DECODIFICADOR DE ÁUDIO MPEG-2 AAC-LC: IMPLEMENTAÇÃO E INTEGRAÇÃO AO TESTBENCH DO MODELO DE REFERÊNCIA DO MODULO INVERSE MODIFIED DISCRETE COSINE TRANSFORM (IMDCT).

Eliton Gonçalves da Conceição[1]; Wagner Luiz Alves de Oliveira[2].

1. Bolsista PROBIC/UEFS, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: eliton.gc@gmail.com

2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: wagner@ecomp.uefs.br

Palavra chave: Decodificador de Áudio, Verificação Funcional, IMDCT.

INTRODUÇÃO

Devido aos avanços tecnológicos, os sistemas digitais estão se tornando cada vez mais complexos para serem construídos e testados. Nesse processo, os testes são responsáveis pela maior parte do tempo de desenvolvimento de um sistema. Os sistemas são desenvolvidos baseados nas especificações, que determinam como o sistema deverá funcionar. A representação em hardware dessa especificação é nomeada de *Register Transfer Level* (RTL).

Nesse processo podem ocorrer erros nessa representação e, caso não sejam detectados, podem acarretar em elevados custos para o projeto. Tal custo pode ser tanto resultante do alto custo financeiro de enviar um projeto para a fábrica que irá confeccionar os componentes com erros, como do tempo que será gasto pela equipe de desenvolvimento do projeto para refazê-lo.

Assim, antes da obtenção das máscaras utilizadas na fabricação de um circuito integrado, é essencial que o RTL correspondente seja testado para verificar se o mesmo está funcionando de acordo com as especificações fornecidas no início do projeto, processo denominado de verificação funcional.

A verificação funcional consiste numa metodologia de teste que permite detectar erros no RTL nas fases iniciais do projeto. De acordo com (Bergeron, 2003), a verificação funcional é um processo usado para demonstrar que o objetivo do projeto é preservado em sua implementação.

Visando o desenvolvimento de um RTL sem erros, esse projeto visa à realização da verificação funcional do módulo *Filterbank and Block Switching* do Decodificador de Áudio MPEG-2 AAC-LC, a partir da implementação do modelo de referência da função seno, integrando-o ao *Testbench* do referido módulo, de forma a assegurar que a implementação RTL deste módulo esteja livre de erros.

MATERIAL E MÉTODO

Foi estudada a metodologia de projeto ipProcess, utilizado pelo Brazil-IP (BARROS, 2005). O Brazil-IP é um consórcio de universidades brasileiras, cujo objetivo é desenvolver mão-de-obra qualificada, ainda na graduação, no projeto de circuitos integrados. Para tal, este consórcio ampara-se fortemente na metodologia ipProcess, voltada para a prototipação em FPGA de sistemas digitais, de forma a validar a especificação dos mesmos, para adequação ao fluxo ASIC (*Application-Specific Integrated Circuit* – Circuito Integrado de Aplicação Específica).

Também foi realizado um estudo sobre verificação funcional, as metodologia e ferramentas utilizadas nesse processo, com um tutorial sobre criação de *Testbench* e um treinamento sobre a utilização da ferramenta *Easy Testbench Creator* (eTBC), para auxílio à criação de *testbenches.*

O eTBC é uma ferramenta de auxílio à criação de *testbenches*, a qual visa automatizar e acelerar o processo de construção do ambiente de verificação, gerando as ligações entre os componentes de um *testbench*. A arquitetura do *testbench* proposta pelo Brazil-IP é mostrada na Figura 3.

