Visible Image Watermarking

Digital Design and Logical Synthesis for Electric Computer Engineering

(36113611)

Course Project

Digital High-Level Design

Version 0.1

Revision Log

Rev	Description	Done By	Date
0.1	Initial document	Ariel Moshe, Amit Nagar Halevy	30/11/2020
0.2	Digital Changes Functional Verification	Ariel Moshe, Amit Nagar Halevy	05/12/2020
0.3			

Table of Content

1. LIST OF FIGURES

2. LIST OF TABLES

3. VERIFICATION PLAN

3.1 General Description

The top module for testing system design is ‘tb_overall’, contains all the modules for testing our device under test (DUT).

The ‘tb_overall’ consist of six main modules as following:

1) Interface – contains all the different buses between all the modules and the DUT.
2) Stimulus – in charge of inserting input images and parameters into the DUT using AMBA protocol.
3) DUT – our device under test - Visible Watermarking.
4) Coverage – determines how much functionality of the design has been exercised. Functional assertions are used to check whether each and every corner of the design is explored and functions properly.
5) Checker – Checking that values are within an acceptable range, set according to the proper order. In addition, does properties checking (input validity, output validity, etc.)
6) Golden Model – compare between the DUT and MATLAB watermarking images result and saves the DUT images in txt file.

3.2 Full System Architecture

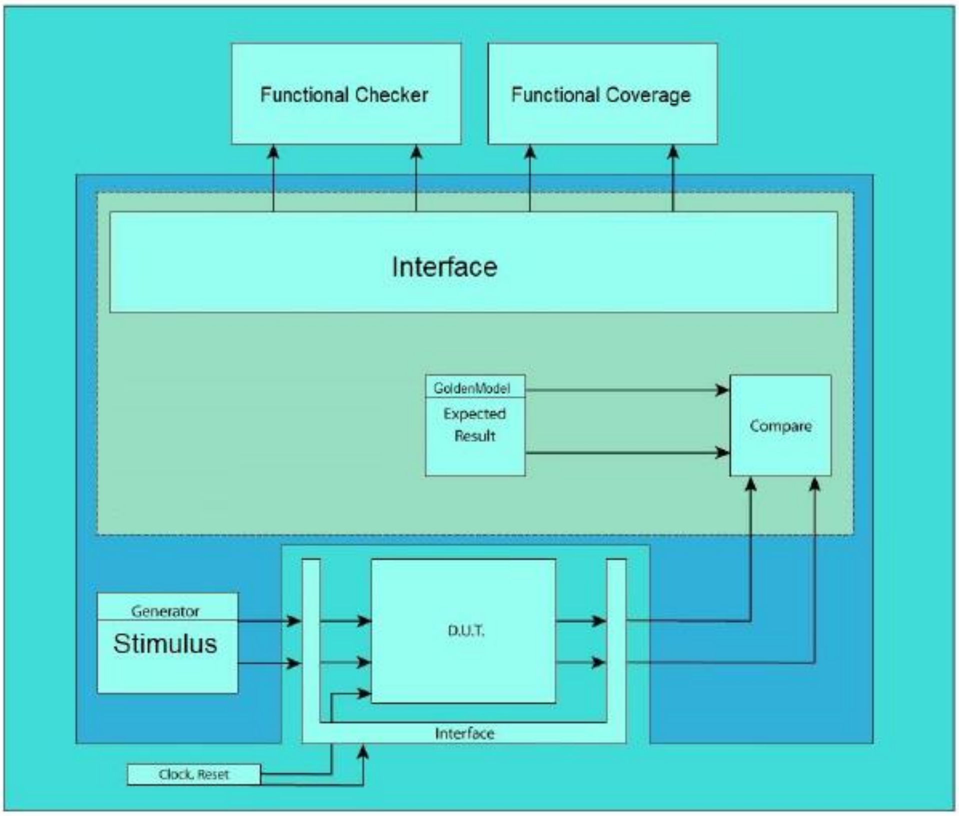


Figure 1: Full System Architecture - Top Testbench.

3.3 Functional Coverage

FUNCTION	EVENT	COVERAGE POINT	BINS	scenario
Reset	**Posedge clock**	**rst**	**0,1**	**Standard**
Address	**Posedge clock**	**PADDR**	**Parameters - [0:9]** **images - [10:2^(Amba_Address -1)]**	**Standard**
PENABLE	**Posedge clock**	**PENABLE**	**0,1**	**Standard**
PSEL	**Posedge clock**	**PSEL**	**0,1**	**Standard**
PWRITE	**Posedge clock**	**PWRITE**	**0,1**	**Standard**

PWDATA	**Posedge clock**	**PWDATA**	**[0:AMBA_Word -1]**	**Standard**
Image_Done	**Posedge clock**	**Image_Done**	**0,1**	**Standard**
Pixel_Data	**Posedge clock**	**Pixel_Data**	**[0:255]**	**Standard**
new_pixel	**Posedge clock**	**new_pixel**	**0,1**	**Standard**
Msize	**Posedge clock & PADDR == 4**	**PWDATA**	**MsizeLow - [1:10]** **MsizeMid - [11:49]** **MsizeHigh - [50:72]**	**Standard/ Extreme**
picture_size	**Posedge clock & PADDR == 2**	**PWDATA**	**vsmall - [200:300]** **regular - [0:619]** **vlarge - [620:720]**	**Standard/ Extreme**
pixel_value	**Posedge clock & PADDR > 9**	**PWDATA**	**plow - [0:40]** **pmid - [41:214]** **phigh - [215:255]**	**Standard/ Extreme**

Table 1: Test Plan Functional Coverage.

3.4 Functional Checker

Condition	Event	Expected Result	info	Scenario
Reset == 1	**Reset**	**Pixel_Data→0|X**	**Check Pixel_Data on reset**	**Standard**
new_pixel ==1	**new_pixel**	**Pixel_Data→[0:255]**	**Check Pixel_Data on new_pixel**	**Standard**
Image_Done == 1	**Image_Done**	**Count == N*N** **Output image size == Input image size**	**Check that all pixels received on Image_Done**	**Standard**

Table 2: Test Plan Functional Checker.

4. VERIFICATION IMPLEMENTATION

4.1 Functional Coverage

4.1.1 Functional Description

Collect data on all DUT port (input/output) and have only input ports. This module composed of different cover groups that arrange signals values in bins. It makes sure that the test covered all standard and extreme scenarios.

4.1.2 Interface

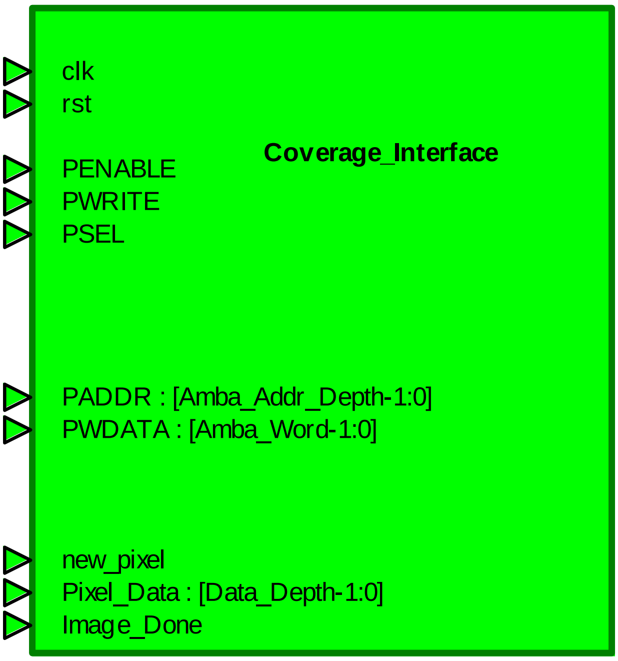


Figure 2: Functional Coverage interface.

4.2 Functional Checker

4.2.1 Functional Description

Collect data on all DUT port (input/output), have only input ports. This module checks the outputs functionality according to different input, using System Verilog Assertions (SVA). This method gives us a binary answer to the question if the design is functional.

4.2.2 Interface

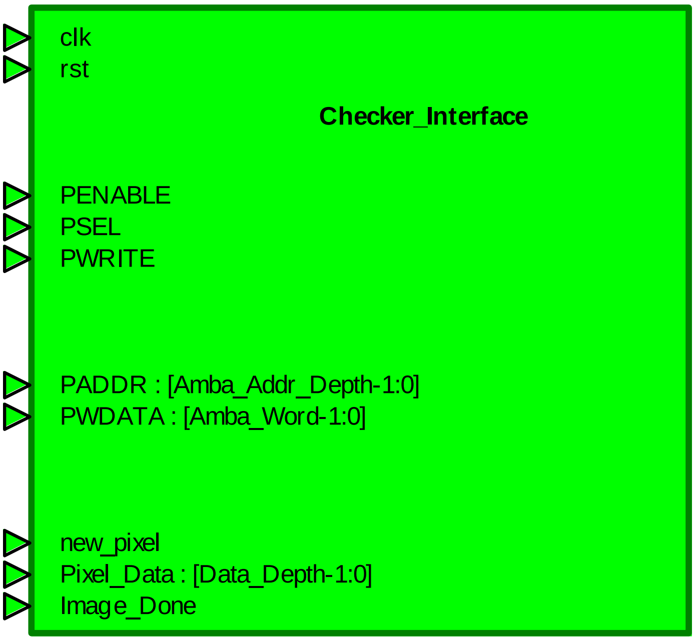


Figure 3: Functional Checker interface.

4.3 Stimulus

4.3.1 Functional Description

Only have output ports, that are the input DUT's port.

Inserting input images and parameters into the DUT using AMBA protocol

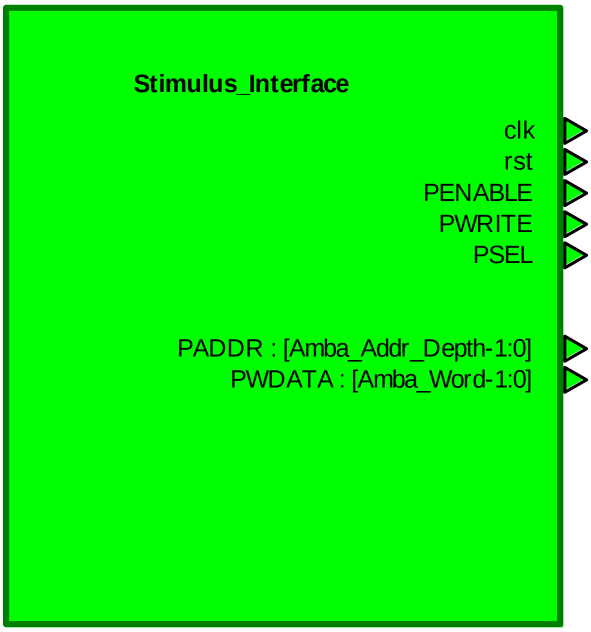


Figure 4: Stimulus interface.

