Juan Manuel Gago Benitez

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

DVBT IP Core

Verification

Contents

[Revision history 2](#_Toc370064334)

[Introduction 3](#_Toc370064335)

[Testbench 3](#_Toc370064336)

[Simulation Procedure 4](#_Toc370064337)

[Code Coverage Report 5](#_Toc370064338)

# 

# Revision history

|  |  |
| --- | --- |
| Version | Comment |
| V1.0 | First release |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

This document details the DVBT IP core testbench. The testbench currently covers the following core features.

# Testbench

The testbench consists in a DVBT RX IP Core connected to a DVBT TX IP core. There is a source of video and audio feeding the DVBT TX IP core.

The testbench is shown in Figure 1.



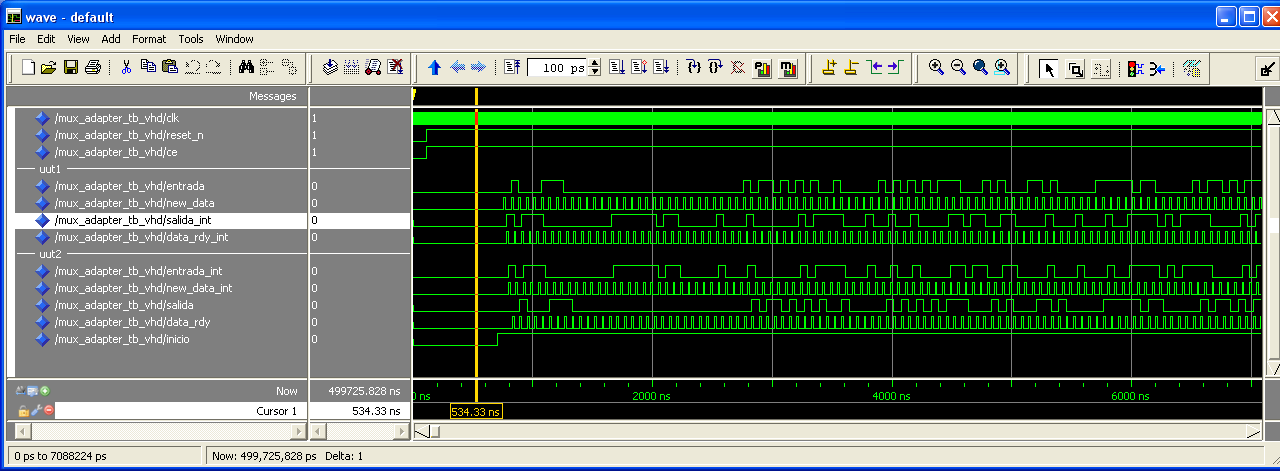
Figure 1 DVBT Testbench

Figure 2 Mux Adapter testbench (mux\_adapter\_tb)

Byte Pa-rallelizer

Byte Serializer

Figure 3 Mux Adapter (reversible)



El proceso para introducción de datos por la entrada del testbench es como sigue.

* Utilizamos el flanco descendente para no penalizar un ciclo.
* Utilizamos la variable cuenta\_ciclos (integer) para mandar un byte al mux\_adapter cada 16 ciclos de reloj de este modo nunca se satura el buffer de entrada.

|  |
| --- |
| if cuenta\_ciclos = 16 then  entrada <= conv\_std\_logic\_vector(valor, 8);  new\_data <= '1';  cuenta\_ciclos := 0;  else  new\_data <= '0';  cuenta\_ciclos := cuenta\_ciclos + 1;  end if; |

This testbench is located at the /sim folder. The associated files are

* dvbt\_core\_tb.v: testbench with the instantiation of cores and generators.
* pixel\_exapander.v: used to convert 1-pixel-per-clock to 2-pixel-per-clock.
* autotest\_crc.v: 16 bit

# Simulation Procedure

* Open the file **bv\_sim.mpf** at the /sim folder
* Create the working design library. Type at the modelsim> prompt
  + vlib work; vmap work work
* In the Project window select Compile > Compile All from the pop-up menu
* Click on **dvbt\_sim** to load the design
* Add all signals to the Wave window. Type at the vsim> prompt
  + add wave -r sim:/dvbt\_core\_tb/\*; run -all
* Verify the transmitted and received video

Table 1 Test pattern for symbols\_per\_clock = 1

|  |  |  |  |
| --- | --- | --- | --- |
| bpc | Video Clock | Link Clock Rate | Test Pattern Gen. (\*) |
| 8 | 80 MHz | 8 / 8 | 00: 01: c7 |
| 10 | 80 MHz | 10 / 8 | 0000: 0040: 31c0 |

(\*) Matlab notation. Values in hexadecimal.

Table 2 Test pattern for symbols\_per\_clock = 2

|  |  |  |  |
| --- | --- | --- | --- |
| Bpc | Video Clock | Link Clock Rate | Test Pattern Gen. (RX) |
| 8 | 40 MHz | 8 / 8 | {00,01}: 01: {c6,c7} |
| 10 | 40 MHz | 10 / 8 | {0000, 0040}: 0040: {3180,31c0} |

## Code Coverage Report

* Open the project and compile the sources. Type at the modelsim> prompt
  + do dvbt\_exclusions.do
* Click on **dvbt\_sim\_coverage** to load the design
* Run the simulation
* Select Tools > Coverage Report > HTML from the Main window menu bar

