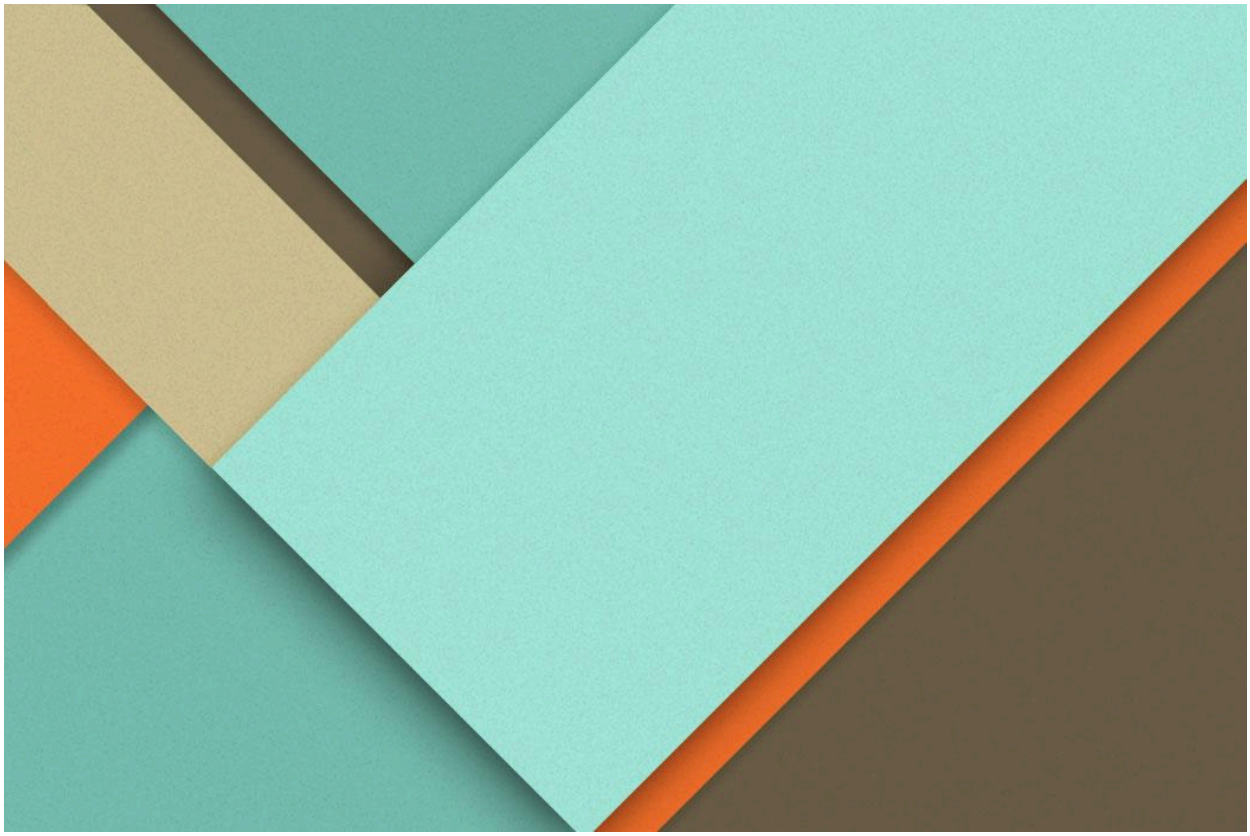


FPGA Project

08.11.2024

Integrating a simple application with MicroBlaze softcore processor

Aryan Mishra - IMT2022502
R Harshavardhan - IMT2022515
Ishan Jha - IMT2022562

Overview

The aim of the project is to develop a hardware-based solution for real-time image processing by implementing various image manipulation effects like grayscale conversion, brightness adjustment, contrast modification, and image flipping on an FPGA platform.

GitHub link(of code used):

https://github.com/Ishaniiitb/FPGA_Project

Cpp functions used:

1. **Grayscale:**

   → This function converts a color image to grayscale by calculating a weighted sum of the red, green, and blue components for each pixel based on the formula Y = 0.299 x R + 0.587 x G + 0.114 x B .