4.4 Golden-Model

4.4.1 Functional Description

Only have input ports, that are the output DUT’s port.

Compare between the DUT and MATLAB watermarking images result and saves the DUT images in txt file. Calculate Mean Squared Error between the MATLAB Golden-Model and DUT.

4.4.2 Interface

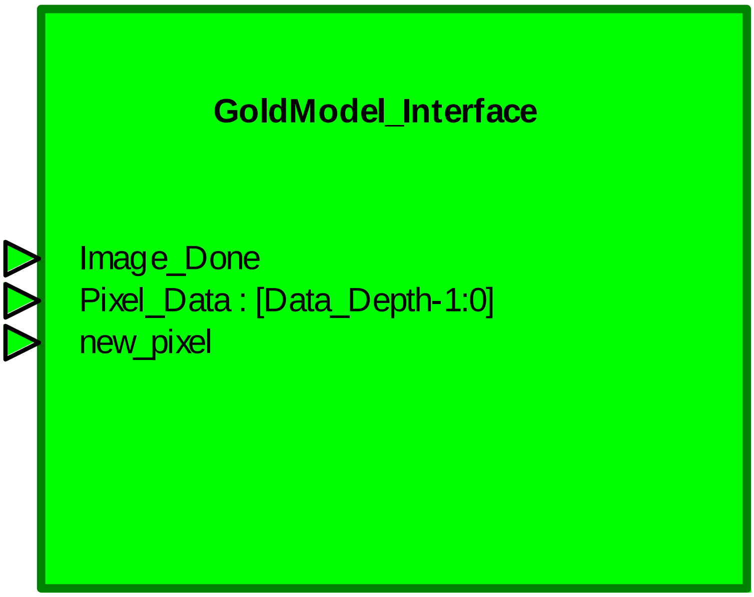


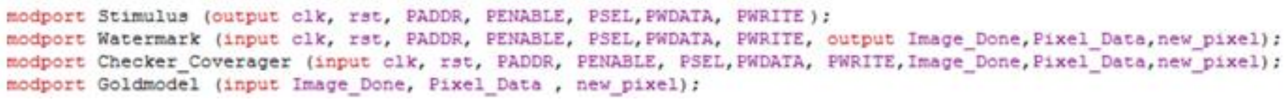
Figure 5: Golden-Model interface.

4.5 Interface & Overall TB

4.5.1 Functional Description

The top module for testing system design is ‘tb_overall’, contains all the modules for testing our device under test (DUT), connected by the interface. Same as the verification plan except that the Golden-Model and Compare blocks are combined.

4.5.2 Interface’s modports:



```
modport Stimulus (output clk, rst, PADDR, PENABLE, PSEL,PWDATA, PWRITE );
modport Watermark (input clk, rst, PADDR, PENABLE, PSEL,PWDATA, PWRITE, output Image_Done,Pixel_Data,new_pixel);
modport Checker_Coverager (input clk, rst, PADDR, PENABLE, PSEL,PWDATA, PWRITE,Image_Done,Pixel_Data,new_pixel);
modport Goldmodel (input Image_Done, Pixel_Data , new_pixel);
```

Figure 6: Interface’s modports.

4.5.3 Block Diagram

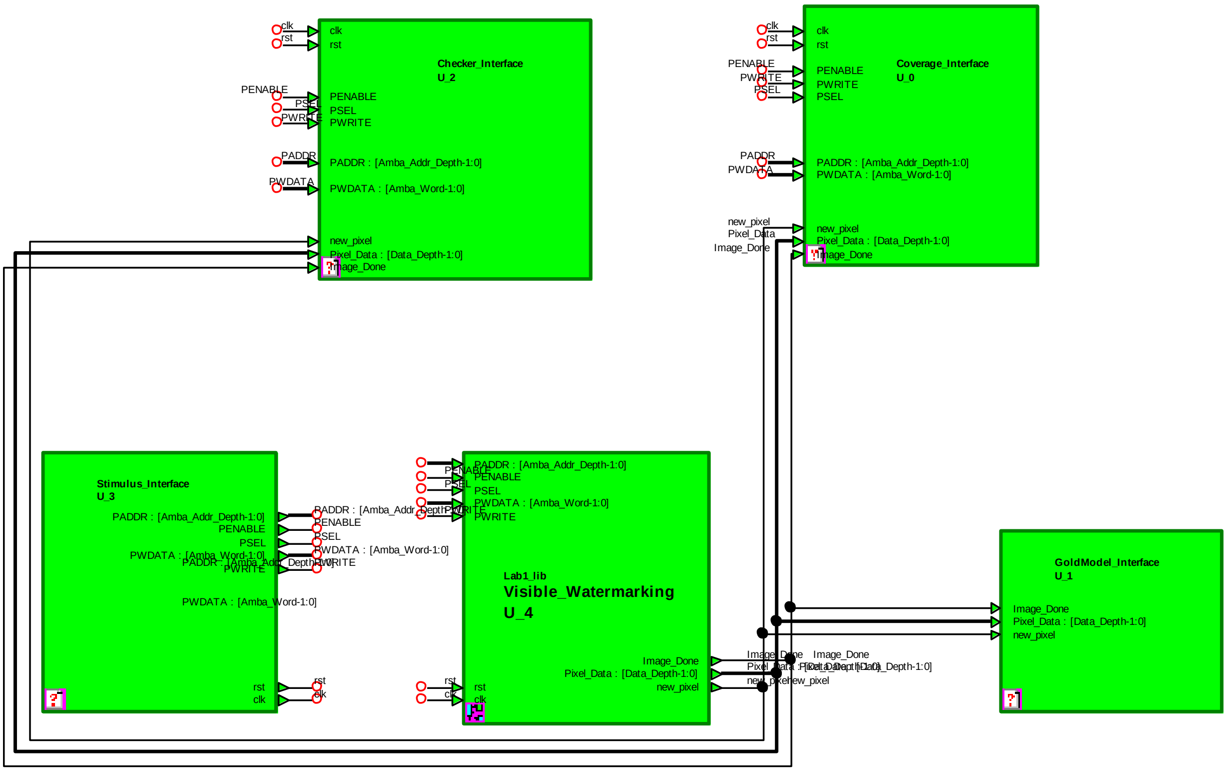
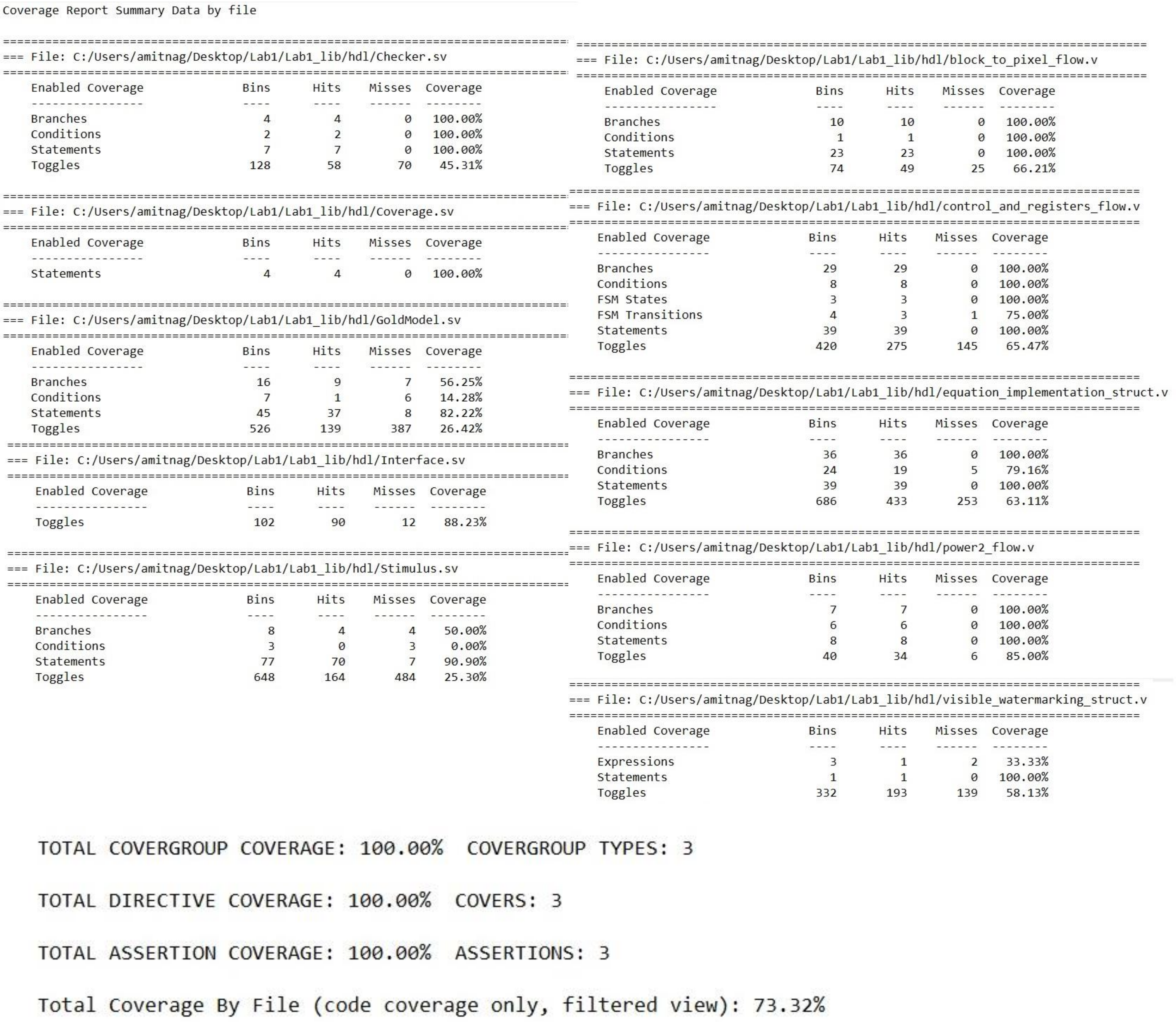


Figure 7: Overall TB block diagram.