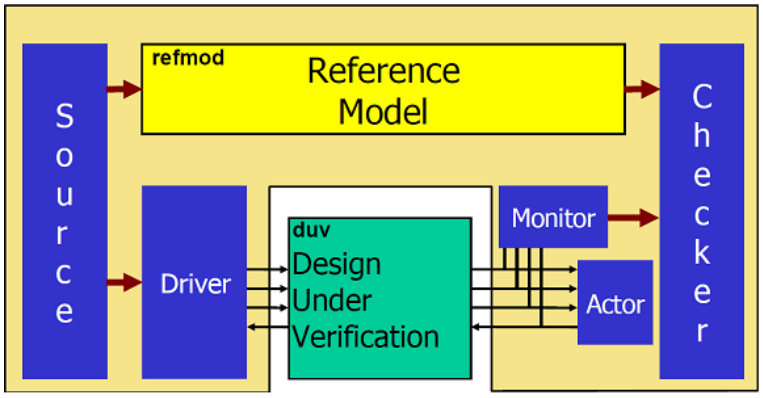

Figura 3. Testbench (Brazil-IP).

O *testbench* é o ambiente responsável pela realização da verificação de um projeto, ele é composto dos seguintes elementos: ***Source***, ***Driver***, ***Monitor***, ***Reference Model*** (**RM**) e ***Checker***. O mecanismo de sincronização do *testbench* é implementado por meio de uma fila **FIFO** (*First-In-First-Out*), através de transações. As funcionalidades de cada um desses blocos são: O ***Source*** é o responsável por prover dados para o **DUV** e para o Modelo de Referência (**RM**). É conectado ao **RM** e ao ***Driver*** por meio de FIFOs. Existe um **FIFO** para cada interface de entrada do **DUV**. O mesmo número de FIFOs vai para o **RM** e para o ***Driver***.

O ***Driver*** recebe dados a nível de transação do ***Source***, traduz para protocolo de sinais e passa-os para o **DUV**. Funciona como uma ponte entre nível de transação e nível de sinal. Existe sempre um ***Driver*** para cada interface de entrada do **DUV**.

O ***Monitor*** é uma ponte entre os sinais e as transações. Responsável por receber os dados a nível de sinais do **DUV** e transformá-los em dados a nível de transação. O ***Monitor*** põe o dado num **FIFO** e passa para o ***Checker***. Existe sempre um ***Monitor*** para cada interface de saída do **DUV**.

O ***Checker*** responsável pela comparação entre os resultados a nível de transação vindos do **RM** e do ***Monitor*** para verificar se eles são equivalentes. O ***Checker*** compara automaticamente as saídas do **RM** e do ***Monitor*** e imprime mensagens de erro se elas não forem equivalentes.

O *Reference Model* (**RM**) é a implementação ideal da funcionalidade do sistema. Recebe dados a nível de transação do ***Source*** através de um **FIFO** e envia dados também a nível de transação em um **FIFO** para o ***Checker***.

O **DUV** é a unidade que está sendo testada, ou seja, o projeto que está sendo desenvolvido pela equipe.

RESULTADOS E DISCUSSÃO

No início do desenvolvimento do projeto houve um remanejamento de atividades e, ao invés de desenvolver o modelo de referência da IMDCT, prosseguiu-se o desenvolvimento do *testbench* do *Windowing/Block Switching*, isso foi possível graças finalização do RTL do mesmo, após realização desta verificação foi desenvolvido o modelo referência da IMDCT, o modelo de referencia foi desenvolvido utilizando linguagem de programação C.

$$x_n = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} spec[k]\cos\left(\frac{2\pi}{N}(n-n_0)\left(k+\frac{1}{2}\right)\right) \; para \; 0 \leq n \leq N \qquad (1)$$

$$n_0 = \left(\frac{N}{2}+1\right)/1 \qquad (2)$$

Na equação em (1) tem-se a equação da IMDCT, onde N pode assumir os valores 256 ou 2048, dependendo do tipo de janela processada pelo decodificador, ao passo que n varia de 0 a 255 ou de 0 a 2047, também em função da janela processada. Após o desenvolvimento, o modelo de referência foi testado e os resultados gerados foram satisfatórios, de acordo com a margem de erro esperada devido a arredondamentos da representação em ponto flutuante da linguagem. Os *specs* correspondem ao vetor contendo as informações do áudio decodificado.

