
- SUM (Somador);
- RGB (Conversor YCbCr para RGB);

Na Tabela 1.1 pode-se verificar as taxas de cobertura de cada módulo. A primeira coluna representa os módulos do decodificador e a segunda coluna a porcentagem da cobertura atingida por cada bloco. A maioria dos blocos alcançou 100% de cobertura durante o processo de verificação, o BS (*Bitstream*) foi o único que não apresentou essa cobertura.

Tabela 1.1: Cobertura individual dos blocos do MPEG-4.

Módulo	Cobertura Acumulada Referente aos Estímulos Aleatórios (%)
BS	**67**
DVM	**100**
CBP	**100**
DCDCT	**100**
SI	**100**
ACDCPI	**100**
QI	**100**
IDCT	**100**
SUM	**100**
RGB	**100**

Os VSA (Vídeos Sintéticos Aleatórios) utilizados como estímulos no BS foram produzidos da seguinte forma: os quadros gerados possuíam *pixels* com valores

introduzidos aleatoriamente, esses valores se encontravam dentro do intervalo [0,255], os quais eram codificados em MPEG-4 com a ajuda do codificador Xvid [9]. Este codificador foi escolhido por possuir as características: Código fonte aberto, implementado em C++, compatível com SystemC e implementa todas as funcionalidades do padrão MPEG-4 [1].

Os vídeos resultantes da codificação possuíam imagens coloridas sem sentido, os quadros poderiam ser comparados com uma imagem colorida emitida por uma televisão estando “fora do ar”. Na Figura 1.2 é apresentado um quadro extraído de um vídeo produzido pelo VSA.

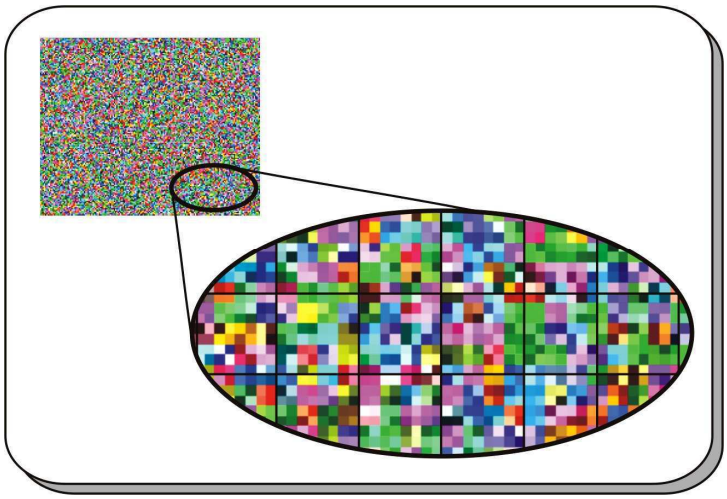

Figura 1.2: Quadro do vídeo sintético aleatório.

Diante da baixa cobertura atingida pelo BS, fica evidente a existência de um problema de cobertura detectado durante a verificação funcional do circuito decodificador de vídeo MPEG-4 [2].

1.1 Objetivos

Este trabalho demonstrar a aplicação de estratégias e técnicas na produção de vídeos sintéticos para que estes sejam utilizados como estímulos, proporcionando a diminuição dos esforços gastos no processo de verificação funcional. Estes esforços são devido às dificuldades encontradas pelos engenheiros no processo de verificação funcional dos decodificadores de vídeos, e a alta complexidade

relacionada à busca por encontrar estímulos adequados que proporcionem a obtenção da cobertura especificada.

Os benefícios serão alcançados com a automatização do processo de geração de vídeo e adição da aleatoriedade aos mesmos, graças à aplicação adequada da aleatoriedade, que adicionará o fator da imprevisibilidade, característica que é buscada na geração de estímulos utilizados no processo de verificação funcional.

O gerador de vídeo proporcionará aos projetistas de decodificadores de vídeos a flexibilidade, simplicidade e diminuição dos esforços gastos durante o processo de verificação funcional, com a obtenção das taxas desejáveis de cobertura, de tal forma que ao final da verificação o circuito digital esteja de acordo com as especificações do projeto.

O gerador de vídeo randômico sintético é denominado RandMovie, este poderá ser acoplado ao *testbench* do *IP-Core* de decodificadores MPEG-4 e produzir vídeos em "tempo simulado". Neste caso, na medida em que cada quadro for sendo decodificado pelo *IP-Core,* o RandMovie gerará novos quadros.

Outra característica do RandMovie é sua capacidade de armazenar os vídeos produzidos para serem reutilizados posteriormente em testes de regressão.

1.1.1 Objetivos Específicos

Os objetivos específicos desse trabalho podem ser descritos da seguinte forma:

- Apresentar as dificuldades da etapa de verificação funcional de um decodificador de vídeo.
- Apresentar o paradigma dos vídeos digitais, demonstrando suas características de implementação;
- Apresentar o conceito das técnicas do RandMovie e uma forma de aplicar eficientemente a aleatoriedade;
- Desenvolver e apresentar uma técnica capaz de produzir vídeos sintéticos aleatórios, que possam ser utilizados como estímulos no processo de verificação funcional, proporcionando uma taxa de cobertura satisfatória.

- Implementar o algoritmo relativo a essa técnica e sua utilização em um *testbench*.

1.2 Estrutura da Dissertação

Esta dissertação está estruturada da seguinte forma:

- **Capítulo 2**: Fundamentação teórica sobre os conceitos básicos de vídeos digitais e verificação funcional, para que os leitores possam compreender o restante do trabalho.

- **Capítulo 3**: Caracterização do problema encontrado pelos engenheiros de verificação funcional de decodificadores de vídeo.

- **Capítulo 4**: Metodologia utilizada para o desenvolvimento do RandMovie.

- **Capítulo 5**: Um estudo de caso e apresentação dos resultados obtidos.

- **Capítulo 6**: Considerações finais e sugestões para trabalhos futuros.

CAPÍTULO 2 – Fundamentação Teórica

Neste capítulo serão abordados alguns conceitos básicos sobre o processo de verificação funcional, a metodologia utilizada, assim como, a métrica para quantificar a eficiência da verificação e os componentes necessários para realização da verificação funcional, e o paradigma sobre vídeos digitais.

2.1 Verificação Funcional

O conceito de verificação funcional adotado nesse trabalho se baseia no livro de Bergeron [14]. A verificação funcional é um processo que verifica, por meio de simulação, se o dispositivo sendo verificado está isento de erros. Ela acompanha o dispositivo em busca de uma verificação completa de todas as funcionalidades especificadas. O ponto de partida de um projeto de verificação é que exista uma especificação, considerada o modelo ideal e esta deve ser respeitada durante toda a fase de projeto.

No processo de verificação, o modelo considerado ideal (Modelo de Referência) é simulado juntamente com o dispositivo sendo verificado e os resultados são comparados.

Um aspecto importante é decidir a granularidade do circuito a ser verificado, ou seja, definir o quanto será subdividido o circuito. Quando se faz a verificação de um projeto, ele é normalmente dividido em blocos funcionais para facilitar a sua verificação. O processo de verificação funcional torna-se mais complexo quando empregado em um bloco muito grande, por ser mais difícil encontrar um erro ao fazer a simulação do projeto inteiro de uma única vez. Quando o projeto é complexo, ao aparecer um erro, é necessário fazer uma investigação de todo o modelo para descobrir onde está a causa desse erro.

A verificação funcional utiliza simulação para verificar o *IP-Core* denominado neste trabalho como DUV. Para que essa simulação seja possível, é necessário que haja um ambiente de verificação que possa receber o DUV, inserir estímulos e comparar suas respostas com as respostas de um modelo ideal. Esse ambiente é denominado de *testbench*.

2.2 Metodologia VeriSC

A metodologia VeriSC adotado neste trabalho é composta de um fluxo de verificação, que não se inicia pela implementação do DUV. Nesse fluxo, a implementação do *testbench* e do Modelo de Referência antecedem a implementação do DUV. Para permitir que o *testbench* seja implementado antes do DUV, a metodologia implementa um mecanismo para simular a presença do DUV com os próprios elementos do *testbench*, sem a necessidade da geração de código adicional, que não seja reutilizável.

VeriSC é uma metodologia que adota o conceito de projetos com hierarquia, de forma que um projeto possa ser dividido em partes a serem implementadas e verificadas. Ela pode ser aplicada para a verificação funcional de DUV's digitais síncronos que utilizam um único sinal de relógio, podendo ser utilizado em circuito que possuam mais de um relógio, se este puder ser dividido em vários subcircuitos com apenas um relógio [7].

Essas características fazem com que a metodologia permita a criação de um *testbench* compatível com o estado da arte descrita na literatura em meados do ano 2003 [14] e dos cenários de testes que possibilitem exercitar as funcionalidades especificadas em um plano de verificação [7].

2.2.1 *Testbench*

O *testbench* é o ambiente de simulação no qual o DUV é inserido para realizar a verificação. Esse ambiente estimula o DUV e compara suas respostas com as respostas de um modelo ideal. O *testbench* deve ser implementado no nível de transações, que é uma transferência de informações de alto nível, de dados ou controle entre o *testbench* e o DUV sobre uma interface.

Um bom *testbench* deve possuir as seguintes características básicas: ser dirigido por cobertura, possuir aleatoriedade direcionada, ser auto-verificável e baseado em transações [14].

A composição do *testbench* utilizada neste trabalho pode ser vista na Figura 2.1. O *testbench* é a parte da figura que está dentro da área cinza. Ele é composto pelos seguintes blocos: *Source*, TDriver(s), TMonitor(es), Modelo de Referência e *Checker*. Cada interligação dos blocos do *testbench* é viabilizado por FIFO(s) (*First In First Out*), representadas na figura como setas mais largas. O *testbench* é ligado ao DUV por interfaces formadas por sinais, representados na figura por setas mais finas.

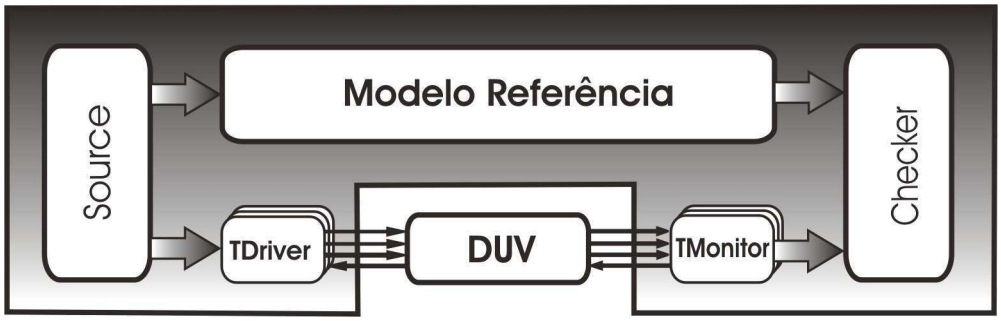

Figura 2.1: Estrutura do *testbench*.

- As FIFOs exercem uma tarefa muito importante no *testbench*, pois elas são responsáveis por controlar o seqüenciamento e o sincronismo dos dados que entram e saem delas. Os dados da FIFO devem ser retirados em seqüência e comparados, de forma que sempre estarão na ordem correta de comparação.
- O DUV é o projeto que está sendo verificado e deve ser implementado em RTL (*Register Transfer Level*), ou seja, no nível de sinais, necessitando assim de algum mecanismo para se comunicar com o *testbench* que trabalha em TLM (*Transaction Level Modelling*). Essa comunicação se faz com a ajuda dos blocos TDriver(s) e TMonitor(es) que traduzem sinais para transações e vice-versa.
- O *Source* é responsável por criar estímulos para a simulação. Esses estímulos devem ser cuidadosamente escolhidos para satisfazer os critérios de cobertura especificados. Todos os estímulos criados pelo *Source* são transações.

- O TDriver é responsável por converter as transações, recebidas pelo *Source*, em sinais e submetê-los para o DUV além de executar o protocolo de comunicação (*handshake*) com o DUV, por meio de sinais.
- O TMonitor executa o papel inverso do TDriver, converte todos os sinais de saída do DUV para transações e repassa as mesmas para o Checker, via FIFO. Além disso, executa um protocolo de comunicação com o DUV, por meio de sinais.
- O *Checker* é o responsável por comparar as respostas resultantes do TMonitor e Modelo de Referência, para saber se são equivalentes.
- O Modelo de Referência contém a implementação ideal (especificada) do sistema. Portanto, ao receber estímulos, ele deve produzir respostas corretas.