5. VERIFICATION RESULTS



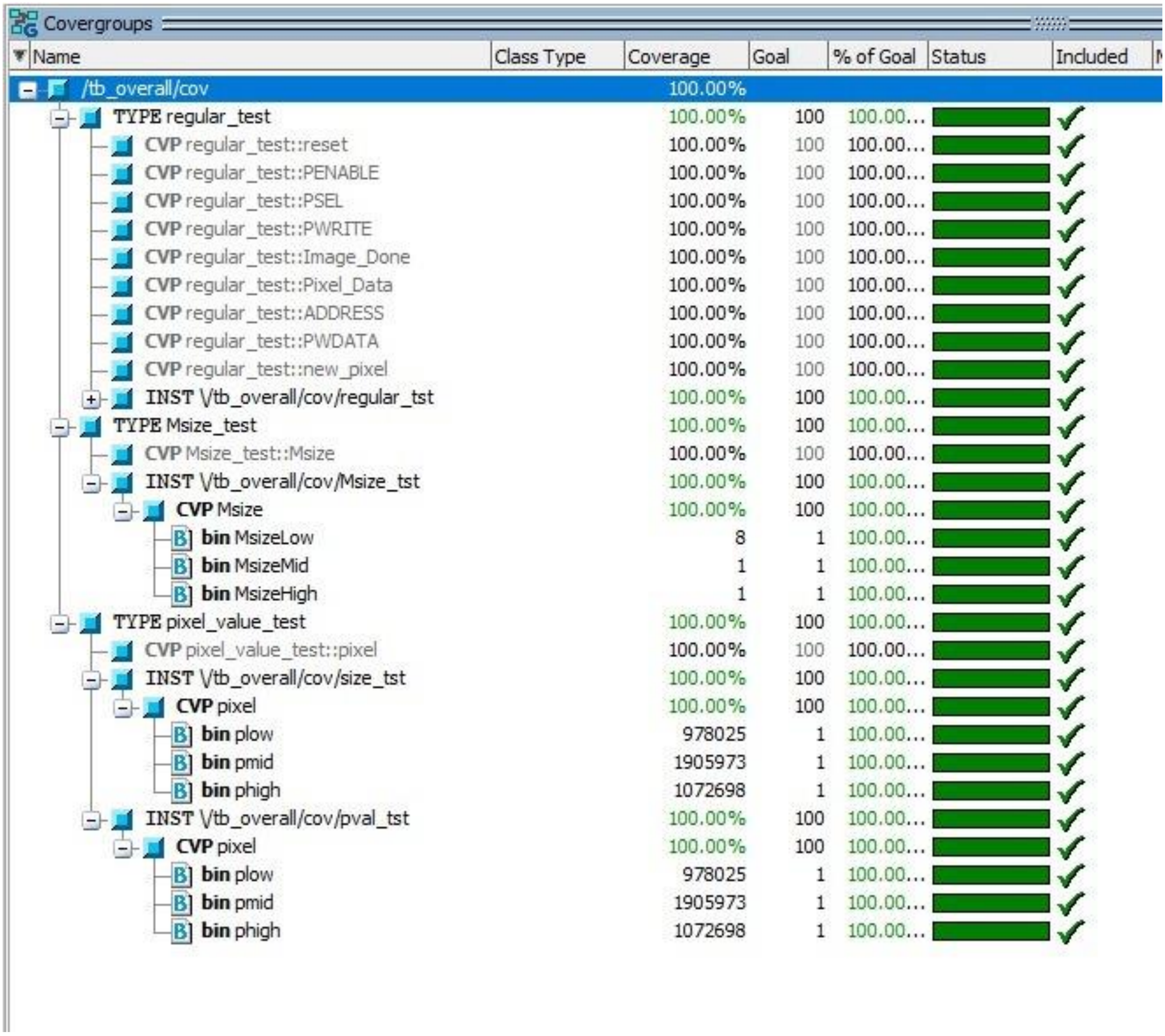
```
Coverage Report Summary Data by file

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Checker.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         4         4         0   100.00%
    Conditions                       2         2         0   100.00%
    Statements                       7         7         0   100.00%
    Toggles                        128        58        70    45.31%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Coverage.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       4         4         0   100.00%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/GoldModel.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        16         9         7    56.25%
    Conditions                       7         1         6    14.28%
    Statements                      45        37         8    82.22%
    Toggles                        526       139       387    26.42%
================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Interface.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        102        90        12    88.23%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Stimulus.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         4         4    50.00%
    Conditions                       3         0         3     0.00%
    Statements                      77        70         7    90.90%
    Toggles                        648       164       484    25.30%
```

```
================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/block_to_pixel_flow.v
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%
    Conditions                       1         1         0   100.00%
    Statements                      23        23         0   100.00%
    Toggles                         74        49        25    66.21%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/control_and_registers_flow.v
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        29        29         0   100.00%
    Conditions                       8         8         0   100.00%
    FSM States                       3         3         0   100.00%
    FSM Transitions                  4         3         1    75.00%
    Statements                      39        39         0   100.00%
    Toggles                        420       275       145    65.47%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/equation_implementation_struct.v
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        36        36         0   100.00%
    Conditions                      24        19         5    79.16%
    Statements                      39        39         0   100.00%
    Toggles                        686       433       253    63.11%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/power2_flow.v
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%
    Conditions                       6         6         0   100.00%
    Statements                       8         8         0   100.00%
    Toggles                         40        34         6    85.00%

================================================================================
=== File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/visible_watermarking_struct.v
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Expressions                      3         1         2    33.33%
    Statements                       1         1         0   100.00%
    Toggles                        332       193       139    58.13%
```

```
TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 3

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 3

TOTAL ASSERTION COVERAGE: 100.00%  ASSERTIONS: 3

Total Coverage By File (code coverage only, filtered view): 73.32%
```

Figure 8: Full Report.

5.1 Functional Coverage Report



Covergroups

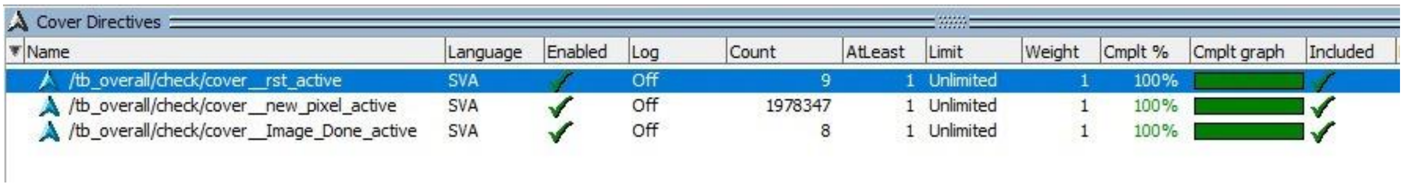
Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/tb_overall/cov		100.00%				
TYPE regular_test		100.00%	100	100.00...	🟩	✓
CVP regular_test::reset		100.00%	100	100.00...	🟩	✓
CVP regular_test::PENABLE		100.00%	100	100.00...	🟩	✓
CVP regular_test::PSEL		100.00%	100	100.00...	🟩	✓
CVP regular_test::PWRITE		100.00%	100	100.00...	🟩	✓
CVP regular_test::Image_Done		100.00%	100	100.00...	🟩	✓
CVP regular_test::Pixel_Data		100.00%	100	100.00...	🟩	✓
CVP regular_test::ADDRESS		100.00%	100	100.00...	🟩	✓
CVP regular_test::PWDATA		100.00%	100	100.00...	🟩	✓
CVP regular_test::new_pixel		100.00%	100	100.00...	🟩	✓
INST \/tb_overall/cov/regular_tst		100.00%	100	100.00...	🟩	✓
TYPE Msize_test		100.00%	100	100.00...	🟩	✓
CVP Msize_test::Msize		100.00%	100	100.00...	🟩	✓
INST \/tb_overall/cov/Msize_tst		100.00%	100	100.00...	🟩	✓
CVP Msize		100.00%	100	100.00...	🟩	✓
bin MsizeLow		8	1	100.00...	🟩	✓
bin MsizeMid		1	1	100.00...	🟩	✓
bin MsizeHigh		1	1	100.00...	🟩	✓
TYPE pixel_value_test		100.00%	100	100.00...	🟩	✓
CVP pixel_value_test::pixel		100.00%	100	100.00...	🟩	✓
INST \/tb_overall/cov/size_tst		100.00%	100	100.00...	🟩	✓
CVP pixel		100.00%	100	100.00...	🟩	✓
bin plow		978025	1	100.00...	🟩	✓
bin pmid		1905973	1	100.00...	🟩	✓
bin phigh		1072698	1	100.00...	🟩	✓
INST \/tb_overall/cov/pval_tst		100.00%	100	100.00...	🟩	✓
CVP pixel		100.00%	100	100.00...	🟩	✓
bin plow		978025	1	100.00...	🟩	✓
bin pmid		1905973	1	100.00...	🟩	✓
bin phigh		1072698	1	100.00...	🟩	✓

Figure 9: Functional Coverage Report

All the checks in the functional coverage report passed 100%, that means that all DUT's ports reached all desired scenarios. For more details look on Table 1 - Test Plan Functional Coverage.

At the initial run, we got on the Msize_test only 66% coverage. The MsizeHigh were 0, meaning we didn't test any block size (M) that was bigger than 50 pixels, so we designed a specific test image with $M = 57$.

5.2 Functional Checker Report



Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/tb_overall/check/cover__rst_active	SVA	✓	Off	9	1	Unlimited	1	100%	🟩	✓
/tb_overall/check/cover__new_pixel_active	SVA	✓	Off	1978347	1	Unlimited	1	100%	🟩	✓
/tb_overall/check/cover__Image_Done_active	SVA	✓	Off	8	1	Unlimited	1	100%	🟩	✓

Figure 10: Functional Checker Report

All the checks in the functional checker report passed 100%.