A Figura 1 mostras a implementação da IMDCT realizada em linguagem C.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PI 3.14159265359

double IMDCT(int sequence, int n, double *spec){
    int n0, N, k;

    double soma = 0.0;

    if(sequence == 2)
          N = 256;
    else
          N = 2048;

    n0 = (N/2 + 1)/2;
    for(k = 0; k < N/2; k++){
          soma = soma + spec[k]*cos((2*PI/N)*(n + n0)*(k + 1/2));
    }
    soma = 2.0/N * soma;
    return soma;
}
```

Figura 3: Código em linguagem C da IMDCT.

A Tabela 1 mostra somente alguns dos valores obtidos com o modelo de referência, uma vez que a tabela completa possui 2048 entradas, os valores foram gerados a partir de valores aleatórios de *specs* mostrado equação 1.

Prosseguiu-se com o desenvolvimento do *testbench*, para realização da verificação funcional do módulo, contudo, como o RTL do módulo ainda não encontra-se finalizado, não foi possível a conclusão da verificação do mesmo. Sendo assim, a conclusão desta verificação fica dependente da finalização do RTL.

Tabela 1. Resultados gerados pelo modelo de referência da IMDCT

Para N = 256	Para N = 2048
IMDCT 0: 3.590078	IMDCT 0: 11.179922

IMDCT 20: 1.409896	IMDCT 2: 10.023532
IMDCT 50: 3.179433	IMDCT 77: 6.165244
IMDCT 75: 3.062534	IMDCT 103: 6.782750
IMDCT 92: 2.519299	IMDCT 126: 10.023532
IMDCT 107: 1.603619	IMDCT 130: 12.545793
IMDCT 130: 0.084542	IMDCT 227: 0.544690
IMDCT 135: 0.624254	IMDCT 240: 1.598975
IMDCT 152: 1.788522	IMDCT 354: 2.781845
IMDCT 178: 2.641439	IMDCT 597: 5.375316
IMDCT 204: 3.428707	IMDCT 779: 7.996041
IMDCT 253: 1.262527	IMDCT 1024: 11.179922

CONCLUSÃO

Diante dos resultados apresentados neste relatório, é possível perceber que os objetivos do projeto foram alcançados de maneira satisfatória e, mesmo mediante as dificuldades encontradas, o bolsista, o professor orientador e a equipe de desenvolvimento encontraram boas soluções para o cumprimento das tarefas previstas no projeto. É importante salientar, ainda, que este trabalho contribuiu para o desenvolvimento do projeto maior, o Decodificador de Áudio MPEG-2 AAC-LC do programa Brazil-IP/UEFS.

Este trabalho, e mais especificamente o projeto Brazil-IP, são de extrema importância para a formação acadêmica de um Engenheiro de Computação, pois proporcionam um aprendizado e uma experiência única, as quais serão determinantes para uma possível atuação profissional, seja na área acadêmica, seja no mercado de trabalho. Enfim, as informações adquiridas e as experiências vivenciadas no desenvolvimento de um trabalho deste porte proporcionam uma formação diferenciada para os estudantes. Outro aspecto importante proporcionado pelo projeto é a interação com diferentes grupos de pesquisa de todo o país.

Por fim, o desenvolvimento do projeto como um todo elevou o nível de conhecimento teórico e prático do bolsista, em relação ao processo de verificação funcional e, também, de outros conceitos referentes ao desenvolvimento de sistemas digitais.

REFERÊNCIAS

BARROS, E. et al. ipprocess: Using a process to teach ip-core development. Microelectronics Systems Education, IEEE International Conference on/Multimedia Software Engineering, International Symposium on, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 27-28, 2005.

BERGERON, J. WritingTestbenches using System Verilog. 233 Spring Street, New York, NY 10013, USA: Springer, 2003.

BRAZIL-IP. Rede brasileira de centros de concepção de Sistemas Digitais e IP-Cores. 2002. Disponível em: <http://www.brazilip.org.br/> 07 Jul. 2011.