2.3 Cobertura

A Cobertura Funcional é uma técnica usada para medir o progresso da simulação e reportar quais funcionalidades não foram testadas, podendo ajudar no monitoramento da qualidade de testes e direcioná-los para criar geradores que cubram áreas que não foram adequadamente testadas.

Na verificação funcional, um grande problema é descobrir o momento de parar a simulação de forma que tenha sido coberto todo o plano de verificação, ou seja, verificado totalmente o DUV. Esse problema é decorrente do fato de que a simulação em si não consegue afirmar se todas as funcionalidades implementadas foram verificadas.

A principal técnica para mostrar que uma simulação obteve sucesso é analisar a cobertura. Isso significa criar uma lista de tarefas (funcionalidades especificadas) a serem testadas e observar se cada uma dessas funcionalidades foi testada durante a fase de simulação.

O trabalho, utiliza a abordagem *black-box*. Dessa forma, a verificação é realizada por meio das interfaces disponíveis sem o conhecimento da estrutura interna do DUV. Portanto, a cobertura realizada no *testbench* pode ser medida nos seguintes blocos:

- **TDriver**: Quando é necessário medir os dados na entrada verificando se o *Source* aleatório está gerando os dados necessários para o dispositivo.

- **TMonitor**: Quando é necessário medir dados de saída do DUV. Os requisitos de saída especificam o conjunto completo de dados a serem observados.
- **Modelo de Referência**: No caso de se desejar medir funcionalidades internas do dispositivo no nível de transação.

2.3.1 Biblioteca BVE-COVER

A biblioteca de cobertura funcional BVE-COVER tem o objetivo de encontrar buracos de cobertura e mostrar o progresso da simulação. Ela é composta de 4 componentes básicos: BVE-COVER (Bucket, Illegal, Ignore e Crosscoverage) [22]. No trabalho foi utilizado a componente BVE-COVER_Bucket.

2.3.1.1 BVE_COVER_Bucket

De acordo com Silva [22], todas as funcionalidades importantes que precisam ser cobertas na simulação são especificadas usando o BVE_COVER_Bucket. A simulação depende da definição dos *buckets*, uma vez que existem duas formas de parar a simulação: por meio do enchimento de todos os *buckets* ou por meio da especificação de um limite de tempo na simulação.

As funcionalidades são especificadas usando a instrumentação de código, como mostrado no Código 1. O engenheiro de verificação deve especificar quais funcionalidades e quantas vezes devem ser executadas para alcançar 100% de cobertura. Por exemplo, no Código 1 ***intra_block==true*** é a funcionalidade desejada e o valor "**100**" é a freqüência de repetições.

Código 1: BVE-COVER BUCKET

```
01    BVE_COVER_BUCKET Cv_bucket_intra;
02    Cv_bucket_intra.begin( );
03          BVE_COVER_BUCKET(Cv_bucket_intra, (intra_block==true, 100));
04          BVE_COVER_BUCKET(Cv_bucket_intra, (intra_block==false, 100));
05    Cv_bucket_intra.end( );
```

Os eventos são coletados diretamente da simulação durante a execução. Cada vez que um evento acontece, a biblioteca faz uma atualização na lista de cobertura. O BVE_COVER_Bucket avalia os pontos de cobertura a cada novo evento da simulação. Portanto, uma vez que um ponto de cobertura é executado, ele atualiza os dados da porcentagem de cobertura calculando a porcentagem do total que foi coberto.

2.4 Vídeos Digitais

A adoção dos vídeos digitais pela indústria vem crescendo rapidamente. A cada dia são lançados no mercado produtos que fazem uso dos vídeos digitais, tais como aparelhos de celulares, filmadoras, etc. Existem problemas com a utilização deste tipo de vídeo, pois estes necessitam de grande quantidade de espaço em memória para serem armazenados e um *bitrate* (taxa de transferência de bits) considerável para serem transmitidos pela internet, por exemplo.

Para minimizar estes problemas, a indústria faz uso intensivo da compressão dos vídeos antes de serem armazenados ou transmitidos, provocando assim a diminuição dos dados necessários para representar a informação contida nos vídeos.

Os codificadores e decodificadores de vídeo vêm se tornando cada vez mais comuns entre os consumidores. As várias aplicações destas técnicas de manipulação de dados dos vídeos em dispositivos multimídias vêm popularizando o conceito da compressão de vídeo.

2.4.1 Compressão de vídeos

O paradigma adotado pela compressão de vídeo está relacionado às características do sistema visual humano, que é ineficiente para distinguir detalhes finos em uma imagem e perceber movimentos rápidos [21]. A partir da Figura 2.2, pode-se ver o enquadramento dos aspectos temporais e espaciais.

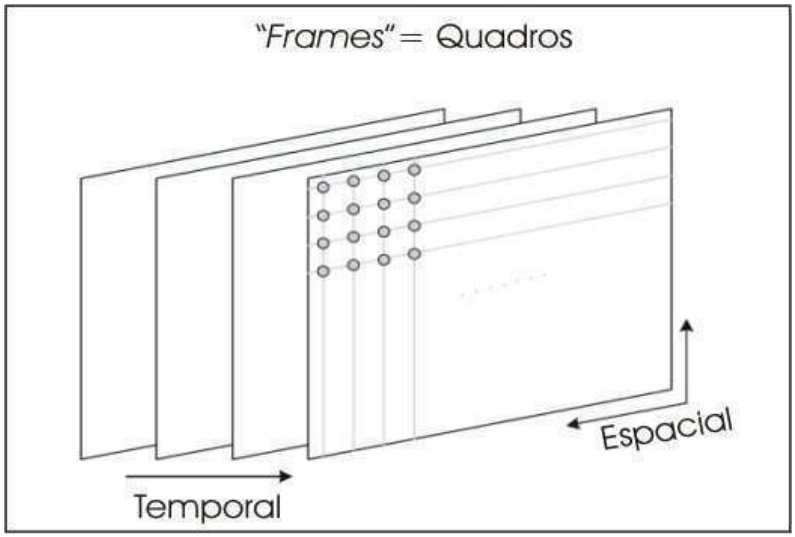

Figura 2.2: Redundâncias de informações temporal e espacial.

2.4.1.1 Redundância espacial

Redundância espacial é o excesso de informações semelhantes em determinadas regiões da imagem. Na Figura 2.3 está contido um exemplo de um único quadro do vídeo. Dentro das regiões destacadas, existe uma pequena variação no conteúdo da imagem, e conseqüentemente, há redundância de espacial significante.

Diante desta característica, são utilizadas técnicas para eliminar a redundância de informações. Algumas técnicas utilizam uma pequena quantidade de informação da imagem para predizer as informações necessárias para recompor o restante da imagem que está localizada nas proximidades da informação que foi utilizada como referência.

Normalmente, para melhorar a eficiência na predição das informações de uma região da imagem, é necessário que a imagem seja submetida ao processo de quantização, que irá suprimir os coeficientes com valores próximos a zero, provocando a perda dos dados responsáveis pelas informações referentes aos detalhes finos da imagem, resultando assim em uma suavização da mesma.

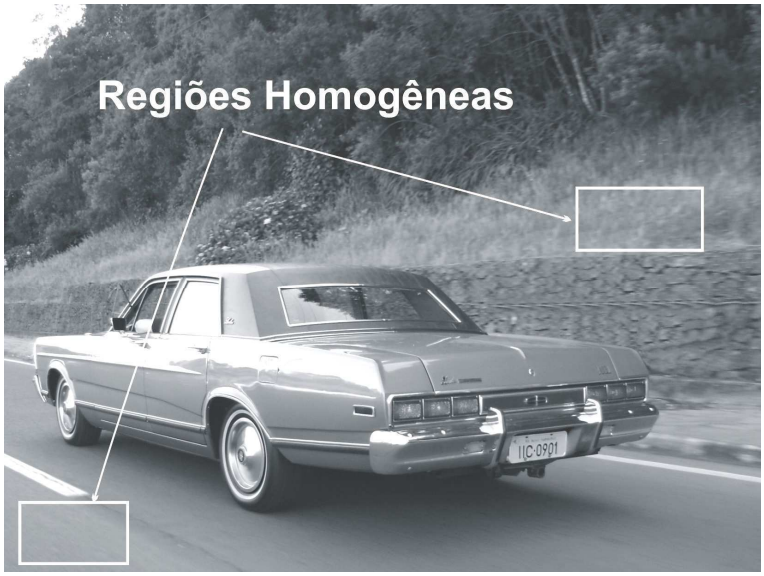

Figura 2.3: Exemplos de regiões homogêneas.

Na Figura 2.4 está contido o mesmo quadro após ter sido submetido a um processo de “suavização da textura” da região que está em segundo plano na imagem. Mesmo após o processo, a imagem ainda é reconhecível, apesar do fato de ter sido retirado uma quantidade considerável de “informações”.

Figura 2.4: Imagem após processo de suavização da textura.

Para realizar o processo de quantização, a imagem passa por uma DCT (*Discrete Cosine Transform*) que está presente em vários codificadores, ela tem a função de converter os blocos 8x8 do domínio espacial para domínio da freqüência. Ela produz coeficientes DC e AC (o coeficiente DC é o coeficiente que tem freqüência zero em ambas as dimensões, os coeficientes AC são coeficientes para os quais a freqüência em uma ou ambas as dimensões não é zero). Então, cada coeficiente pode ser considerado como uma freqüência bi-dimensional.

A DCT tem a característica de concentrar a energia no coeficiente DC e nos coeficientes AC mais próximos do coeficiente DC levando a produção de coeficientes AC de médias e altas freqüências com pouca energia [11]. Na Figura 2.5, pode-se ver a representação da concentração de energia após a DCT.

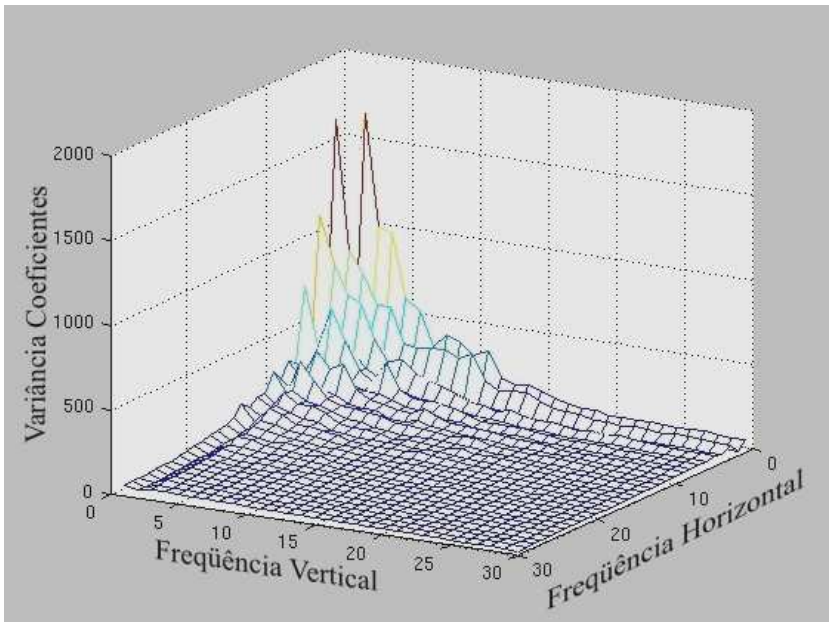

Figura 2.5: Concentração de energia após DCT.

2.4.1.2 Redundância Temporal

Redundância Temporal significa a redundância de informações que existe entre imagens consecutivas. Nas Figuras 2.5 e 2.6 estão contidos quadros que foram extraídos do trailer do filme *Star Wars*. O trailer continha 30 quadros por segundo. Nesta freqüência de amostragem de quadros, 1/30 segundos, ocorrem pequenas

mudanças entre quadros consecutivos do vídeo. Boa parte da imagem fica inalterada entre os quadros consecutivos.

Figura 2.5: Quadro n do filme *Star Wars*.

Figura 2.6: Quadro n+1 do filme *Star Wars*.

Diante desta característica, ao invés de codificar toda a imagem, só é necessário codificar as informações que não são redundantes, utilizando a imagem anterior como referência.

A técnica mais utilizada é a estimação do movimento, que considera o movimento por blocos, ou seja, todos os *pixels* dentro de um bloco obedecem ao mesmo tipo de movimento. Um modelo simples de movimento consiste em considerar somente a translação.

O processo de EM (Estimação de Movimento) de um macrobloco 16x16 *pixels* é codificado com base na utilização de outro macrobloco igual ou semelhante que reside em um quadro de referência que normalmente é o anterior.