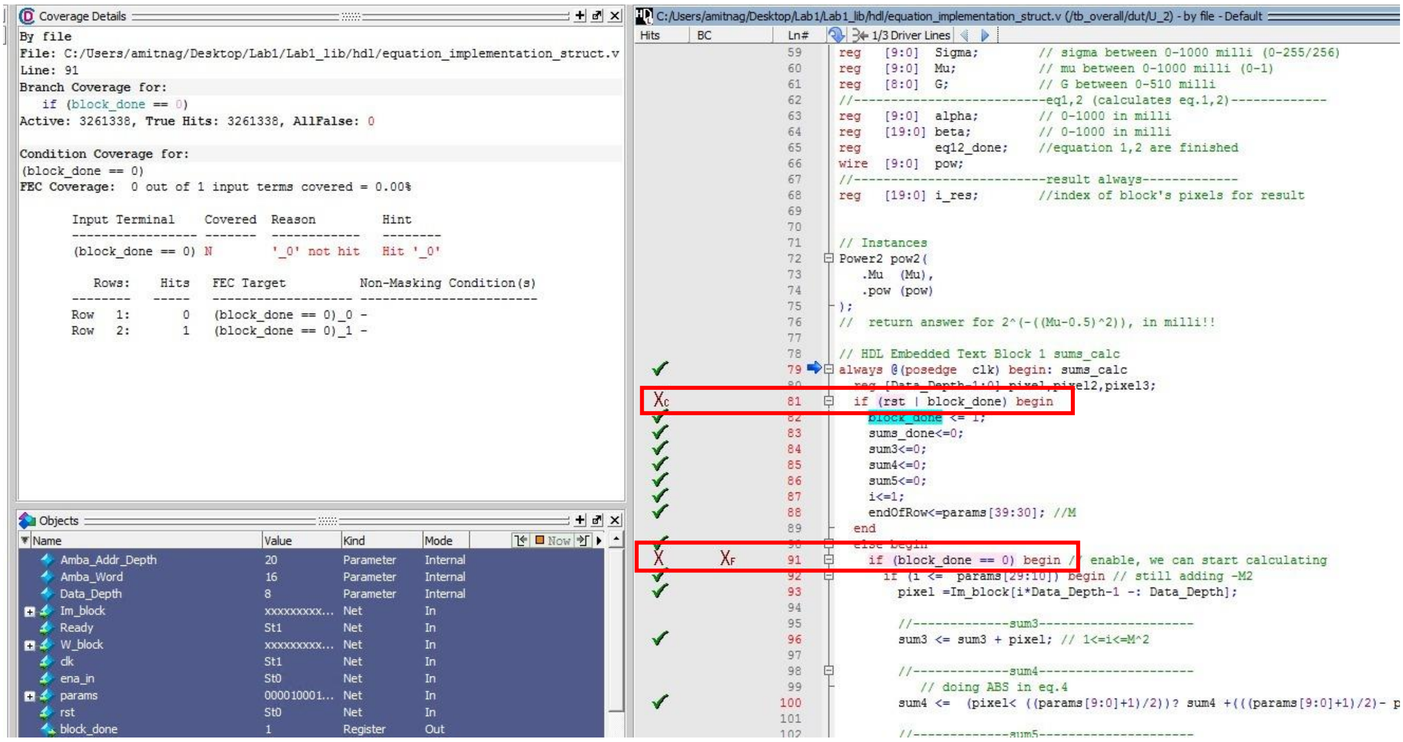
During simulations, Image_Done_Active test was incorrect. Therefore, we deduced that not all the pixels of the image received, so we checked the DUT code and notice that there is double readding from the registers at same clock and fixed it.

In addition, new_pixel_active test was incorrect. We got X's outputs instead of number in the range, then we find that there is synchronization problem between the modules.

We used Questasim wave simulation to identify the problem and to find the location of the error in the code.

5.3 Code Coverage Report

The following figures illustrate errors discovered using the code coverage report:



Coverage Details

```
By file
File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/equation_implementation_struct.v
Line: 91
Branch Coverage for:
   if (block_done == 0)
Active: 3261338, True Hits: 3261338, AllFalse: 0

Condition Coverage for:
(block_done == 0)
FEC Coverage:  0 out of 1 input terms covered = 0.00%

       Input Terminal    Covered  Reason        Hint
       ----------------- -------  ------------  --------
       (block_done == 0) N        '_0' not hit  Hit '_0'

          Rows:    Hits   FEC Target          Non-Masking Condition(s)
       --------   -----   ------------------- ------------------------
       Row   1:       0   (block_done == 0)_0 -
       Row   2:       1   (block_done == 0)_1 -
```

Objects

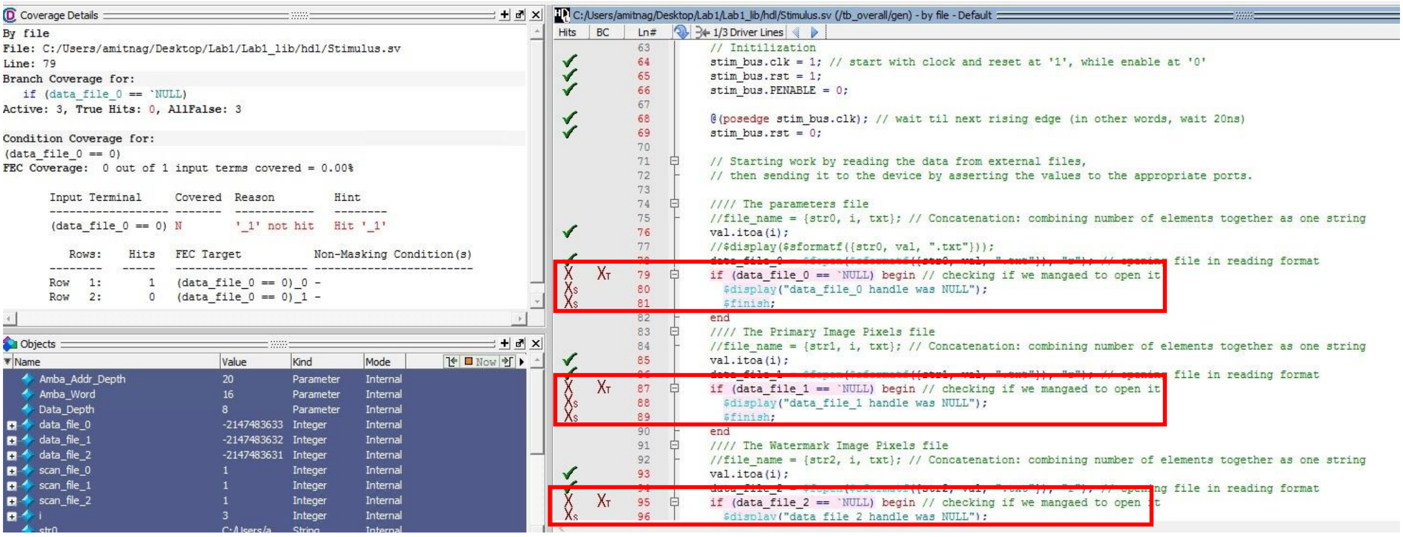
Name	Value	Kind	Mode
Amba_Addr_Depth	20	Parameter	Internal
Amba_Word	16	Parameter	Internal
Data_Depth	8	Parameter	Internal
Im_block	xxxxxxxxx...	Net	In
Ready	St1	Net	In
W_block	xxxxxxxxx...	Net	In
clk	St1	Net	In
ena_in	St0	Net	In
params	000010001...	Net	In
rst	St0	Net	In
block_done	1	Register	Out

C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/equation_implementation_struct.v (/tb_overall/dut/U_2) - by file - Default

```
Hits  BC   Ln#
           59   reg   [9:0]  Sigma;        // sigma between 0-1000 milli (0-255/256)
           60   reg   [9:0]  Mu;           // mu between 0-1000 milli (0-1)
           61   reg   [8:0]  G;            // G between 0-510 milli
           62   //-------------------------eq1,2 (calculates eq.1,2)------------
           63   reg   [9:0]  alpha;        // 0-1000 in milli
           64   reg   [19:0] beta;         // 0-1000 in milli
           65   reg          eq12_done;    //equation 1,2 are finished
           66   wire  [9:0]  pow;
           67   //-------------------------result always------------
           68   reg   [19:0] i_res;        //index of block's pixels for result
           69
           70
           71   // Instances
           72   Power2 pow2(
           73      .Mu   (Mu),
           74      .pow  (pow)
           75   );
           76   //  return answer for 2^(-((Mu-0.5)^2)), in milli!!
           77
           78   // HDL Embedded Text Block 1 sums_calc
✓          79   always @(posedge  clk) begin: sums_calc
           80     reg [Data_Depth-1:0] pixel,pixel2,pixel3;
Xc         81     if (rst | block_done) begin
✓          82       block_done <= 1;
✓          83       sums_done<=0;
✓          84       sum3<=0;
✓          85       sum4<=0;
✓          86       sum5<=0;
✓          87       i<=1;
✓          88       endOfRow<=params[39:30]; //M
           89     end
✓          90     else begin
X     XF   91       if (block_done == 0) begin // enable, we can start calculating
✓          92         if (i <=  params[29:10]) begin // still adding -M2
✓          93           pixel =Im_block[i*Data_Depth-1 -: Data_Depth];
           94
           95           //------------sum3--------------------
✓          96           sum3 <= sum3 + pixel; // 1<=i<=M^2
           97
           98           //------------sum4--------------------
           99             // doing ABS in eq.4
✓          100          sum4 <=  (pixel< ((params[9:0]+1)/2))? sum4 +(((params[9:0]+1)/2)- p
           101
           102          //------------sum5--------------------
```

Figure 11: Code Coverage Report ex.1

Example of unnecessary ‘if’ statement in the code, thus we removed the second line.



```
Coverage Details
By file
File: C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Stimulus.sv
Line: 79
Branch Coverage for:
   if (data_file_0 == `NULL)
Active: 3, True Hits: 0, AllFalse: 3

Condition Coverage for:
(data_file_0 == 0)
FEC Coverage:  0 out of 1 input terms covered = 0.00%

      Input Terminal     Covered  Reason        Hint
      ------------------ -------  ------------  --------
      (data_file_0 == 0) N        '_1' not hit  Hit '_1'

        Rows:    Hits   FEC Target             Non-Masking Condition(s)
      --------   -----  ---------------------- -------------------------
      Row   1:       1  (data_file_0 == 0)_0 -
      Row   2:       0  (data_file_0 == 0)_1 -
```