   → It updates each pixel's color channels to the computed grayscale value, ensuring a uniform grey tone across RGB channels for each pixel.

2. **Brighten:**

   → This function adjusts the brightness of an image by modifying each pixel's RGB values based on a specified amount parameter, which ranges from -1 to 1. A positive amount brightens the image, while a negative amount darkens it.

   → It calculates new RGB values by adding or subtracting from each channel, capping the results within the valid range (0–255) to prevent overflow or underflow.

3. **Contrast:**

   → This function adjusts the contrast of an image by scaling the difference of each pixel's RGB values from the midpoint (128) based on a specified amount, which ranges from 0 to 100.

   → The contrast factor is calculated, and each pixel's RGB values are modified accordingly, with values clamped between 0 and 255 to stay within valid color limits.

4. **Flip:**

   → This function flips an image either horizontally, vertically, or both based on the parameters flipHorizontal and flipVertical, which are set to 1 or 0 to indicate the desired flipping.

   → If flipHorizontal is 1, each row's pixels are swapped from left to right. If flipVertical is 1, the rows are swapped from top to bottom.

Problems faced:

1. **Implementing it using MicroBlaze softcore processor:**

   → In the beginning, the project sought to implement image processing using the MicroBlaze softcore processor. However, this method encountered some restrictions, especially in terms of managing image input through block RAM.

   → The MicroBlaze processor did not have enough ports to directly connect with block RAM for image data storage and retrieval, which hindered the efficiency of image processing.

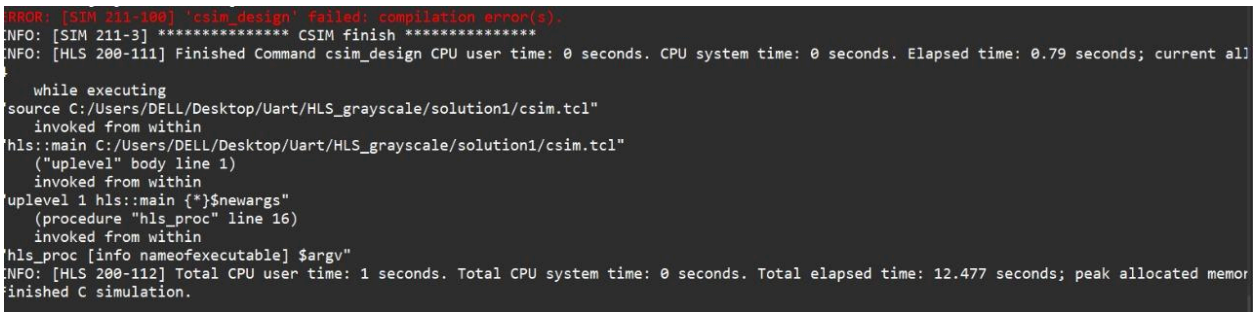
   → Due to this limitation, as suggested to us, we proceeded without using the processor.

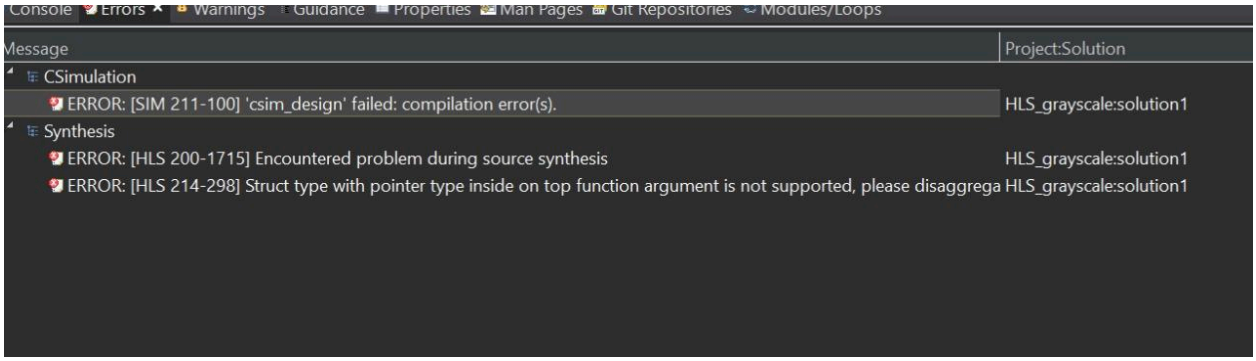
2. **Constraints of HLS and array handling:**