Durante a busca de um bloco de *pixels* referenciado no quadro anterior, é selecionado o bloco que está mais próximo e semelhante ao que está sendo codificado, gerando assim, pouca diferença entre as coordenadas dos vetores de movimento, como representado na Figura 2.7.

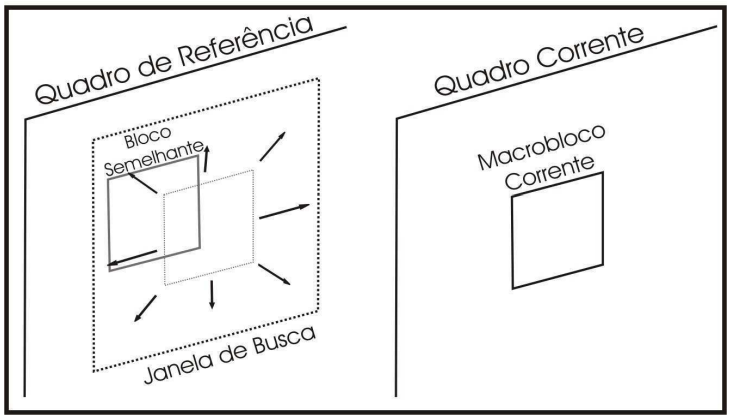

Figura 2.7: Busca do bloco semelhante no quadro de referência.

Dependendo do tipo do macrobloco, pode-se obter até quatro vetores de movimento codificados. Quando o macrobloco semelhante é detectado no quadro de referência, um ou mais vetores são usados para descrever o posicionamento do macrobloco atual em relação ao referido bloco. Na Figura 2.8 pode-se ver a geração do vetor.

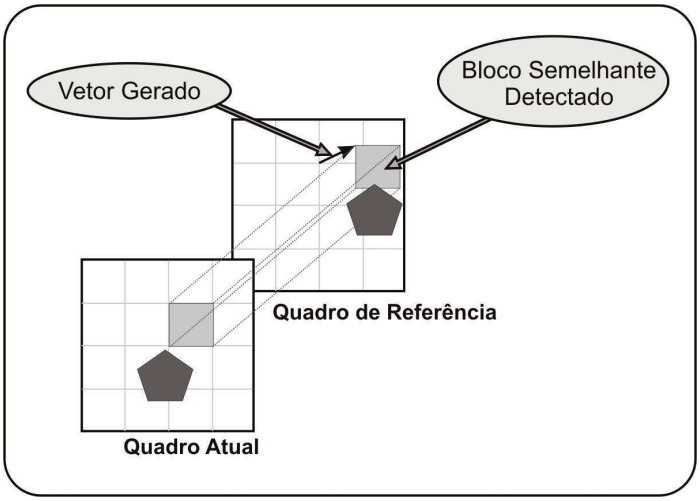

Figura 2.8: Geração do vetor de movimento.

2.4.2 Descompressão de vídeos

O processo de descompressão ou “decodificação” do arquivo de vídeo comprimido é realizado da seguinte forma: o quadro corrente é gerado com base na reutilização de dados já decodificados existentes em uma cena anterior ou posterior. Alguns macroblocos 16x16 *pixels* sofrem um processo de translação com base nos vetores de movimento e/ou substituição por blocos derivados do módulo de decodificação de textura. Após esta etapa, são adicionadas à imagem pequenas diferenças contidas entre as cenas corrente e de referência. Na Figura 2.9 estão os componentes necessários para decodificação do quadro.

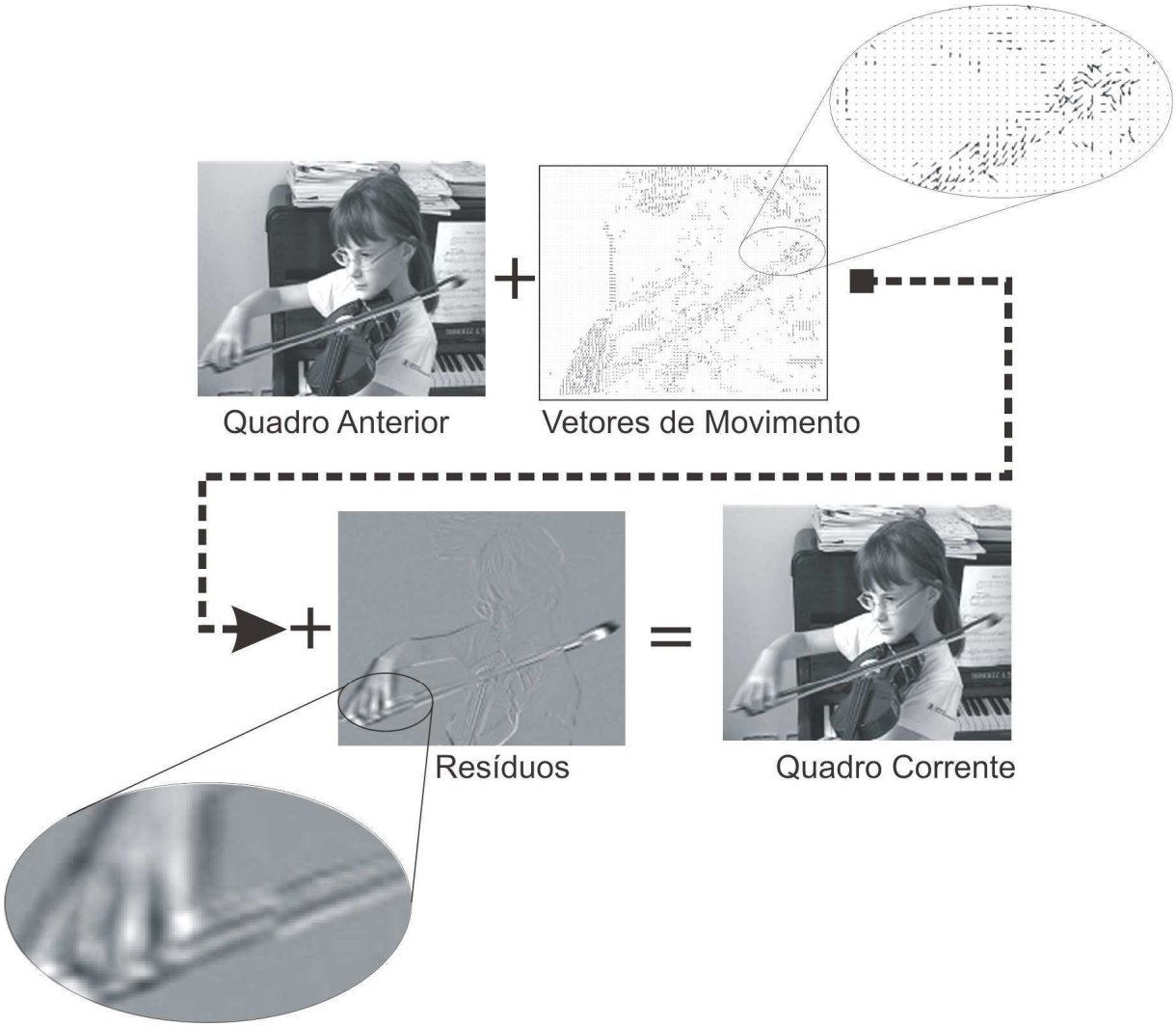

Figura 2.9: Decodificação do quadro.

Pode-se dividir um decodificador de vídeo em quatro módulos, que podem ser visualizados na Figura 2.10.

Figura 2.10: Esquemático de decodificador de vídeo.

Os módulos podem ser descritos da seguinte forma:

- ***Demultiplexador Bitstream*** é responsável por decodificar o cabeçalho de vídeo, extraindo as informações necessárias para configurar os demais módulos durante o processo de decodificação do vídeo, extrair os valores VLC (*Variable Length Codes)* e decodificá-las.
- ***Compensação de Movimento*** realiza o processo de geração de uma imagem utilizando vetores de movimento. Estes predizem o sentido da translação que os *pixels* realizam na imagem, usando como referência um quadro anterior ou posterior e adicionando à imagem gerada, os resíduos da imagem original, recebidos junto com os vetores para minimizar as distorções.
- ***Decodificador da Textura*** realiza o processo de decodificação da textura da imagem.
- ***Composição da Imagem*** combinar as imagens vindas da decodificação dos vetores de movimento e da decodificação da textura.

2.5 Considerações Gerais

Neste capítulo foram apresentados as informações necessárias ao entendimento do restante do trabalho, abordando os conceitos sobre verificação funcional, metodologia VeriSC e o detalhamento dos componentes necessários para sua

aplicação (ambiente de simulação "*testbench*" e cobertura que é usada como métrica).

O paradigma da codificação e decodificação do vídeo digital, que é baseado na eliminação de redundância de informações nos aspectos espacial e temporal, torna viável a compactação eficiente dos dados ao custo de uma pequena perda da informação visualmente relevante. Esse fato resulta em uma considerável aplicação do conceito de compressão de vídeo digital pelas indústrias de dispositivos eletrônicos.

No próximo capítulo serão descritos o problema existente na etapa da verificação funcional de um decodificador de vídeo MPEG-4 e as causas do problema.

CAPÍTULO 3 - Caracterização do Problema

Neste capítulo será apresentado o funcionamento de um circuito decodificador de vídeo MPEG-4, para auxiliar na contextualização do problema existente na verificação funcional quando é aplicado aos decodificadores de vídeos. Diante da caracterização do problema, serão demonstradas as estratégias utilizadas nos trabalhos relacionados e uma proposta de solução.

3.1 Circuito Decodificador de Vídeo

Para contextualizar o funcionamento de um circuito decodificador de vídeo MPEG-4, foi utilizado como exemplo o decodificador implementado na UFCG [2]. O diagrama esquemático do circuito pode ser visto na Figura 3.1.

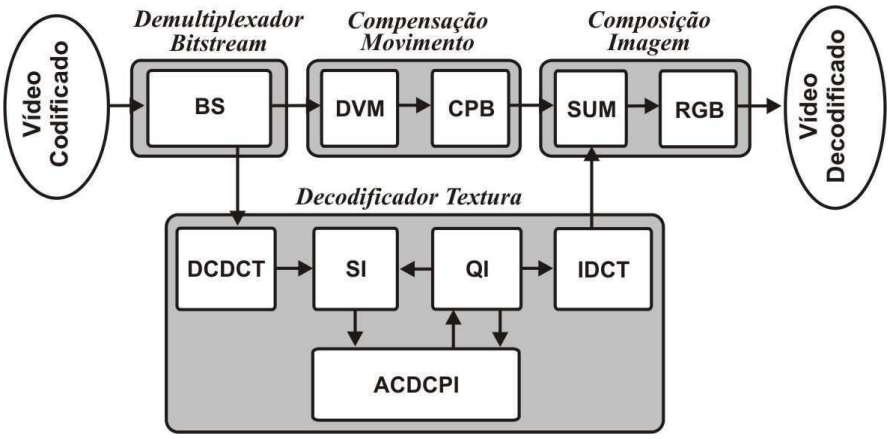

Figura 3.1: Esquemático do decodificador de vídeo MPEG-4.

O circuito decodificador está dividido em 10 módulos, sendo agrupados em 4 grupos funcionais, abrangendo a demultiplexação do bitstream, compensação de movimento, decodificação da textura e composição da imagem.

A seguir, é apresentada a descrição funcional de cada módulo:

- **BS (*Bitstream*)**

O BS é um demultiplexador de cabeçalho MPEG-4 e decodificador de VLC. Ele é responsável por receber o fluxo de vídeo compactado no formato do MPEG-4 e alimentar os demais blocos com dados e/ou parâmetros de configuração. Na Figura 3.2 é representado o diagrama esquemático do BS.

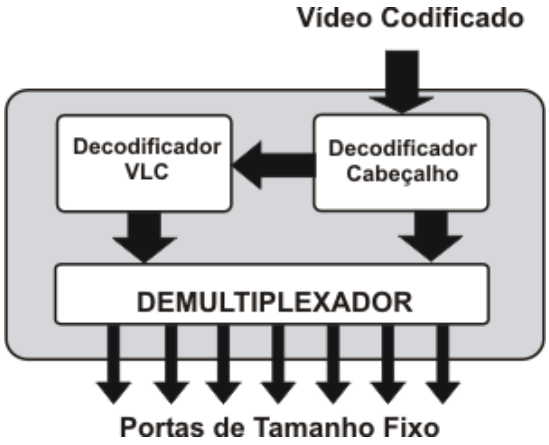

Figura 3.2: Esquemático *Bitstream*.