Objects

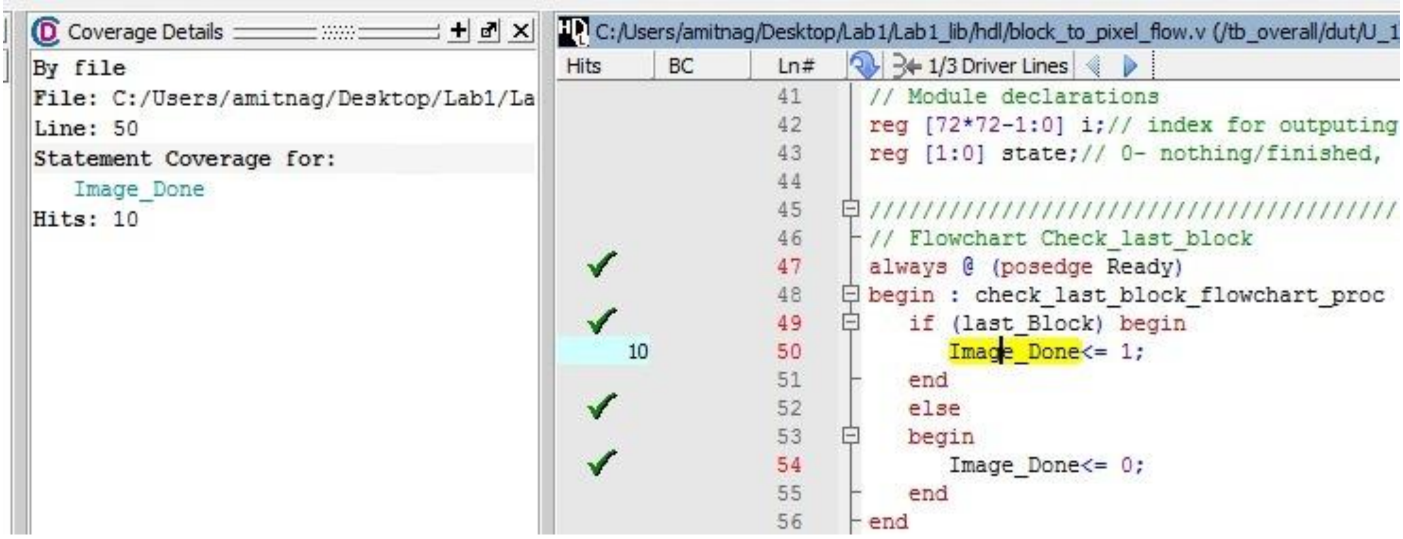
Name	Value	Kind	Mode
Amba_Addr_Depth	20	Parameter	Internal
Amba_Word	16	Parameter	Internal
Data_Depth	8	Parameter	Internal
data_file_0	-2147483633	Integer	Internal
data_file_1	-2147483632	Integer	Internal
data_file_2	-2147483631	Integer	Internal
scan_file_0	1	Integer	Internal
scan_file_1	1	Integer	Internal
scan_file_2	1	Integer	Internal
i	3	Integer	Internal
str0	C:/Users/a	String	Internal

C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/Stimulus.sv (/tb_overall/gen) - by file - Default

```
    // Initilization
    stim_bus.clk = 1; // start with clock and reset at '1', while enable at '0'
    stim_bus.rst = 1;
    stim_bus.PENABLE = 0;

    @(posedge stim_bus.clk); // wait til next rising edge (in other words, wait 20ns)
    stim_bus.rst = 0;

    // Starting work by reading the data from external files,
    // then sending it to the device by asserting the values to the appropriate ports.

    //// The parameters file
    //file_name = {str0, i, txt}; // Concatenation: combining number of elements together as one string
    val.itoa(i);
    //$display($sformatf({str0, val, ".txt"}));
    data_file_0 = $fopen($sformatf({str0, val, ".txt"}), "r"); // opening file in reading format
    if (data_file_0 == `NULL) begin // checking if we mangaed to open it
      $display("data_file_0 handle was NULL");
      $finish;
    end
    //// The Primary Image Pixels file
    //file_name = {str1, i, txt}; // Concatenation: combining number of elements together as one string
    val.itoa(i);
    data_file_1 = $fopen($sformatf({str1, val, ".txt"}), "r"); // opening file in reading format
    if (data_file_1 == `NULL) begin // checking if we mangaed to open it
      $display("data_file_1 handle was NULL");
      $finish;
    end
    //// The Watermark Image Pixels file
    //file_name = {str2, i, txt}; // Concatenation: combining number of elements together as one string
    val.itoa(i);
    data_file_2 = $fopen($sformatf({str2, val, ".txt"}), "r"); // opening file in reading format
    if (data_file_2 == `NULL) begin // checking if we mangaed to open it
      $display("data_file_2 handle was NULL");
```

Figure 12: Code Coverage Report ex.2

Example of ‘if’ statement that is always ‘false’ in the code, because these lines only activated when the opening of the input file was unsuccessful.



```
Coverage Details
By file
File: C:/Users/amitnag/Desktop/Lab1/La
Line: 50
Statement Coverage for:
   Image_Done
Hits: 10
```

C:/Users/amitnag/Desktop/Lab1/Lab1_lib/hdl/block_to_pixel_flow.v (/tb_overall/dut/U_1

```
// Module declarations
reg [72*72-1:0] i;// index for outputing
reg [1:0] state;// 0- nothing/finished,

////////////////////////////////////////
// Flowchart Check_last_block
always @ (posedge Ready)
begin : check_last_block_flowchart_proc
  if (last_Block) begin
    Image_Done<= 1;
  end
  else
  begin
    Image_Done<= 0;
  end
end
```

Figure 13: Code Coverage Report ex.3

All the other lines of code got covered. e.g., in fig.6, the number of hits of Image_Done rising to ‘1’ was 10 as the number of images tested.

5.4 Golden Model Comparison

The following table shows the result of our design compared to the golden model.

# Image	N	M	Mean Squared Error
1	344	2	0.444438
2	428	4	0.950148
3	228	1	0.484322
4	375	25	0.493724
5	559	1	0.277031
6	603	9	0.509160
7	270	10	0.476571
8	467	1	0.427995
9	222	2	0.506493
10	684	57	0.353748

Table 3: Mean Squared Errors

We chose to check the Mean Squared Error between the design and the golden model images.

$$MSE = \frac{1}{\#Image\_pixels} \sum_{i=1}^{\#Image\_pixels} (pixel_{DUT} - pixel_{Golden\ Model})^2$$

The MSEs received are below 1, which means that the result is very similar to the golden model result.

The critical part that defines the difference gap between the results is that we use precision of 10^{-3} in the calculations. Furthermore, our design calculates the expression $2^{-(\mu_k-0.5)^2}, 0 \le \mu_k \le \frac{255}{256}$ for eq.1-2 using constant quantization, that achieve fast calculations and hardware efficiency with minor error.

Human eye test using MATLAB plot of output txt files:

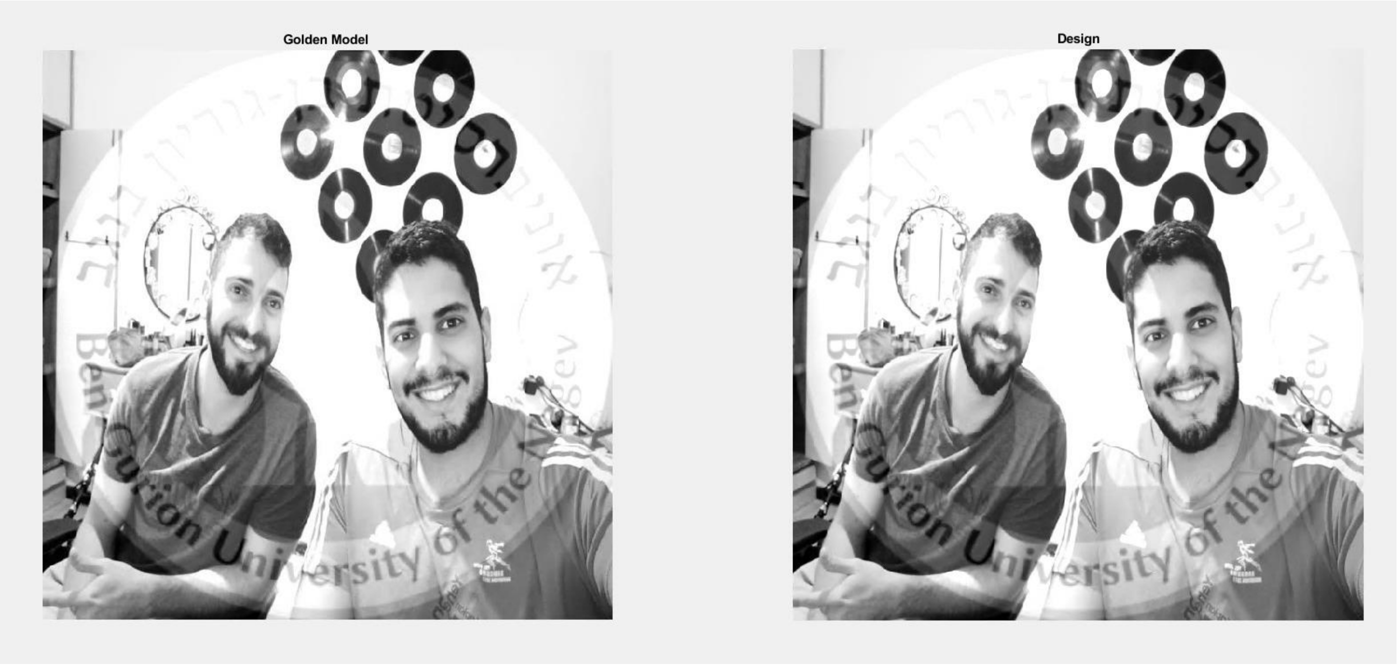


Figure 14: Design works on us (not in the MSE table).