   → The original code was structured with pointers and 3D arrays to manage pixel data (RGB values). HLS, however, was unable to synthesize this effectively because it does not support pointers or 3D arrays.

   → Although we refactored the code to utilize vectors—thus enabling us to retain the image data—still it turns out that internally pointers are used for vector iteration, which results in comparable limitations.

   → This necessitated additional modifications to the data structures and functions in order to align with HLS's constraints.

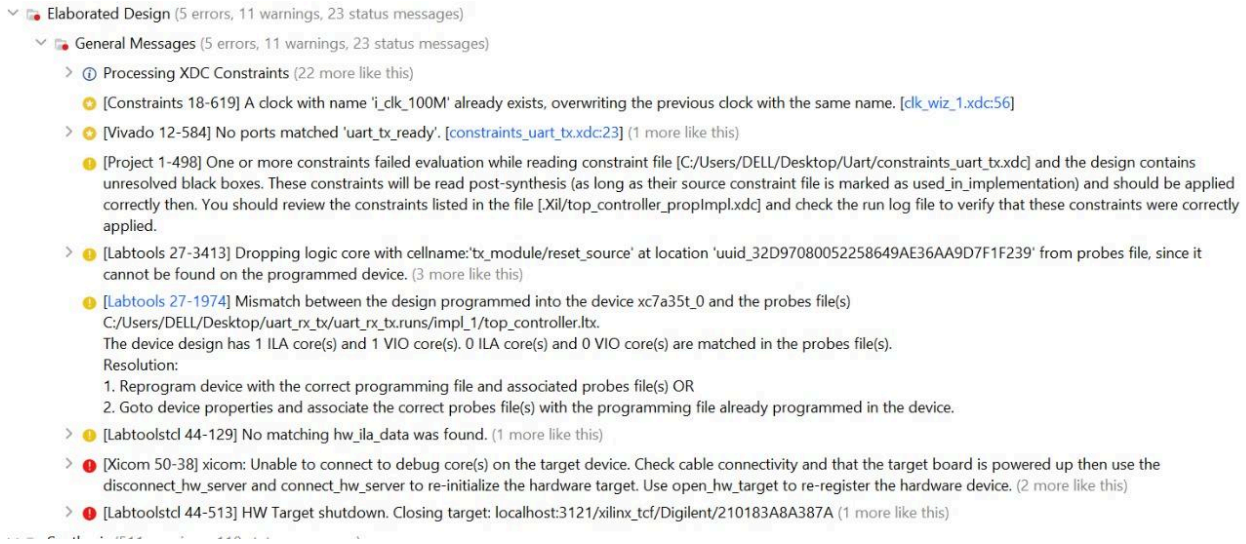


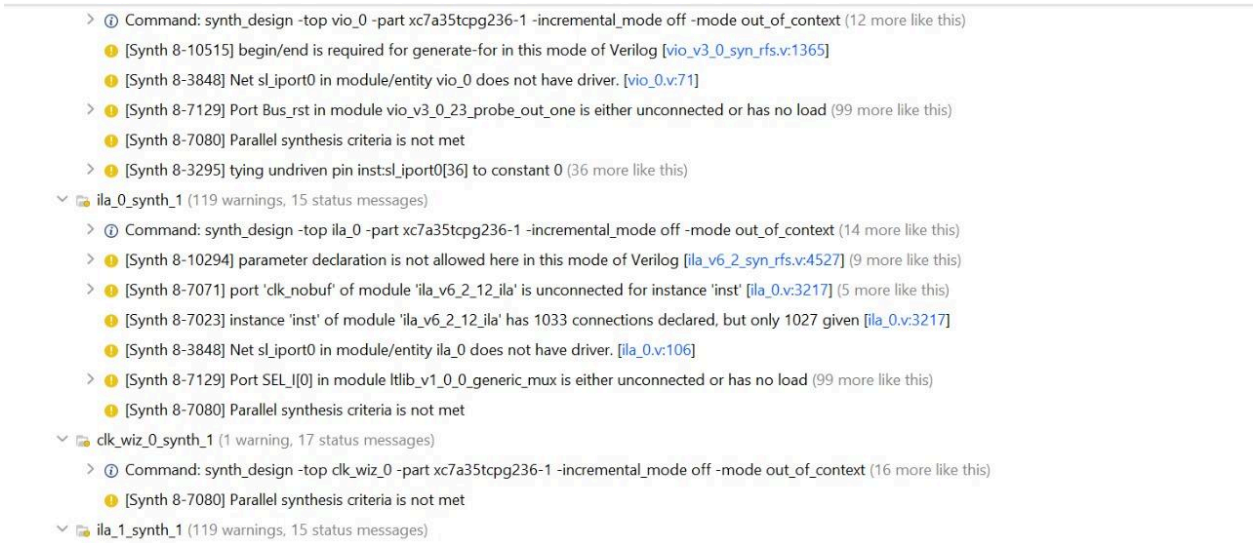


3. Editing IP to handle only few pixel values at a time

→ Tried editing cpp function to compute only a few pixel values. This would involve sending only predetermined pixel values to the fpga, have the IP perform operation and send it back to PC via UART.

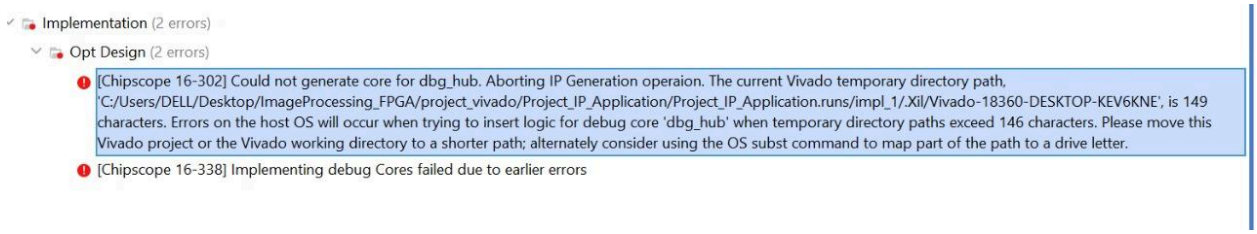
→ Grounding of ports of block ram initiated lead to improper storage of image/ pixel values.





4. **Error while sending image due to file path length limitation via uart:**

   The error is caused by the file path length limitation on our operating system (Windows). Vivado uses a temporary directory to generate and manage IP cores, and the full path to this directory is too long.



Methodology used:

1. The RGB values for each pixel are transmitted to the FPGA via UART, enabling the FPGA to receive image data in a structured format(1-D array).
2. The RGB values received are stored in a Block RAM of the FPGA.
3. Storing the values, it is inputted to a custom IP core created using HLS, applying one of the effects: grayscaling, brightness, contrast or flip on the image.
4. The modified pixel data is then the output from the IP core and it's stored in a dual-port RAM.
5. The modified array is then sent back to the PC via UART.