O módulo possui 7 portas de saída (2 com dados referentes à imagem e 5 com dados referentes à configuração).

- **DVM (Decodificação de Vetor de Movimento)**

Sua função é realizar a decodificação dos vetores de movimento por meio dos cálculos de diferenças entre vetores. Este módulo recebe dados provenientes do BS, após decodificá-los, encaminha os vetores para módulo CBP.

- **CBP (Copiador de Blocos Predição)**

Sua função é copiar do quadro de referência os valores de luminância e crominância para compor a próxima imagem de cada macrobloco. O vetor de movimento de cada bloco de luminância informa onde se encontram seus valores. Os valores de crominância devem ser calculados a partir de média dos vetores de luminância.

- **DCDCT (Decodificação de Coeficientes DCT)**

Sua função é recompor as carreiras de zeros contidas no arranjo de coeficientes do macrobloco, que foram suprimidas na codificação. Este módulo recebe informações do BS no formato de uma tripla (magnitude, quantidades de zeros que antecede a magnitude, se é o último coeficiente do

macrobloco). O produto resultante deste módulo é um arranjo unidimensional de valores que é repassado para módulo SI.

- **SI (*Scan* Inverso)**

Este módulo tem a função de transformar um arranjo unidimensional em uma matriz bidimensional. Ele recebe o arranjo com os coeficientes na ordem que foram codificados, construindo as matrizes em função do tipo de predição, enviando-as posteriormente para módulo ACDCPI.

- **ACDCPI (AC e DC Predição para Intra)**

Este módulo executa a predição inversa nos macroblocos do quadro Intra, em que os coeficientes do macrobloco corrente são calculados com base nas informações decodificadas anteriormente.

- **QI (Quantização Inversa)**

O módulo QI realiza a quantização inversa dos coeficientes. Este é proporcionado pela multiplicação por um fator de escala. Os coeficientes desquantizados são enviados para a IDCT.

- **IDCT (Inversa da DCT)**

A IDCT realiza a transformada inversa da DCT, ou seja, transforma os valores dos coeficientes da matriz, que estão no domínio da freqüência, em uma matriz de *pixels*, dentro do domínio espacial. A matriz de *pixels* é encaminhada para módulo SUM.

- **SUM (Somador)**

O SUM realiza a soma dos valores de luminância e valores de crominância dos *pixels* que chegam da compensação de movimento e da decodificação da textura. Os resultados da soma das luminâncias são armazenados na memória para ser utilizado na composição do quadro da imagem corrente e servir posteriormente como quadro de referência. O resultado das somas das crominâncias é encaminhado ao módulo RGB.

- **RGB (Conversor YCbCr para RGB)**

A função deste módulo é converter os valores Y, Cb e Cr de cada *pixel* em valores R, G e B.

3.2 Contextualização do Problema

Diante da baixa cobertura do BS apresentado na seção inicial deste trabalho, foram utilizados vídeos naturais como estímulos no processo de verificação na tentativa de alcançar a cobertura especificada. Foram aplicados três vídeos naturais, cada um com 600 quadros. A cobertura no módulo BS foi verificada nas portas de saída, pois era necessário conferir se o BS demultiplexava os dados corretamente. Na Tabela 3.1 pode-se verificar as taxas de cobertura do módulo. A tabela está representando o tipo de estímulo adotado e a porcentagem da cobertura atingida pelo módulo BS.

Tabela 3.1: Cobertura do módulo BS do MPEG-4.

Tipo de Estímulo	Cobertura no Módulo BS (%)
Vídeo Sintético Aleatório	67
Vídeo Natural "*Deep"*	86
Vídeo Natural "*Drift"*	84
Vídeo Natural "*Beer*"	86
Cobertura Acumulada	88

As simulações da verificação funcional mostraram que a cobertura atingida no BS não era satisfatória, mesmo utilizando os vídeos sintéticos e naturais em conjunto. Na Tabela 3.2 são apresentadas as medidas de cobertura extraídas individualmente nas portas de saída do módulo BS.

Tabela 3.2: Cobertura das portas de saída de BS.

Porta de saída	Vídeo Sintético Aleatório (%)	Vídeos Naturais (%)
DVM	33	65
DVM_VOP	70	100
CBP	100	100
DCDCT	35	37
SI	100	100
ACDCPI	81	100
QI	49	100

Diante dos dados apresentados sobre a cobertura, fica evidente a existência de um problema de cobertura no circuito decodificador de vídeo. Com a descrição funcional dos módulos do decodificador MPEG-4 apresentados na seção anterior, fica visível que o tipo de estímulo necessário para a verificação funcional do módulo BS é o mesmo estímulo utilizado para a verificação do decodificador no nível de topo.

Como a verificação do módulo BS ficou comprometida, ocorreu um comprometimento na verificação funcional em termos do sistema por completo, no nível de topo.

Partindo da hipótese de que esse problema é devido ao uso de estímulos inadequados, para chegar numa abordagem capaz de gerar estímulos melhores, foi realizada uma análise mais detalhada sobre a função e implementação do bloco BS.

3.3 Principais Causas do Problema

Com a análise detalhada dos dados obtidos das simulações, foram descobertas as razões da baixa taxa de cobertura. As portas de saída do BS para os módulos DCDCT e DVM apresentaram uma baixa taxa de cobertura devido às características de codificação dos vídeos citadas no capítulo anterior. Em seguida, foi detectada a ausência de coeficientes AC de média e alta freqüência e a baixa variância dos coeficientes de baixa freqüência. Outro problema descoberto foi a baixa variância dos vetores de movimento.

O problema com o módulo DVM é ocasionado devido à estratégia utilizada pelo módulo EM (Estimação de Movimento) do codificador. Que durante a busca de um bloco de *pixels* referenciado no quadro anterior, seleciona o bloco que está mais próximo e semelhante ao que está sendo codificado, gerando assim, pouca diferença entre as coordenadas dos vetores de movimento, como representado na Figura 2.7.

Com relação à DCDCT, as características do processo de quantização adotadas pelos codificadores de vídeos (dependendo dos parâmetros de quantização) podem causar perdas significativas de coeficientes de média e alta freqüência. Na Figura 3.3 é representado um bloco 8x8 no domínio da freqüência passando pelo processo de quantização, os coeficientes que estão na área de cor

cinza são os coeficientes AC de média e alta freqüência, pode-se ver que todos os coeficientes perderam informações após o processo de quantização. A freqüência dos coeficientes AC dentro da matriz aumentar na direção da esquerda para direita e de cima para baixo.

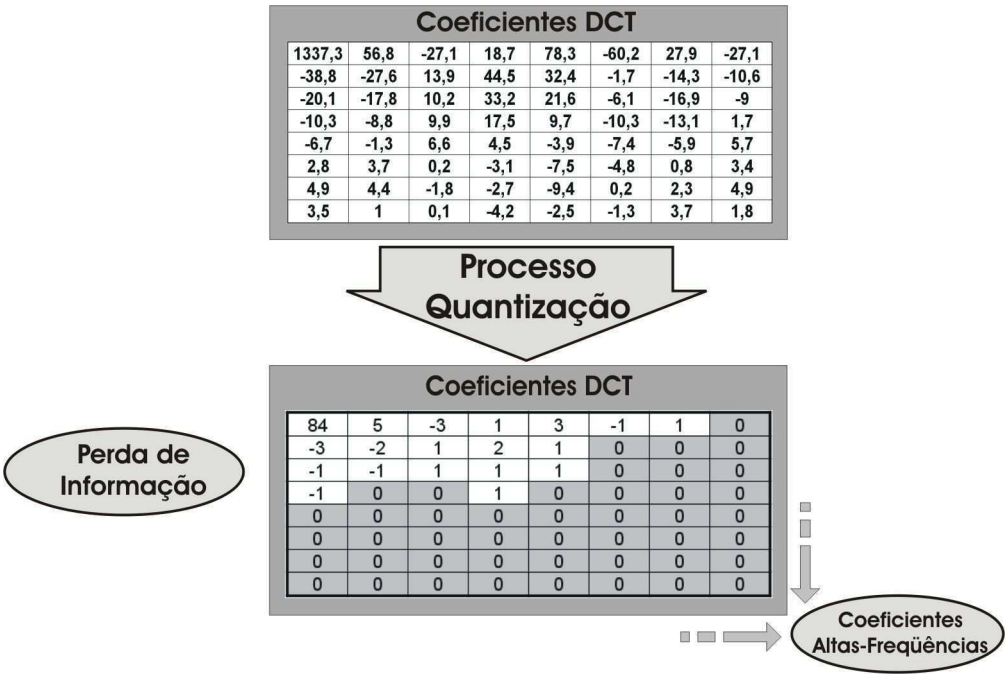

Figura 3.3: Processo de quantização.

3.4 Trabalhos Relacionados

Devido à complexidade do circuito de um decodificador de vídeo, os engenheiros executam o processo de verificação funcional em duas fases: na primeira fase realizam a verificação individual de cada módulo funcional do *IP-Core* e na segunda fase realizam a verificação no nível do topo.

Na primeira fase, a geração dos estímulos torna-se menos complexa devido ao isolamento das funcionalidades de cada módulo e da especificação do projeto.

Na segunda fase, o tipo de estímulo adequado é um *bitstream* (seqüência de bits contendo as informações de um vídeo codificado). O processo de geração do *bitstream* possui uma complexidade considerável devido ao alto relacionamento e acoplamento entre as variáveis de configuração e de dados do arquivo de vídeo comprimido.

Os trabalhos relacionados nesta dissertação não apresentam informações quantitativas e qualitativas para serem comparadas com os resultados obtidos do RandMovie. Eles relatam apenas a forma como foram executadas as verificações funcionais dos *IP-cores.*

Diante das dificuldades citadas, Parash [16] relata um estudo de caso sobre um decodificador de vídeo, que passou pelo processo de verificação formal utilizando a ferramenta RuleBase, desenvolvida pela IBM *Haifa Research Laboratory* [17]. A equipe executou a verificação no nível de módulos e usou o RuleBase para criar ambientes de testes utilizando a especificação do projeto como base.

Segundo Seong-Min Kim, et al. [13] foi realizada uma verificação funcional no nível de módulos individuais e *top-level.* O processo foi aplicado sobre um CODEC (codificador e decodificador) de vídeo, além de utilizar modelos de memórias externas descritas em HDL (*Hardware Description Language*), os quais armazenavam o código, executado pelo processador ARM (processador para sistemas embarcados), uma seqüência de bits com dados de configuração dos módulos e vetores de teste gerados pelo modelo de referência. Ao final do processo, os dados eram extraídos e comparados com os dados provenientes do modelo de referência.

Seong Mo Park, et al. [18] descreve no seu trabalho a verificação funcional realizada sobre um IP-Core do decodificador H.264. Juntamente com sua equipe, implementou um modelo de referência na linguagem C, utilizando modelos de memórias externas SDRAM, adotando um conjunto de vídeos como estímulos, totalizando 600 quadros de testes. Comparando, assim, os resultados obtidos do modelo de referência com os resultados provenientes dos modelos RTL (*Register Transfer Level*) e FPGA (*Field-Programmable Gate Array*).

Normalmente, é adotado um vídeo natural que é codificado com os mesmos parâmetros da especificação do *IP-Core*. Vários engenheiros utilizam estes vídeos em seus processos de verificação, no trabalho de Miyashita [12] foi produzido um vídeo utilizando o software VirtualDub [20] o qual foi codificado com Xvid [9]. O vídeo resultante foi injetado no *testbench* e os resultados da decodificação pelo IP-Core foram comparados com os da decodificação realizada pelo Xvid do mesmo arquivo de vídeo.

3.5 Considerações Gerais

Neste capítulo, foram descritos as funcionalidades dos módulos de um decodificador MPEG-4 e os problemas ocorridos durante a sua verificação funcional. As principais causas deste problema foram destacados após uma análise detalhada do funcionamento do decodificador e dos dados coletados durante as simulações.

O processo de quantização dos coeficientes durante a codificação do vídeo provoca perdas de informação e ao final do processo os coeficientes de média e alta freqüência apresentam valores irrelevantes, próximos a zero. A EM produz vetores de movimento que apresentam variância reduzida.

O vídeo digital, utilizado para ser codificado e aplicado como estímulo no processo de verificação do decodificador, deve possuir características que durante o processo de codificação, os coeficientes de média e alta freqüência não sejam suprimidos.

No próximo capítulo, serão apresentadas as estratégias adotadas para amenizar os efeitos dos resultados produzidos pelos módulos DCT e EM do codificador.