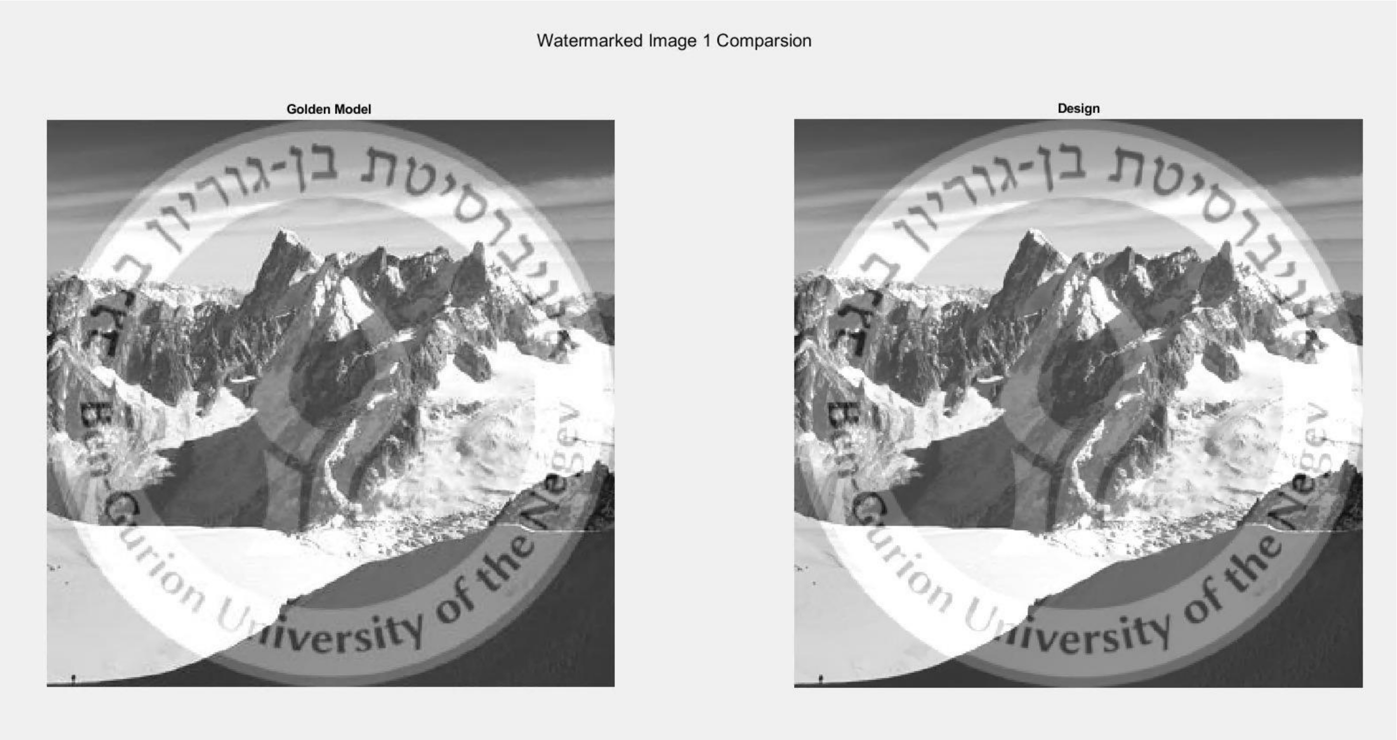


Figure 15: Watermarked Image 1 Comparison.

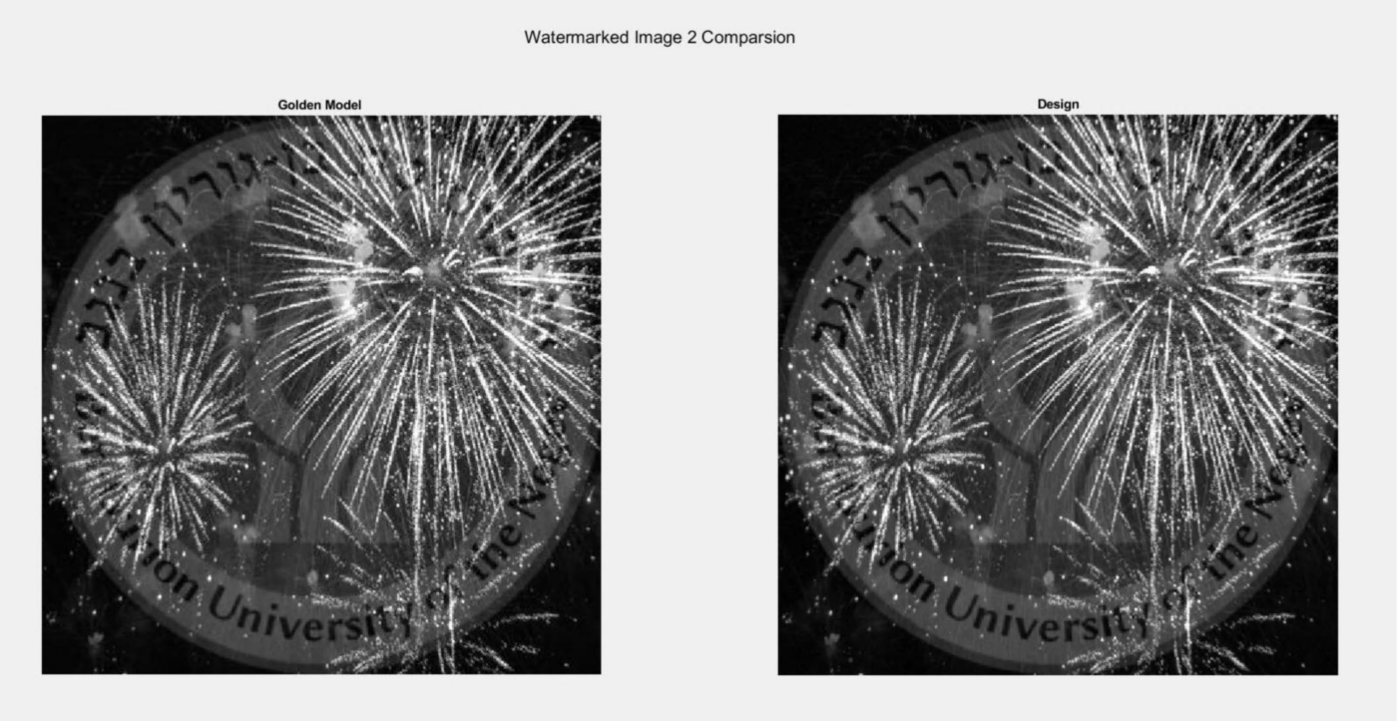


Figure 16: Watermarked Image 2 Comparison.

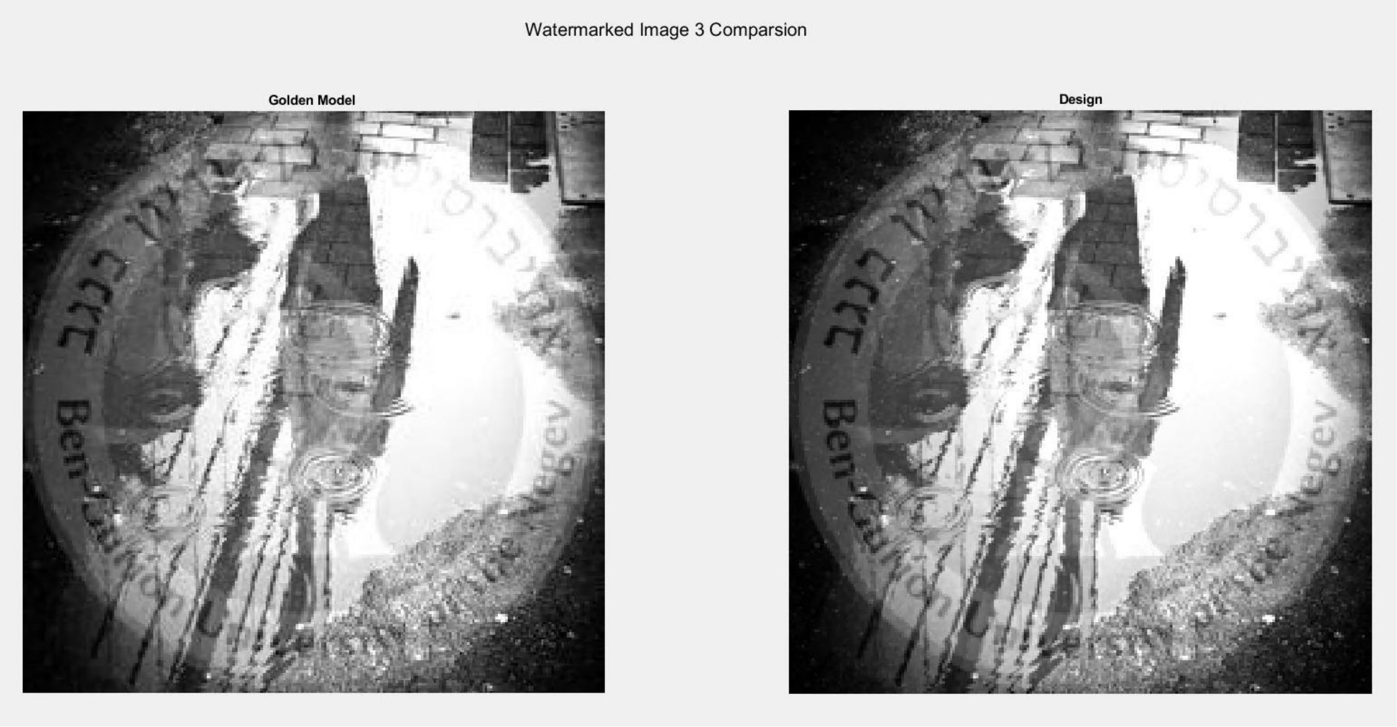


Figure 17: Watermarked Image 3 Comparison.

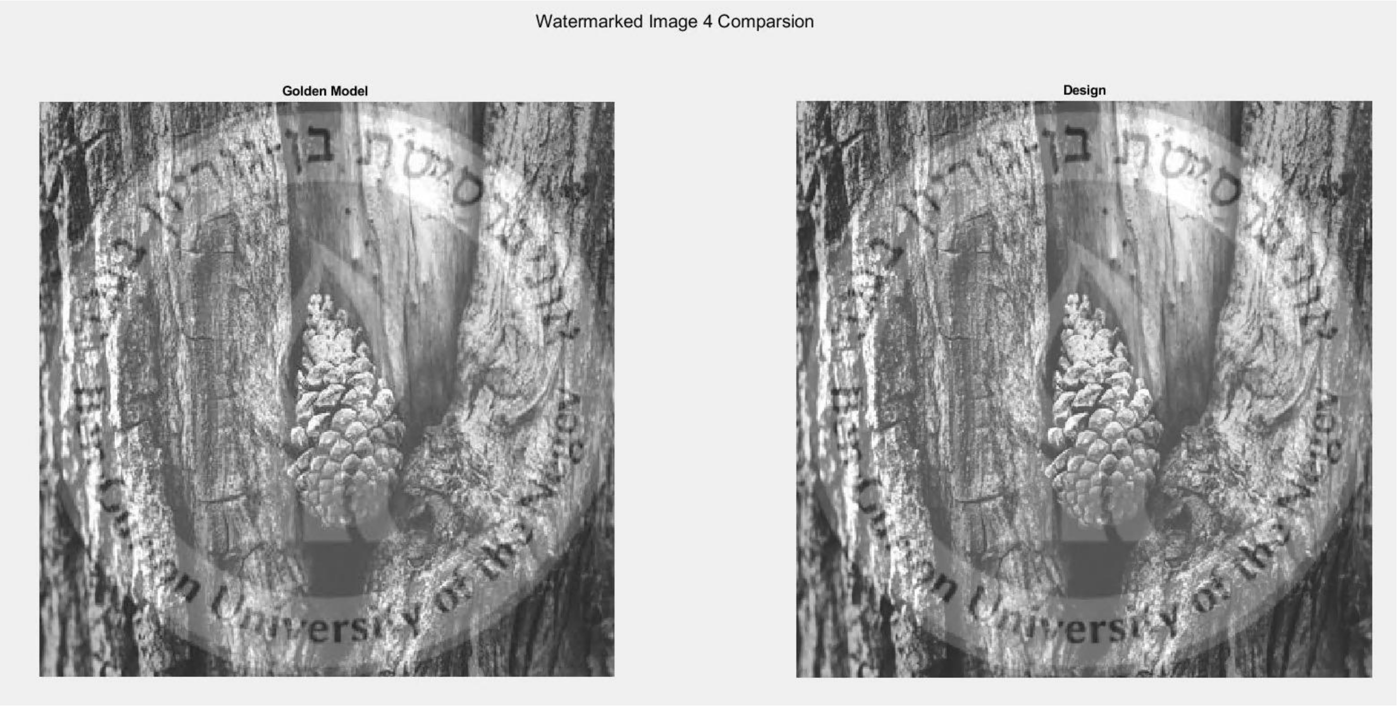


Figure 18: Watermarked Image 4 Comparison.