Implementation details:

```
================================================================
== HW Interfaces
================================================================
* AP_MEMORY
+--------------------+----------+
| Interface          | Bitwidth |
+--------------------+----------+
| image_r_address0   | 12       |
| image_r_address1   | 12       |
| image_r_d0         | 32       |
| image_r_d1         | 32       |
| image_r_q0         | 32       |
| image_r_q1         | 32       |
| tempData_address0  | 12       |
| tempData_q0        | 32       |
+--------------------+----------+

* Other Ports
+-----------+---------+-----------+
| Interface | Mode    | Bitwidth |
+-----------+---------+-----------+
| amount1   | ap_none | 32       |
| amount2   | ap_none | 32       |
| ap_return |         | 32       |
| brighten  | ap_none | 32       |
| flip      | ap_none | 32       |
| grayscale | ap_none | 32       |
| invert    | ap_none | 32       |
| sepia     | ap_none | 32       |
| sharpen   | ap_none | 32       |
+-----------+---------+-----------+

* TOP LEVEL CONTROL
+-----------+------------+-----------------------------------+
| Interface | Type       | Ports                             |
+-----------+------------+-----------------------------------+
| ap_clk    | clock      | ap_clk                            |
| ap_rst    | reset      | ap_rst                            |
| ap_ctrl   | ap_ctrl_hs | ap_done ap_idle ap_ready ap_start |
+-----------+------------+-----------------------------------+
```

```
================================================================
== SW I/O Information
================================================================
* Top Function Arguments
+-----------+-----------+----------+
| Argument  | Direction | Datatype |
+-----------+-----------+----------+
| tempData  | in        | int*     |
| image     | inout     | int*     |
| grayscale | in        | int      |
| brighten  | in        | int      |
| flip      | in        | int      |
| invert    | in        | int      |
| sepia     | in        | int      |
| sharpen   | in        | int      |
| amount1   | in        | float    |
| amount2   | in        | float    |
| return    | out       | int      |
+-----------+-----------+----------+

* SW-to-HW Mapping
+-----------+-------------------+---------+----------+
| Argument  | HW Interface      | HW Type | HW Usage |
+-----------+-------------------+---------+----------+
| tempData  | tempData_address0 | port    | offset   |
| tempData  | tempData_ce0      | port    |          |
| tempData  | tempData_q0       | port    |          |
| image     | image_r_address0  | port    | offset   |
| image     | image_r_ce0       | port    |          |
| image     | image_r_we0       | port    |          |
| image     | image_r_d0        | port    |          |
| image     | image_r_q0        | port    |          |
| image     | image_r_address1  | port    | offset   |
| image     | image_r_ce1       | port    |          |
| image     | image_r_we1       | port    |          |
| image     | image_r_d1        | port    |          |
| image     | image_r_q1        | port    |          |
| grayscale | grayscale         | port    |          |
| brighten  | brighten          | port    |          |
| flip      | flip              | port    |          |
| invert    | invert            | port    |          |
| sepia     | sepia             | port    |          |
| sharpen   | sharpen           | port    |          |
| amount1   | amount1           | port    |          |
| amount2   | amount2           | port    |          |
| return    | ap_return         | port    |          |
+-----------+-------------------+---------+----------+
```

Results:

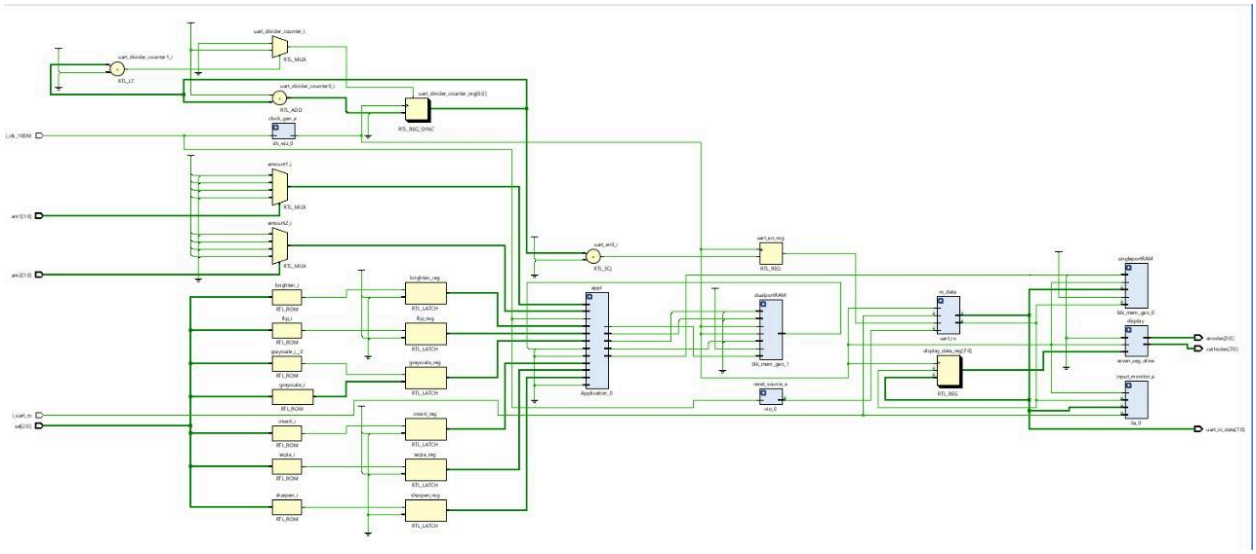
1. **Max. clock frequency:** 137.23 MHz(1/7.287ns)
2. **Resource Utilization:**