CAPÍTULO 4 - Descrição do “RandMovie”

Neste capítulo, estão descritos a metodologia e estratégias adotadas durante o desenvolvimento da solução para geração de vídeos aleatórios, adequados para serem utilizados como estímulos no processo de verificação, visando à melhoria da cobertura nas portas DCDCT e DVM do BS.

4.1 Arquitetura do RandMovie

A arquitetura do RandMovie foi projetada para ser simples, flexível e reutilizável. Deste modo, poderá ser usado no *testbench* do circuito decodificador MPEG-4. Foi desenvolvido um gerador de quadros aleatórios que produz uma seqüência de quadros com características peculiares e que são submetidos a um codificador para realizar a compressão do vídeo no formato desejado.

O RandMovie pode ser dividido em 3 partes:

- **Modelos de quadros**, que tem a função de fornecer as características que o quadro deverá conter;
- **Seletor Aleatório**, selecionará o modelo de quadro que será produzido, obedecendo a probabilidade de 60% e 40% para modelos 1 e 2, respectivamente, que serão explicados mais adiante;
- **Construtor de quadros**, produzirá o quadro *pixel* à *pixel* seguindo as regras recebidas pelo seletor aleatório. Os quadros gerados estarão no espaço de cor YCbCr e com tamanho de 176x144 *pixels*.

Na medida em que os quadros vão sendo criados, eles são entregues em seqüência para um codificador que realizará a compressão dos mesmos. Na Figura 4.1 é apresentada a arquitetura do RandMovie.

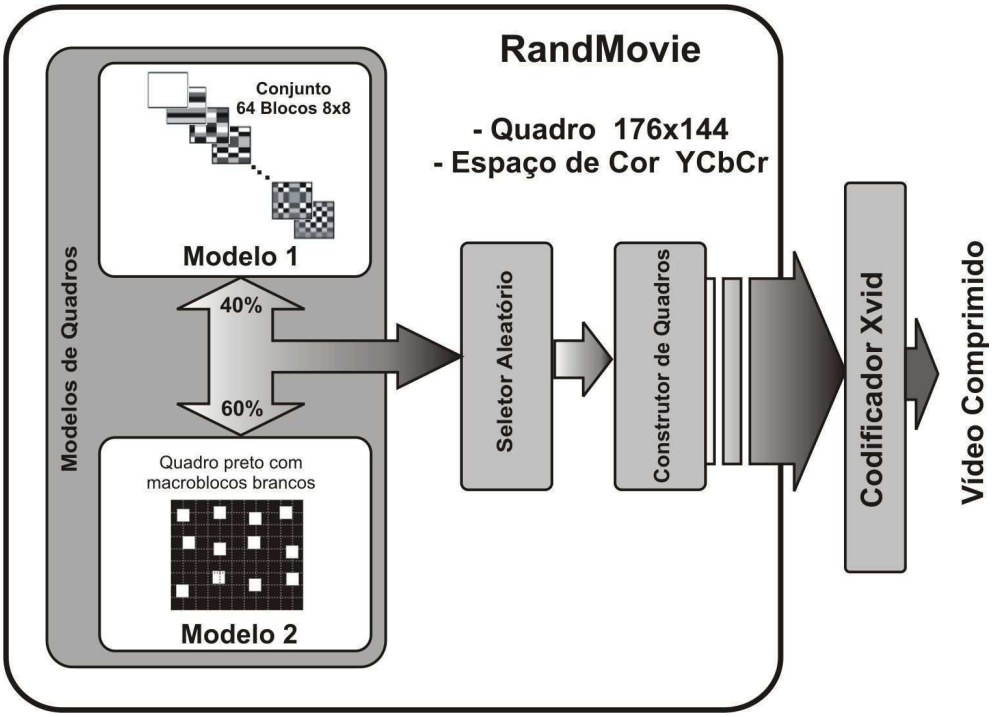

Figura 4.1: Arquitetura do RandMovie.

4.2 Estratégia para Porta DCDCT do BS

Devido à DCT provocar uma concentração de energia nos coeficientes DC e AC de baixa freqüência, os quadros que são submetidos a ela, devem possuir características que forcem a distribuição de energia, entre os coeficientes DC e AC equivalentemente. Isso faz com que os coeficientes AC de média e alta freqüência possam ter uma quantidade suficiente de energia que durante o processo de quantização não sejam suprimidos.

De acordo com Rao, et al. [11], qualquer bloco 8x8 de *pixels* pode ser representado como uma soma ponderada de 64 padrões de base conforme representado na Figura 4.2. A saída da DCT é um conjunto de pesos para estas bases (coeficientes da DCT). Multiplicando-se cada padrão base por seu peso e somando todos, obtêm-se como resultado a imagem original.

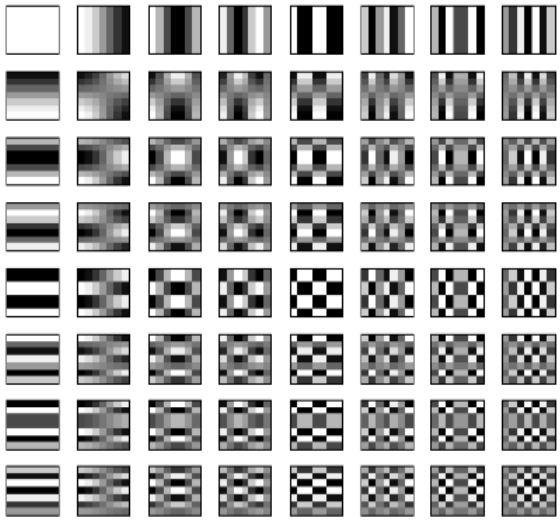

Figura 4.2: Padrões de Base, blocos 8x8.

Com base nestes padrões, podem-se gerar quadros aleatórios em que cada bloco de *pixels* 8x8 do quadro, possuirá algum dos 64 blocos do padrão de base. Deste modo, a DCT gerará os coeficientes DC e AC com valores que depois do processo de quantização, ainda será possível obter os coeficientes AC de média e alta freqüência com valores significativos. Este tipo de quadro será denominado modelo 1.

Na Figura 4.3 é apresentado um quadro extraído de um vídeo gerado pelo RandMovie. Pode-se observar no zoom aplicado a uma parte do quadro, os blocos com os padrões de base.

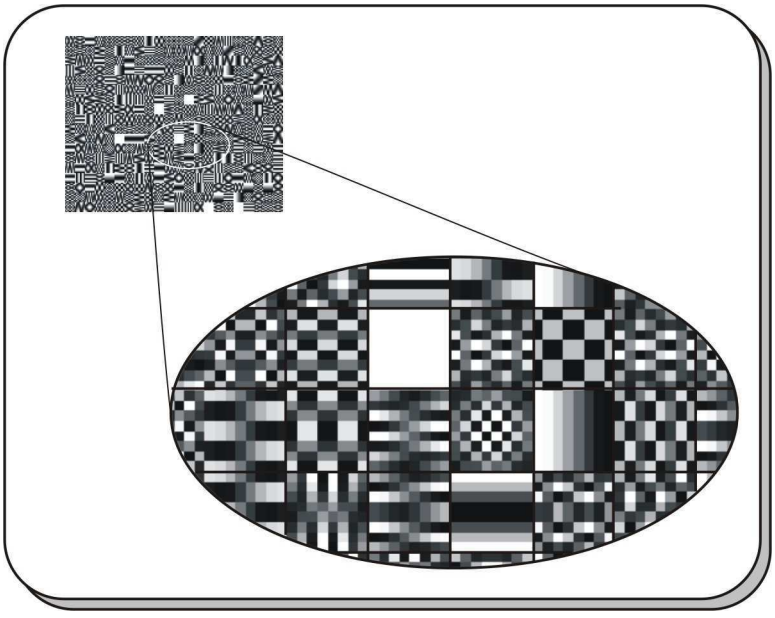

Figura 4.3: Quadro gerado.

Adotando esta estratégia, é possível obter blocos de coeficientes AC com uma variância de valores superior ao obtido dos vídeos naturais e do gerador VSA. Com relação ao coeficiente DC, este atingirá seu valor máximo, pois um bloco totalmente branco após a DCT irá conter toda a energia concentrada no coeficiente DC.

4.2 Estratégia para Porta DVM

Devido às características do processo de estimação de movimento adotado pelo codificador, que consiste em utilizar informações contidas em quadro anteriores para predizer o movimento, deve-se reduzir a quantidade de imagens semelhantes dentro da janela de busca para forçar os algoritmos de busca a percorrer toda área da janela.

Dessa forma, devem-se criar quadros com cores extremas para maximizar a diferença entre os macroblocos 16x16 *pixels* e que dentro dos limites da janela de busca só exista um único macrobloco semelhante, como mostrado na Figura 4.4. Este tipo de quadro será denominado modelo 2.

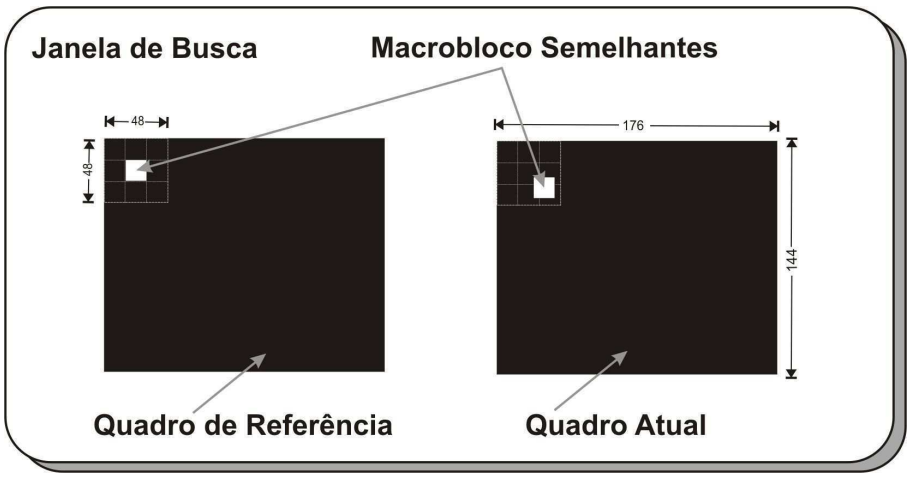

Figura 4.4: Janela de busca e macroblocos semelhantes.

A janela de busca tem área nove vezes maior do que o macrobloco [1], este possui 16x16 *pixels* o que leva a uma janela de busca na ordem de 48x48 *pixels*. Na Figura 4.5 é apresentado um macrobloco dentro dos limites de uma janela de busca.

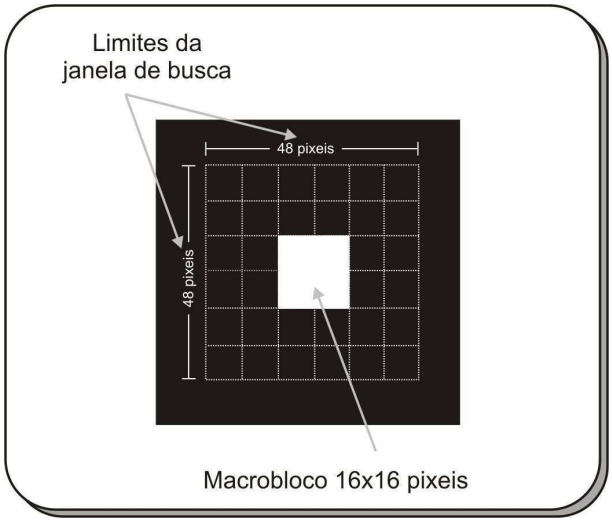

Figura 4.5: Janela de busca.

Cada quadro é subdividido em quadrantes de 48x48 *pixels*, onde será inserido apenas um único macrobloco branco por quadrante. Na Figura 4.6 é apresentado um quadro particionado em quadrantes.

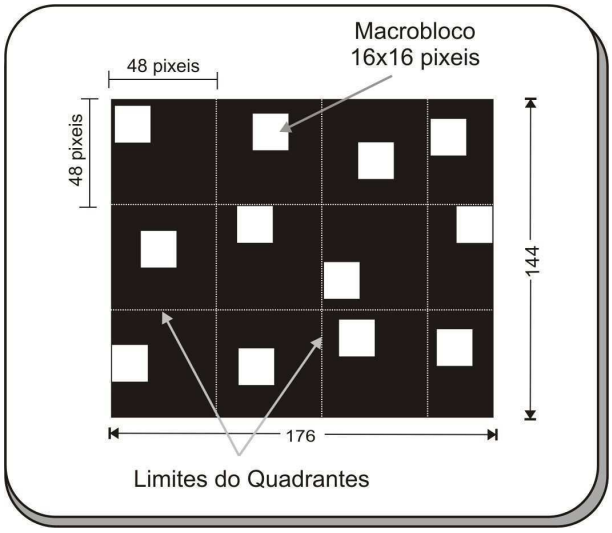

Figura 4.6: Quadro particionado em quadrantes 48x48 *pixels*.