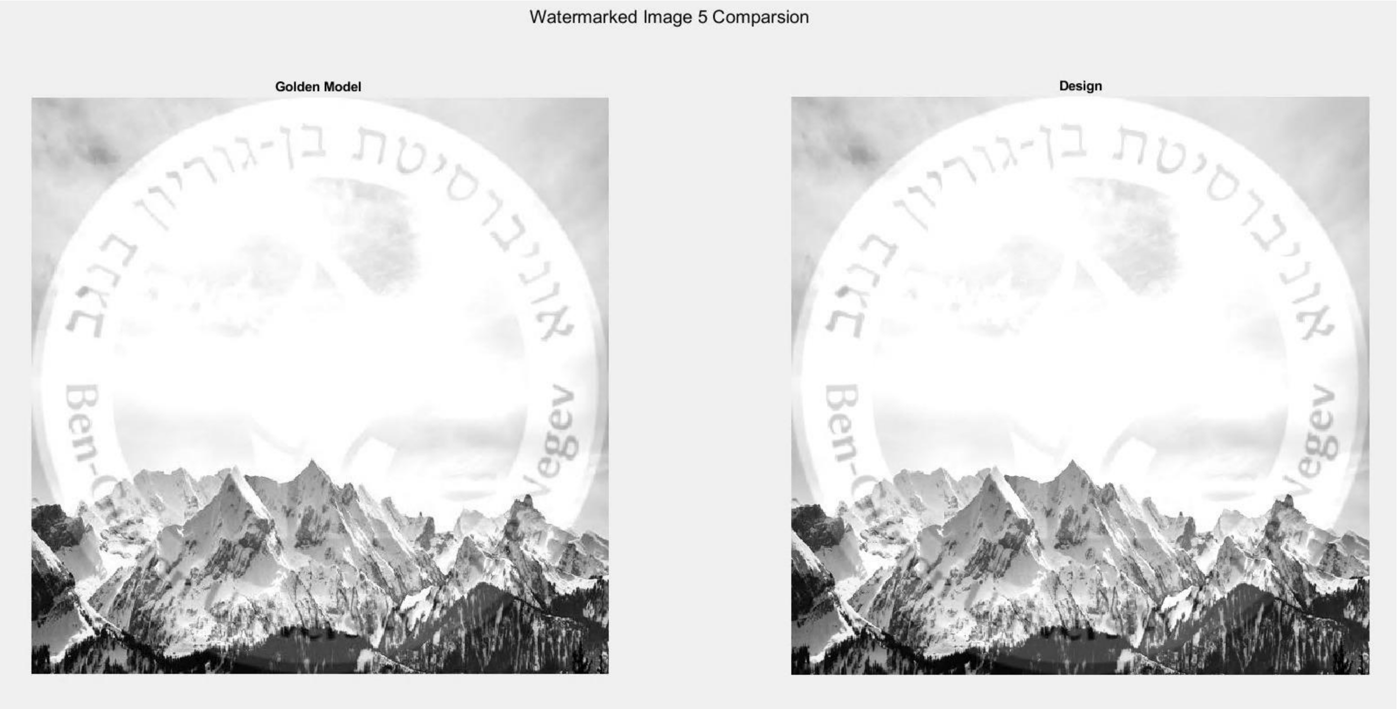


Figure 19: Watermarked Image 5 Comparison.

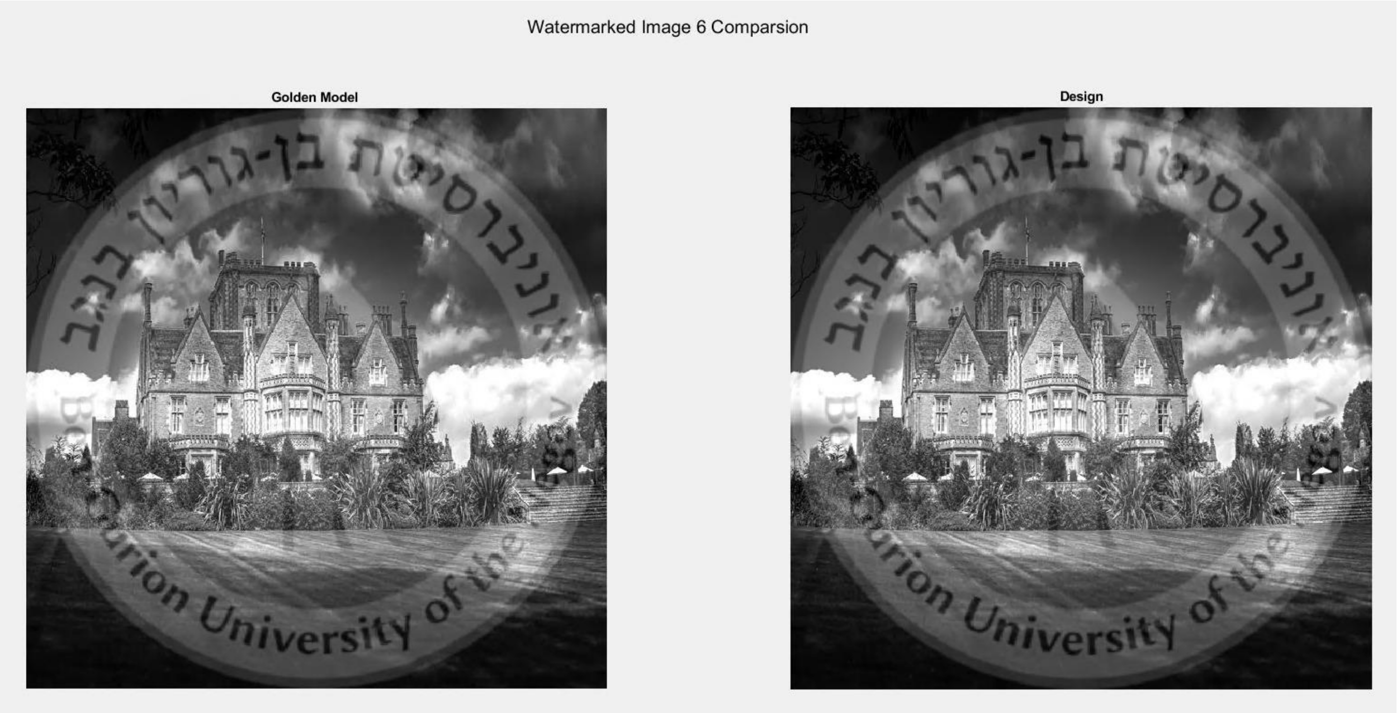


Figure 20: Watermarked Image 6 Comparison.

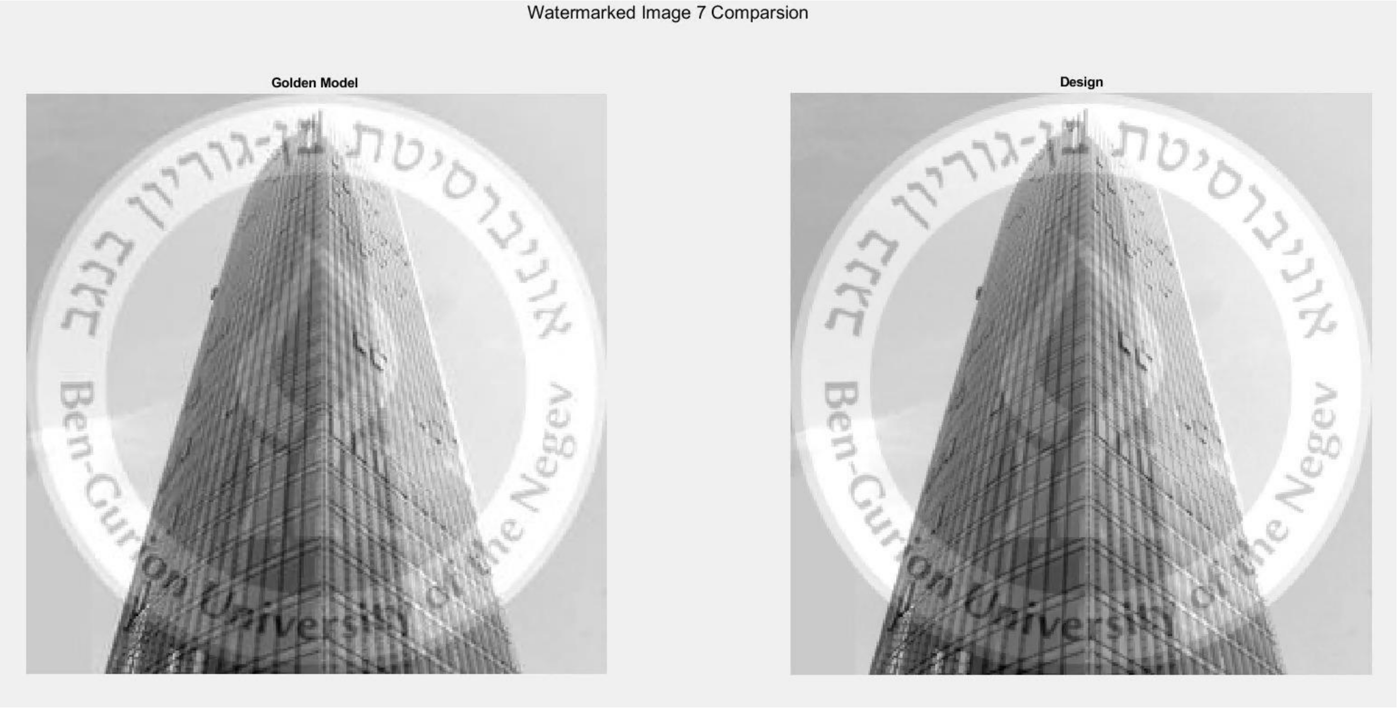


Figure 21: Watermarked Image 7 Comparison.