+ Performance & Resource Estimates:

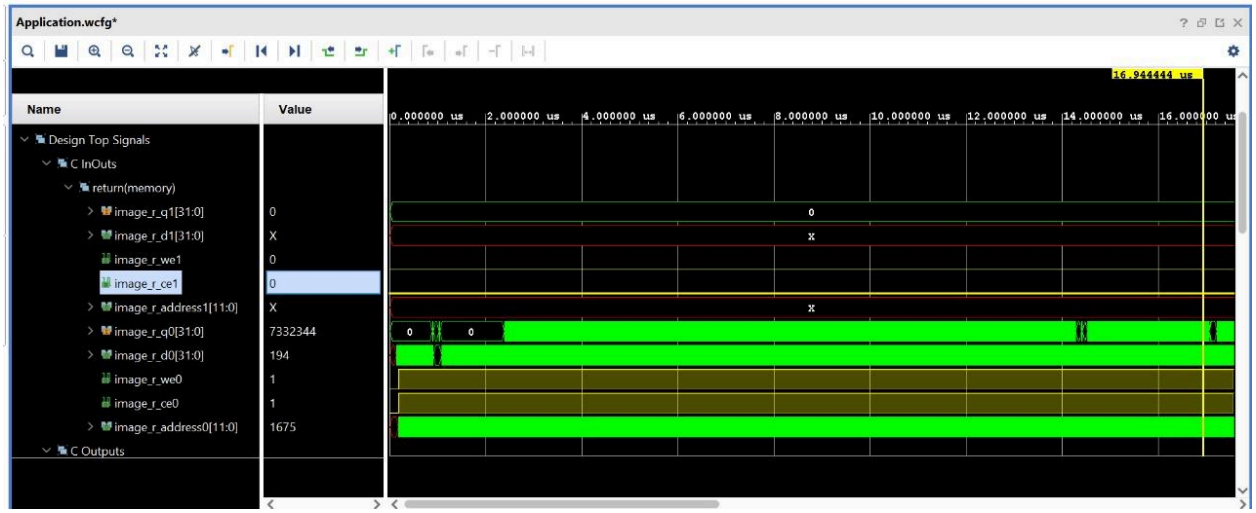
PS: '+' for module; 'o' for loop; '*' for dataflow