Cada quadrante será subdividido em regiões a Figura 4.7, representa a divisão em regiões do quadrante, onde a área cinza não conterá macroblocos e a área preta será a região útil de 32x32 *pixels* onde será posicionado aleatoriamente o macrobloco dentro dos limites da região. Desta forma serão utilizados 12 macroblocos brancos por quadro de 176x144 *pixels*, limitando assim a quantidade de informações entre os quadros.

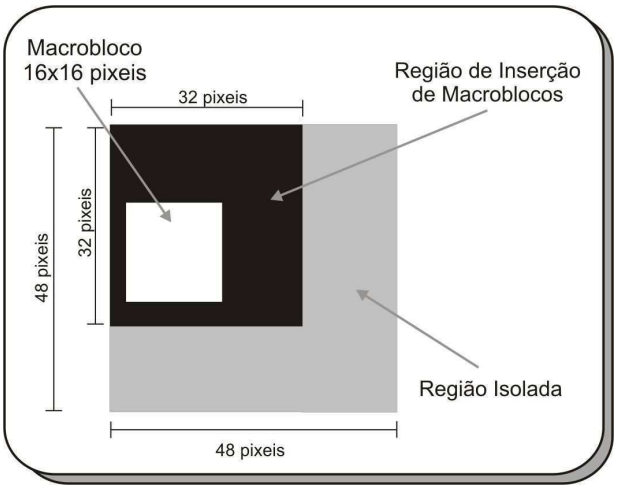

Figura 4.7: Regiões do quadrante.

Na Figura 4.8 é apresentado um quadro com os macroblocos inseridos dentro da região útil e um exemplo dos limites da janela de busca em relação ao mesmo. A região isolada tem a função de evitar que os algoritmos de busca utilizem as informações de um quadrante vizinho para referenciar o quadrante atual, como pode ser visto na Figura 4.9.

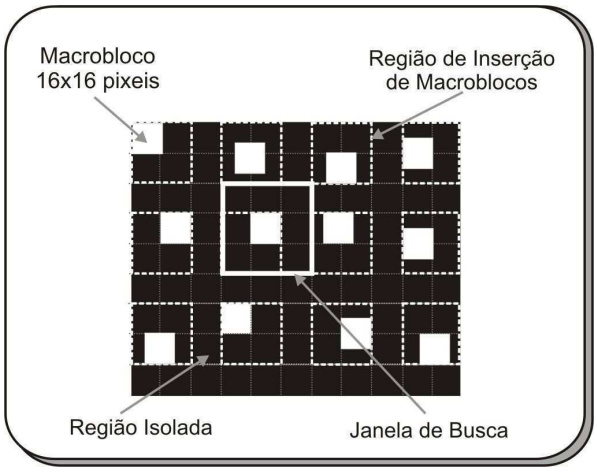

Figura 4.8: Inserção de macroblocos e janela de busca.

Ainda na Figura 4.9, pode ser visto a geração do vetor de movimento. Após detectação da imagem semelhante dentro da janela de busca no quadro anterior é gerado um vetor, que irá descrever a translação do macrobloco entre os quadros.

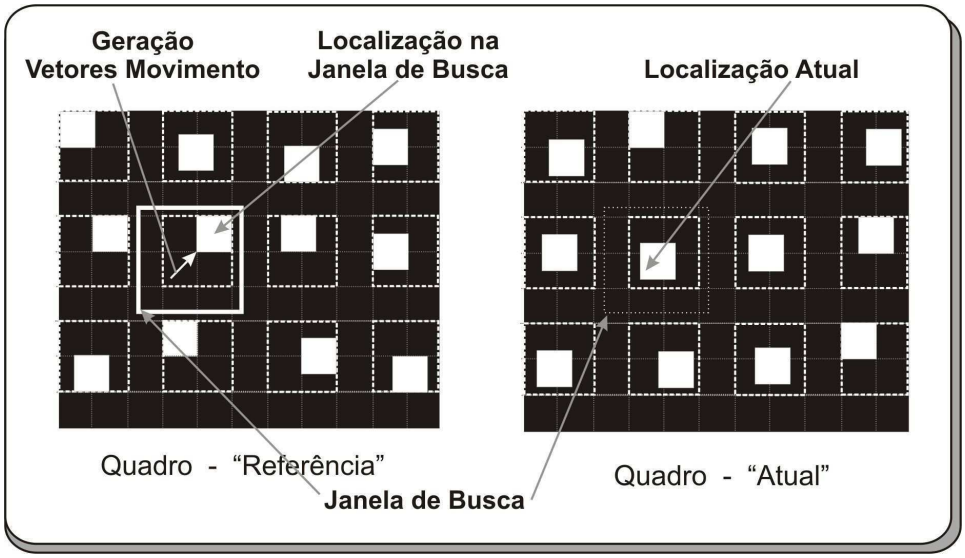

Figura 4.9: Compensação de movimento.

4.2 Combinação das Estratégias

As estratégias adotadas provêm um total de 2 modos de criação dos quadros para produção dos vídeos, o primeiro modo utiliza-se da construção do quadro utilizando os padrões de base e o segundo modo baseia-se na criação de quadros pretos com alguns macroblocos de *pixels* brancos inseridos no quadro em posições aleatórias.

Para garantir que o RandMovie possa gerar os estímulos necessários para excitar a estimação de movimento, é necessário direcionar a aleatoriedade da escolha do modelo que será adotado para geração de um novo quadro, pois deve-se garantir algumas seqüências de quadros pretos com macroblocos de *pixels* branco dentro do vídeo para proporcionar o funcionamento correto da estratégia da subseção 4.2.2 durante o processo de geração dos vetores de movimento. Na Figura 4.10 é apresentado um exemplo de uma seqüência de frames.

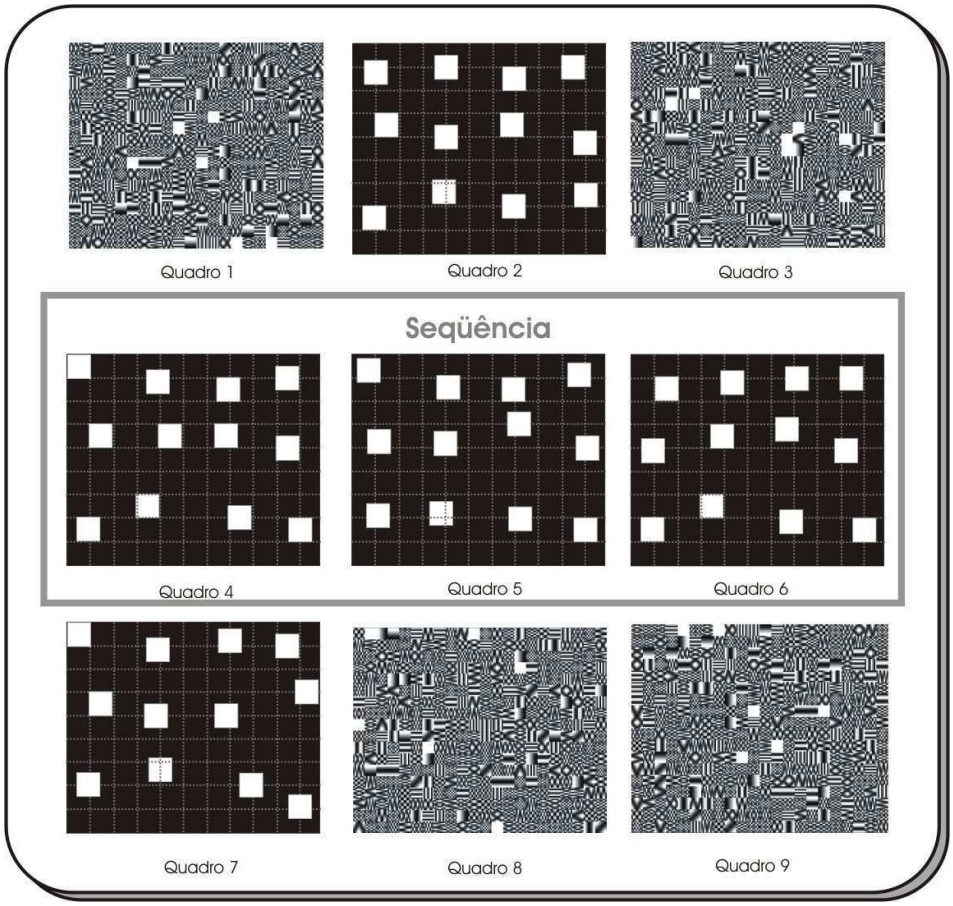

Figura 4.10: Combinação das estratégias.

A obtenção do efeito desejado pelo gerador na produção dos estímulos que atendam à codificação da textura e compensação de movimento é garantida pela redistribuição da probabilidade entre os dois modos de criação dos quadros. Deste modo, o modelo 1 que utiliza os padrões de base como referência possui a probabilidade de ocorrência de 40%, enquanto que o modelo 2 que utiliza quadro preto com macroblocos brancos possui 60% de probabilidade de ser selecionado.

Os valores de 40% e 60% foram obtidos empiricamente, foram realizadas diversas simulações utilizando outras porcentagens, iniciando de 50% e aplicando uma variância de 5% a cada nova simulação, até executar a simulação utilizando as porcentagens de 10% e 90%.

As porcentagens de 40% e 60%, foi escolhido por apresentar o menor tempo para atingir a cobertura máxima fornecida pelo RandMovie.

4.4 Implementação do RandMovie

O RandMovie foi implementado utilizando a linguagem C++ e possui as seguintes características:

- Flexibilidade;
- Programação de alto e baixo nível, devido à necessidade de manipulação de *bits*;
- Compatibilidade com a biblioteca SystemC, que é uma linguagem HDL bastante utilizada para descrever circuitos digitais no nível RTL.

As especificações das variáveis e faixas de valores para geração de um vídeo aleatório com textura e informações de movimento foram retiradas do padrão MPEG-4 [1]. As principais variáveis neste processo vieram dos módulos DCT e DVM, pois estes são os responsáveis por receber os dados provenientes das imagens dos quadros codificados.

As principais variáveis:

- DCT
  - *Level* [-2048, 2047], magnitude do coeficiente;
  - *Run* [0,63], quantidade de zeros antes da magnitude;
  - *Last* [0,1], se é ou não último coeficiente da matriz.
- DVM
  - Vetor Horizontal [-32, 32];
  - Vetor Vertical [-32,32]
  - Resíduo Horizontal [0,32];
  - Resíduo Vertical [0,32];
  - Skip [0,1];

4.5 Considerações Gerais

Neste capítulo, foi apresentada a metodologia aplicada no desenvolvimento do RandMovie abordando a arquitetura do gerador, implementação e as estratégias usadas na solução do problema.

Foram utilizados os padrões de base para amenizar os efeitos do processo de quantização e adotado a redução das informações entre quadros consecutivos para incentivar o módulo EM a produzir vetores que abranjam toda a faixa de valores.

No próximo capítulo, serão apresentados os resultados obtidos com a aplicação do RandMovie no *testbench* do circuito decodificador MPEG-4.

CAPÍTULO 5 - Apresentação e Análise dos Resultados

Neste capítulo, está descrita a utilização do RandMovie em um ambiente real. Será apresentando um estudo de caso consistindo no seu acoplamento ao *testbench* do *IP-core* do MPEG-4 desenvolvido no LAD (Laboratório de Arquiteturas Dedicadas) da UFCG e na realização de uma nova verificação funcional. Os primeiros resultados obtidos foram publicados no 20th *Symposium on Integrated Circuits and Systems Design* (SBCCI'2007) no Rio de Janeiro [15]. O resultado, com as vantagens da aplicação do direcionamento da aleatoriedade no processo de geração de vídeo sintético, foi aceito para ser publicado no 3th *International Conference on Signal Processing and Multimedia Applications* (SIGMAP'2008) na cidade do Porto em Portugal, no mês de Julho de 2008 [23].

5.1 Estudo de Caso: Aplicação do RandMovie ao *Testbench* do MPEG-4

O RandMovie foi adicionado junto com o codificador Xvid ao módulo *Source* do *testbench* do decodificador de vídeo MPEG-4. As modificações do *Source* consistiram na adição do núcleo do codificador Xvid e RandMovie.

