CSE 577 - Verification Plan

Quang Nguyen and Shenbo Li

Contents

[Objective 1](#_Toc449626164)

[Canonical Data 1](#_Toc449626165)

[Testing Plan 1](#_Toc449626166)

[Approach and structure 1](#_Toc449626167)

[Test Cases 2](#_Toc449626168)

[Fully Pipelined 2](#_Toc449626169)

[Changing Valid Signal 2](#_Toc449626170)

[Largest Values 2](#_Toc449626171)

[Smallest Values 2](#_Toc449626172)

[All Non-zero Skinscore Values 2](#_Toc449626173)

# Objective

This report outlines the verification plan for the skintone detector module.

# Canonical Data

The canonical data that will be used for comparing the output results of each test cases will be generated using the provided C code. TCr, TCb, xx, yy, skinscore, and return values are generated for each increment of 5 of Y, Cr and Cb. The attached excel spreadsheet, *“canonical\_data.xlsx”,* contains the values of these variables.

# Testing Plan

## Approach and structure

In order to ensure that each of the test cases are easily reproducible, the test bench for each of the test cases will drive the skintone detector module with the desired Y, Cr, and Cb values and write the resulting output values to a csv file for later viewing.

The output csv file will have 3 columns. The first column will contains the current clock cycle. The second column will contains valid\_out signal for the corresponding clock cycle. The last column will contains the returned skinscore value for each of the corresponding clock cycle.

## Test Cases

### Fully Pipelined

The first thing that the verification plan should check is whether or not the module is properly pipelined or not. The test case to test this can be found in the attached file *“tc\_fullyPipelined.sv”*. This test case will drive the module with 3 different Y, Cr, and Cb values. The output csv file should show that the module is producing the proper value at every cycle.

### Changing Valid Signal

This test case tests whether or not the valid\_out signal is working as expected. The test case can be found in the attached file *“tc\_changingValid.sv”*. It functions by driving the module with 9 different Y, Cr, and Cb values. The first and last 3 values will have a valid signal of 1. The output csv file should show the proper valid\_out pattern of 1, 1, 1, 0, 0, 0, 1, 1, 1.

### Largest Values

It is more likely for something to go wrong at the max value of TCr, TCb, xx, yy, or skinscore value; therefore, this test case used the canonical data to find the max value for each of these variables and drive the value of Y, Cr, and Cb to the value that corresponds with the desired variable. The test case to test this can be found in the attached file *“tc\_largestValues.sv”*. The output csv file should show that the module is producing the proper value at every cycle.

### Smallest Values

It is more likely for something to go wrong at the smallest value of TCr, TCb, xx, yy, or skinscore value; therefore, this test case used the canonical data to find the minimum value for each of these variables and drive the value of Y, Cr, and Cb to the value that corresponds with the desired variable. The test case to test this can be found in the attached file *“tc\_smallestValues.sv”*. The output csv file should show that the module is producing the proper value at every cycle.

### All Non-zero Skinscore Values

This test case uses the canonical data to find only find combinations of Y, Cr, and Cb that produce skinscore values that are greater than 0. The test case to test this can be found in the attached file *“tc\_allSkinscoreValues.sv”*. The output csv file should show that the module is producing the proper value at every cycle.