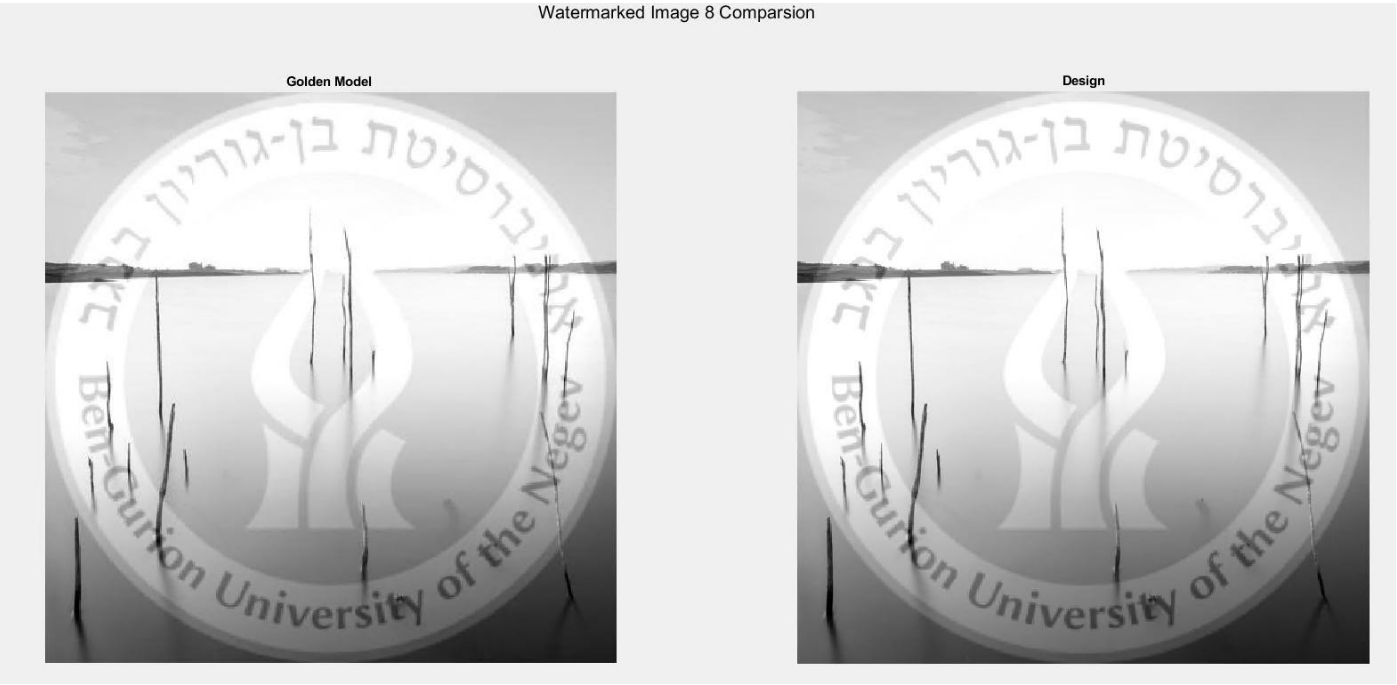


Figure 22: Watermarked Image 8 Comparison.

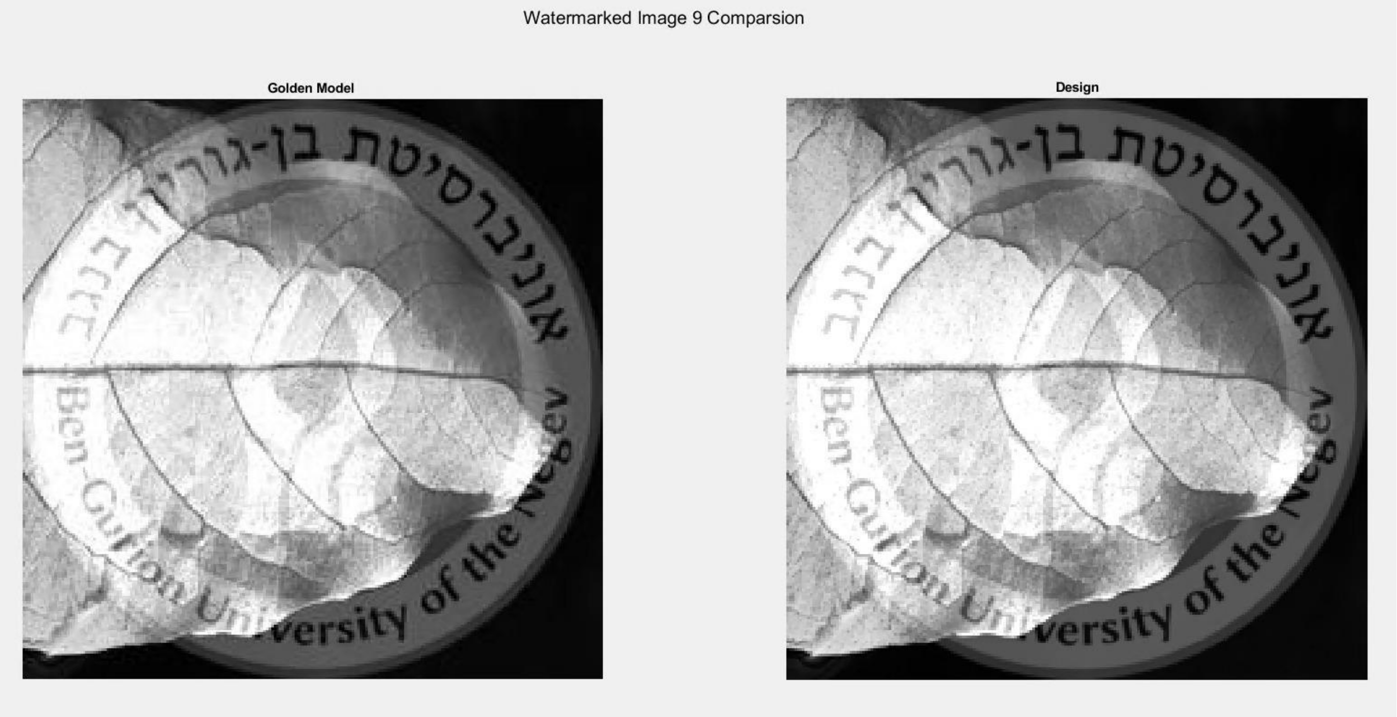


Figure 23: Watermarked Image 9 Comparison.

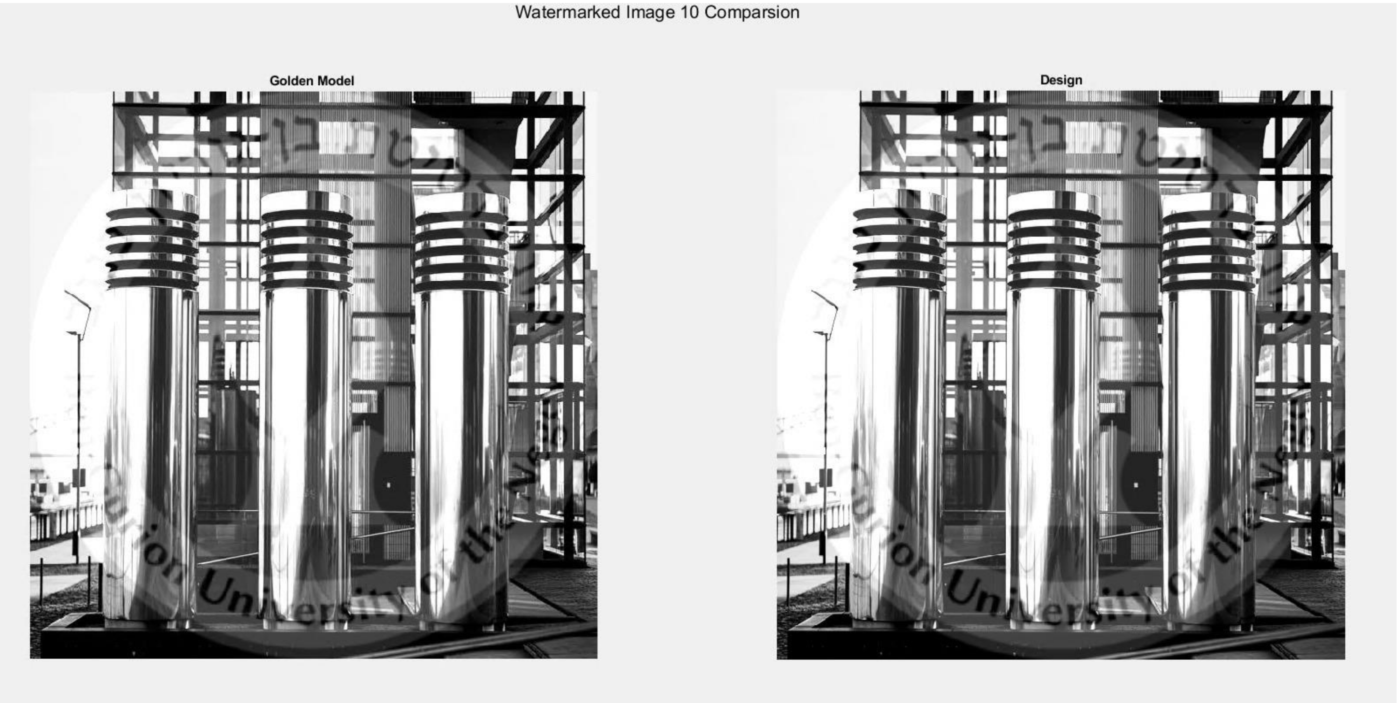


Figure 24: Watermarked Image 10 Comparison.

As one can see, the differences almost unnoticed for different images and parameters.

5.5 Formal Checker

After compiling each Verilog module, we got ‘0’ Warning.

6. APPENDIX

6.1 References

[1] Amba standard Moodle→Amba Specifications