Os parâmetros de configuração do codificador, que definem as características do formato do vídeo, foram definidos dentro do *Source* com base nas características do *IP-core* do decodificador que se deseja realizar a verificação funcional e são repassados para o codificador sempre no início da simulação.

A seqüência de *bits* que contém os dados referentes aos quadros gerados pelo RandMovie são transmitidos ao Xvid na proporção em que o DUV vai

consumindo os estímulos. Na Figura 5.1 está contida a representação da estrutura do *testbench* junto com DUV, RandMovie e Xvid.

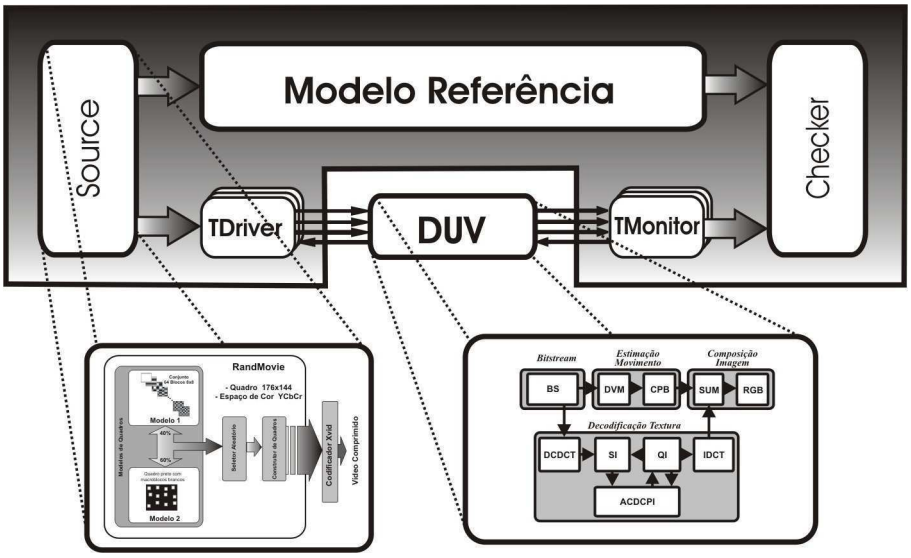

Figura 5.1: RandMovie e decodificador aplicado ao *testbench*.

5.2 Avaliação dos Resultados

Finalizada a etapa de acoplamento, foi executado um conjunto de simulações no módulo BS isoladamente e ao IP-Core completo do MPEG-4, utilizando como métrica a taxa de cobertura. Assim, foi possível obter novos dados que qualificam as características do RandMovie. Nas próximas seções serão apresentados os resultados atuais *versus* anteriores.

5.2.1 Aplicação da Aleatoriedade Direcionada

Na Figura 5.2 está contido um trecho do arquivo de *log* de uma das execuções da simulação do módulo BS utilizando o RandMovie. É possível observar alguns pontos tais como: as seqüências de quadros com as características do modelo 2 (*Frames to Motion Estimation*) e do modelo 1 (*Frames to DCT*); a proporcionalidade entre a geração dos quadros de aproximadamente 40% e 60% entre os modelos 1 e 2, totalizando para esta amostra 21 quadros do modelo 1 e 36 quadros do modelo 2.

```
-bash-3.00$ pwd
/home/silveira/workspace/mpeg/trunk/modules/bitstream/bench
-bash-3.00$ ./tb.sim


         SystemC 2.1.v1 --- Mar 23 2007 20:01:18
      Copyright (c) 1996-2005 by all Contributors
              ALL RIGHTS RESERVED
TB Transaction Recording has started, file = txdb.txt
Note: VCD trace timescale unit is set by user to 1e-12 sec.

Frames to DCT
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to DCT
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to DCT
Frames to DCT
Frames to DCT
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to Motion Estimation
Frames to DCT

-bash-3.00$
```

Figura 5.2: Arquivo de *log* de uma simulação.

5.2.2 Taxas de Cobertura nas Portas do BS

Com base nos resultados obtidos ao final das simulações. O RandMovie proporcionou uma cobertura equivalente ou superior, em todas as portas do módulo BS em relação aos resultados dos VSA e dos vídeos naturais. As variáveis medidas, junto com suas respectivas faixas de valores, e os parâmetros dos *buckets* estão descrit as na Tabela 5.1. Na Figura 5.3 estão contidos os valores obtidos nas portas de saída do módulo BS.

Tabela 5.1: Portas do BS e suas variáveis.

Porta do BS	Variáveis da Porta	Faixa de Valores	BVE_COVER_BUCKET	
			Funcionalidades	Freqüência Repetição
DCDCT	*Level*	[-2048, 2047]	*-2048 <= level < -1024*	100
			-1024 <= level < -128	3.000
			-128 <= level < 0	5.000
			0 <= level < 128	5.000
			128 <= level < 1024	3.000
			1024 <= level <= 2047	100
	Run	[0,63]	*-1 < run < 20*	500
			20 <= run <= 63	500
	Last	[0,1]	*last == 0*	2.000
			last == 1	600
DVM	Vetor Horizontal	[-32, 32]	*0 <= H_Vector <= 32*	1.000
			-32 <= H_Vector <= 0	1.000
	Vetor Vertical	[-32,32]	*0 <= V_Vector <= 32*	1.000
			-32 <= V_Vector <= 0	1.000
	Resíduo Horizontal	[0,32]	*0 <= H_Residual <= 32*	1.000
	Resíduo Vertical	[0,32]	*0 <= V_Residual <= 32*	1.000
	Skip	[0,1]	*skip == 1*	500
			skip == 0	500

CBP	*RND*	[0,1]	*rnd == 0*	500
			rnd == 1	500
	RND_Intra	[0,1]	*intra == 0*	500
			intra == 1	500
DVM_VOP	*Fcode_Forward*	[1,2]	*1 <= forwad <= 2*	100
SI	*AC_Pred_Flag*	[0,1]	*ac_pred_flag == 0*	500
			ac_pred_flag == 1	500
	Intra	[0,1]	*intra == 0*	500
			intra == 1	500
ACDCPI	*AC_Pred_Flag*	[0,1]	*ac_pred_flag == 0*	500
			ac_pred_flag == 1	500
	SVH	[0]	*svh == 0*	100
	Intra	[0,1]	*intra == 0*	500
			intra == 1	500
QI	*Quantiser_Scale*	[0,31]	*0 <= scale < 16*	500
			16 <= scale <= 31	500
	Quant_Type	[0]	*quant_type == 0*	100
	SVH	[0]	*svh == 0*	100
	Intra	[0,1]	*intra == 0*	500
			intra == 1	500

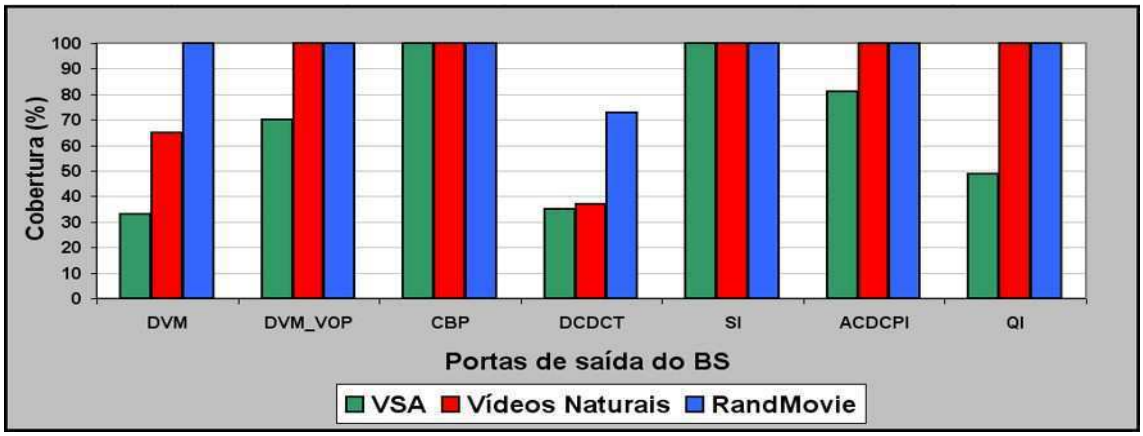

Figura 5.3: Cobertura nas portas de saída do BS utilizando VSA, Vídeos Naturais e o RandMovie.

Em relação ao gráfico da Figura 5.3, existe uma questão a ser apresentada com relação à porta DCDCT, pois nesta porta não foi possível obter a taxa de cobertura máxima especificada. Diante deste fato, foi realizada uma análise minuciosa na implementação do RandMovie e no módulo DCT do codificador Xvid, o que revelou uma deficiência do Xvid, para a versão utilizada v0.9 do codificador. Este codificador realiza uma saturação entre os coeficientes com valores fora da faixa de [-1024, +1023], esta saturação foi detectada após análise dos valores colhidos pelos *buckets* da biblioteca BVE-COVER.

O fato apresentado demonstra uma deficiência com relação à estratégia adotada pelo RandMovie em alocar a responsabilidade de codificar o vídeo para um software desenvolvido por terceiros.

O gráfico da Figura 5.4 expressa a progressão das taxas de cobertura em relação ao tempo. Ao final de 5 horas de simulação, o RandMovie obteve uma diferença de aproximadamente 10% e 30% superior em termos de cobertura aos vídeos naturais e VSA, respectivamente.

A máquina utilizada para executar as simulações, possui a seguinte configuração de hardware: Processador AMD Sempron 2400+, 1GB de memória DDR PC 2700 e HD 80GB ATA 100.

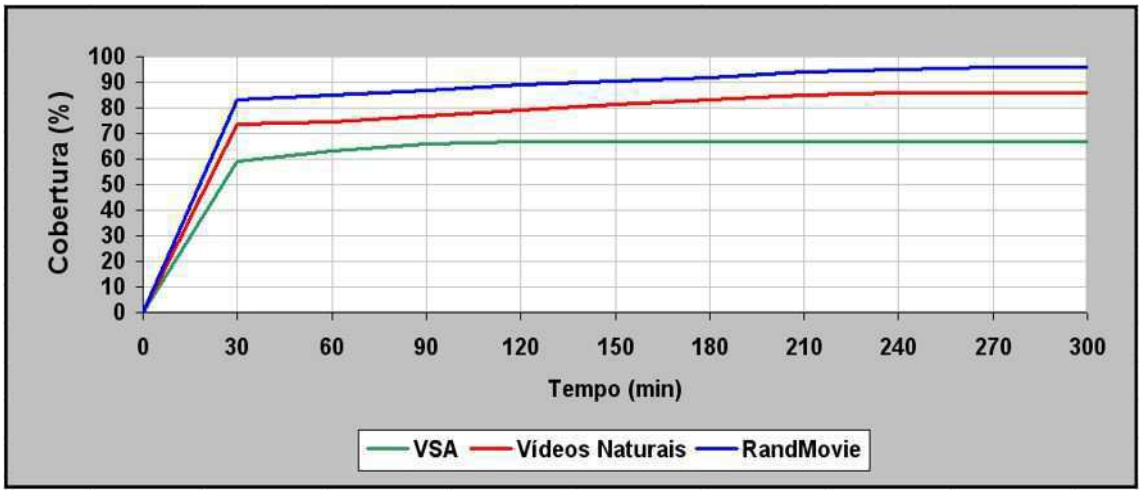

Figura 5.4: Cobertura entre VSA, Vídeos Naturais e o RandMovie.

5.2.3 Custo computacional do RandMovie

Uma consideração deve ser ressaltada com relação ao RandMovie, como gerador, ele se restringe à produção de quadros, assim, existe a necessidade de um codificador de vídeo para realização da sua codificação (compressão) durante a simulação, que necessita de um consumo extra de poder computacional para realização de tal tarefa.

Devido à necessidade de poder computacional para a codificação, na Figura 5.5 pode-se observar que os vídeos naturais entre os 2 (dois) minutos iniciais da simulação obtiveram uma cobertura superior ao RandMovie. Sendo assim, esta dependência de um poder computacional extra é um ponto negativo do RandMovie para simulações em que o tempo é restrito, quando considerados os parâmetros de cobertura *versus* tempo.

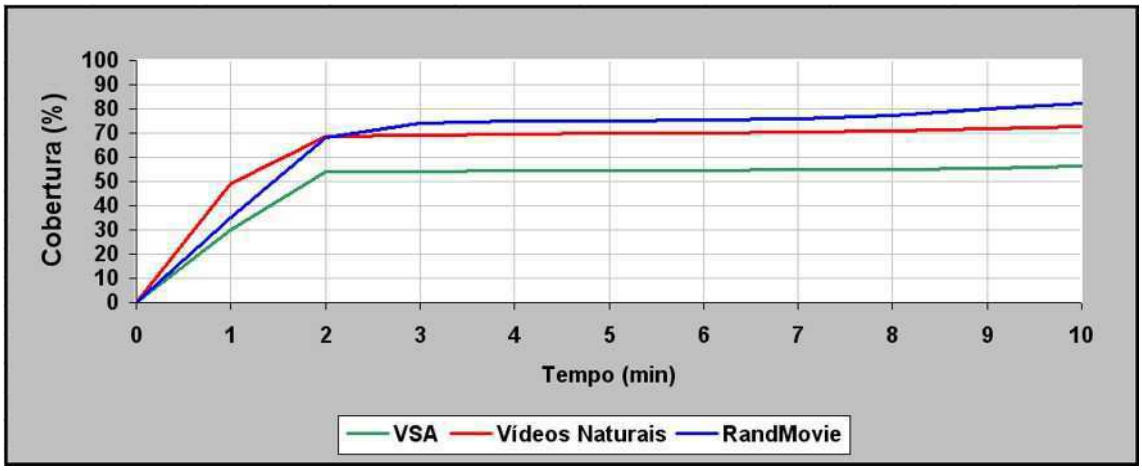

Figura 5.5: Cobertura entre VSA, Vídeos Naturais e o RandMovie.

Para os parâmetros aplicados ao Xvid neste estudo de caso, o consumo computacional gasto para a realização da codificação do vídeo foi de aproximadamente 3% do tempo total gasto durante toda a simulação. Então, mesmo com a necessidade de um consumo computacional extra durante a simulação, o RandMovie possui uma eficiência considerável. Na Figura 5.6 está contido um gráfico com a representação do tempo gasto para realização da verificação funcional *versus* o tempo gasto para codificação do vídeo.

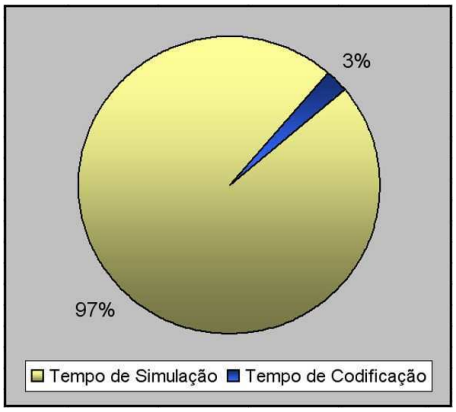

Figura 5.6: Custo computacional para codificação do vídeo.

5.2.4 Exposição de Falha no Circuito

Outro benefício obtido com a cobertura proporcionada pelo RandMovie foi a exposição de uma falha na cobertura realizada em simulações anteriores, descobrindo, assim, um erro na implementação da interface de comunicação entre o BS e o DVM. A falha foi detectada durante uma simulação utilizando o RandMovie, em que o *Checker* acusou uma divergência entre os valores recebidos do modelo de referência e do *TMonitor*.

Este erro poderia ocasionar uma falha durante a operação no IP-Core do MPEG-4 [2] durante a compensação do movimento da imagem. O problema ocorreu em decorrência de uma falha humana durante a integração dos módulos do MPEG-4, pois o engenheiro usou um barramento com 6 bits quando deveria ter utilizado um barramento com 7 bits para cobrir a faixa [-32, +32]. Na Figura 5.7 é apresentado o diagrama em blocos do MPEG-4 com a localização do erro marcado pelo círculo vermelho.

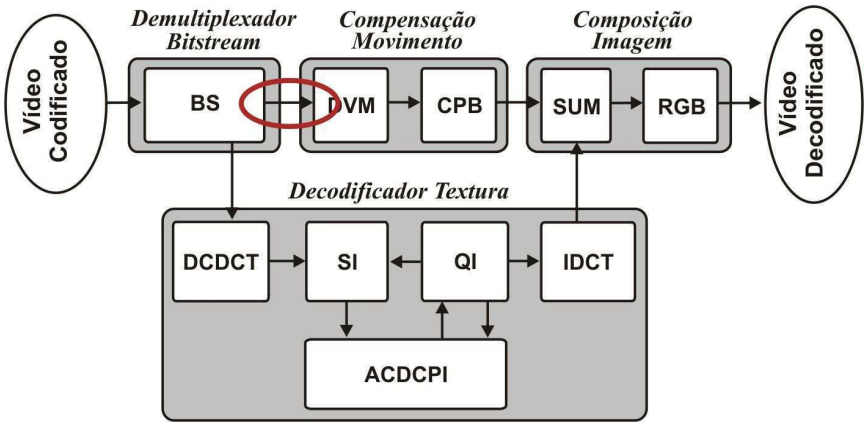

Figura 5.7: Diagrama de blocos do MPEG-4 com indicação do erro.

5.2.5 Considerações Gerais

Com os resultados apresentados, pode-se identificar alguns benefícios proporcionados pelo RandMovie:

- Aplicação da aleatoriedade aos estímulos utilizados durante o processo de verificação funcional dos circuitos decodificadores de vídeo;
- Taxa de cobertura superior em relação a outras fontes de estímulos, permitindo a exposição de falhas no circuito;
- Eficiência do RandMovie, que consome apenas 3% do tempo de processamento durante a simulação.

CAPÍTULO 6 – Considerações Finais

Diante dos resultados apresentados, o RandMovie produz estímulos que contêm características necessárias para serem utilizados no processo de verificação funcional de circuitos digitais. A qualidade dos estímulos gerados foi comprovada no estudo de caso apresentado, que auxiliou no processo de verificação funcional de um decodificador de vídeo MPEG-4.

A arquitetura do RandMovie apresentada neste trabalho possui pontos positivos e questões a serem aprofundadas:

Pontos positivos:

- Aplicação do fator da aleatoriedade no processo de geração de um vídeo sintético, capaz de ser utilizado eficientemente como estímulo na verificação funcional de decodificadores de vídeo MPEG-4;
- Simplicidade na implementação do gerador de vídeo;
- Flexibilidade em utilizá-los nos processos de verificação funcional de outros decodificadores de vídeo MPEG-4 com perfis e níveis diferentes do que foi utilizado no estudo de caso, bastando modificar os parâmetros de codificação do Xvid.

Questões a serem aprofundadas:

- A dependência de um codificador desenvolvido por terceiros para gerar os vídeos;
- Ser susceptível a deficiências do codificador adotado, como foi apresentado pelo Xvid que realiza uma saturação nos coeficientes da DCT que ultrapassem a faixa de valores de [-1024, 1023];
- Custo computacional para geração dos estímulos em diferentes perfis.

Este trabalho proporcionou um aprofundamento do conhecimento nos âmbitos do hardware e software, com a abordagem do conceito sobre metodologia de

verificação funcional, que é parte do fluxo de desenvolvimento dos circuitos digitais. Além de proporcionar o aprofundamento no paradigma que envolve a complexidade sobre os codificadores e decodificadores de vídeos, com a abordagem de suas técnicas e estratégias para obtenção do melhor custo benefício em termos de qualidade da imagem e quantidade de dados para representá-la.

6.1 Sugestões para Trabalhos Futuros

Com o crescimento cada vez maior da utilização de decodificadores de vídeos implementados em hardware, para serem aplicados em dispositivos móveis e da diversidade dos padrões de vídeo existentes na atualidade, existe a necessidade de buscar meios eficientes para amenizar os custos durante o processo de desenvolvimento de circuitos digitais dotados de decodificadores de vídeo.

Diante do exposto, como sugestões para trabalhos futuros, podem ser citadas:

1. Aplicação do RandMovie em processo de verificação funcional de outros decodificadores MPEG-4 que contenham parâmetros de decodificação diferentes do *Simple Profile Level 0*;
2. Realização de novos estudos do fator de impacto utilizando outros codificadores de vídeo MPEG-4 perante o custo computacional necessário para compressão do vídeo;
3. Investigação da viabilidade da correção do módulo responsável pela transformada da DCT do Xvid, viabilizando a geração de coeficientes que contemple toda a faixa de valores [-2048, 2047];
4. Investigação da viabilidade de utilização de cores no processo de geração dos quadros para aumentar a qualidade dos estímulos com relação as variáveis responsáveis pelo resíduo entre as imagens.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] INTERNATIONAL Standard ISO/IEC 14496-2. **Information technology-Coding of audio-visual objects - Part 2: Visual**. 2nd edition. Switzerland: December 2001.

[2] ROCHA, A. K. et al. Silicon validated ip cores designed by the Brazil-IP network. **IP/SOC 2006**. Grenoble: December 2006.

[3] BERGERON, J. **Functional Verification of HDL Models**. 2nd edition. Boston: Kluwer Academic Publisher, 2003.

[4] MONACO, J.; HOLLOWAY, David; RAINA, Rajesh. Functional verification methodology for the PowerPC 604 microprocessor. **DAC '96**. New York: ACM, 1996, p. 319–324.

[5] AHI, A. M. Design verification of the hp 9000 series 700 pa-risc workstations. **Hewlett-Packard Journal**, August, 1992. Disponível em: <http://findarticles.com/p/articles/mi_m0HPJ/is_n4_v43/ai_12475720/pg_1> Acesso em: 27 de novembro 2007.

[6] BRAZIL-IP. Disponível em: <www.brazilip.org.br/fenix> Acesso em: 27 novembro 2007.

[7] SILVA, K. R. G. et al. A methodology aimed at better integration of functional verification and rtl design. **Design Automation for Embedded Systems**. Springer Netherlands. v. 10, n. 4 / December, 2005, p. 285-298.

[8] SILVA, K. R. G. da. An automatic testbench generation tool for a systemC functional verification methodology. **SBCCI '04**. New York: ACM Press, 2004. p. 66–70

[9] TEAM, X. (2003). Xvid API 2.1 reference (for 0.9.x series).

[10] ITU-T Recommendation H.264 (03/2005). **Series H: Audio visual and multimedia systems**: Advanced video coding for generic audiovisual services.

[11] RAO, K. R.; YIP, P. **Discrete cosine transform: algorithms, advantages, applications**. San Diego: Academic Press Professional, Inc., 1990.

[12] MIYASHITA, G. **High-Level Synthesis of a MPEG-4 Decoder Using SystemC**. 2003. Master Thesis - Informatics and Mathematical Modelling, Technical University of Denmark, October, 2003.

[13] KIM, Seong-Min, et al. Hardware-Software Implementation of MPEG-4 Video Codec, **ETRI Journal**, v. 25, n. 6, December, 2003, p.489-502.

[14] BERGERON, J. **Writing Testbenches**: Functional Verification of HDL Models. 2nd edition. Norwell: Kluwer Academic Publishers, 2003.

[15] SILVEIRA, G. S.; SILVA, K. R. da, MELCHER, E. U. Functional verification of an MPEG-4 decoder design using a random constrained movie generator. **SBCCI ’07.** New York: ACM Press, September, 2007. p. 360-364.

[16] PARASH, A. Formal Verification of an MPEG Decoder Chip - A case study in the industrial use of formal methods. **Proceedings of the Workshop on Advances in Verification (WAVe).** Chicago: July 2000.

[17] BEER, I. RuleBase: An Industry-Oriented Formal Verification Tool, **DAC'96.** California: June, 1996**.**

[18] PARK, S. Mo et al. VLSI Implementation of H.264 Video Decoder for Mobile Multimedia Application**. ETRI Journal***,* v. 28, n. 4, August, 2006*.*

[19] PEREIRA, F. **MPEG-4: Concepts, Tools and Applications**, Journal of de Réseaux et Systèms Réports-Calculateus-Parallelos, Special Issue on Image and Video, v. 12, n. 3-4, p. 299-313, April de 2000.

[20] VIRTUALDUB. Disponível em: <http://www.virtualdub.org> Acesso em 27 Novembro, 2007.

[21] RIACHARDSON, I. E. G. **H.264 and MPEG-4 Video Compression**: Video Coding for Next-generation Multimedia. England: Wiley, 2003

[22] SILVA, K. R. G. da. **Uma Metodologia de Verificação Funcional para Circuitos Digitais. 2007.** (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Paraíba, 2007.

[23] SILVEIRA, G. S.; SILVA, K. R. da, MELCHER, E. U. A Random Constrained Movie versus a Random Unconstrained Movie applied to the Functional Verification of an MPEG4 decoder design. **SIGMAP '08.** July, 2008.