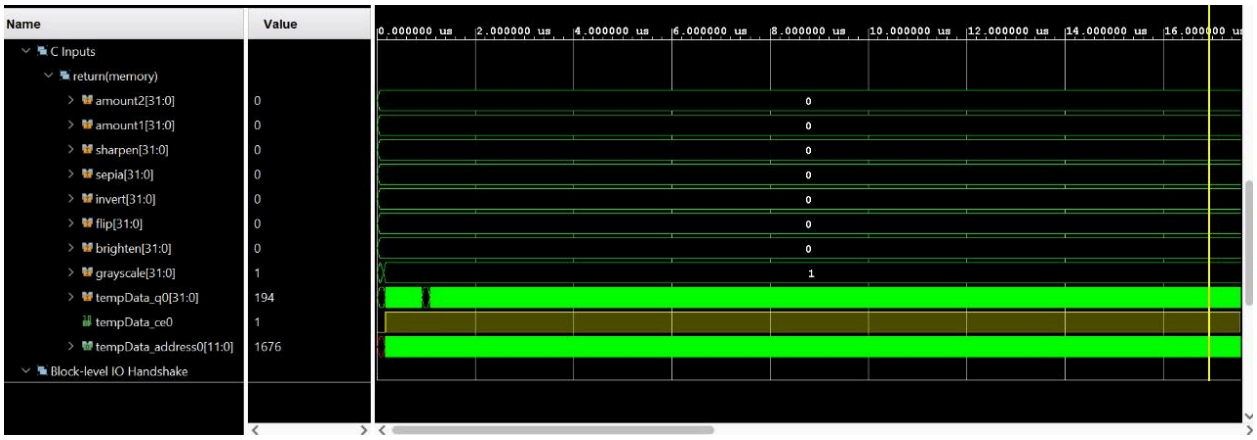
Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ Application	-	0.01	-	-	-	-	-	no	-	46 (51%)	9482 (22%)	14489 (69%)	-
+ Application_Pipeline_VITIS_LOOP_61_1	-	0.79	2074	2.074e+04	-	2074	-	no	-	1 (1%)	164 (~0%)	256 (1%)	-
o VITIS_LOOP_61_1	-	7.30	2072	2.072e+04	4	1	2070	yes	-	-	-	-	-
+ p_hls_fptosi_float_i32	II	0.01	1	10.000	-	1	-	yes	-	-	66 (~0%)	640 (3%)	-
+ p_hls_fptosi_float_i32	II	0.01	1	10.000	-	1	-	yes	-	-	66 (~0%)	640 (3%)	-
+ Application_Pipeline_VITIS_LOOP_85_2_VITIS_LOOP_87_3	-	0.21	2106	2.106e+04	-	2106	-	no	-	-	1133 (2%)	1806 (8%)	-
o VITIS_LOOP_85_2_VITIS_LOOP_87_3	II	7.30	2104	2.104e+04	38	3	690	yes	-	-	-	-	-
+ Application_Pipeline_VITIS_LOOP_100_4_VITIS_LOOP_102_5	-	0.01	2091	2.091e+04	-	2091	-	no	-	-	790 (1%)	928 (4%)	-
o VITIS_LOOP_100_4_VITIS_LOOP_102_5	II	7.30	2089	2.089e+04	23	3	690	yes	-	-	-	-	-
+ p_hls_fptosi_float_i32	II	0.01	1	10.000	-	1	-	yes	-	-	66 (~0%)	640 (3%)	-
+ Application_Pipeline_VITIS_LOOP_184_10_VITIS_LOOP_186_11	-	0.26	2075	2.075e+04	-	2075	-	no	-	-	184 (~0%)	451 (2%)	-
o VITIS_LOOP_184_10_VITIS_LOOP_186_11	II	7.30	2073	2.073e+04	7	3	690	yes	-	-	-	-	-
+ Application_Pipeline_VITIS_LOOP_197_12_VITIS_LOOP_199_13	-	0.21	2175	2.175e+04	-	2175	-	no	-	25 (27%)	3891 (9%)	5840 (28%)	-
o VITIS_LOOP_197_12_VITIS_LOOP_199_13	II	7.30	2173	2.173e+04	107	3	690	yes	-	-	-	-	-
+ Application_Pipeline_VITIS_LOOP_216_14_VITIS_LOOP_217_15	-	0.01	2094	2.094e+04	-	2094	-	no	-	-	859 (2%)	879 (4%)	-
o VITIS_LOOP_216_14_VITIS_LOOP_217_15	II	7.30	2092	2.092e+04	26	3	690	yes	-	-	-	-	-
+ p_hls_fptosi_float_i32	II	0.01	1	10.000	-	1	-	yes	-	-	66 (~0%)	640 (3%)	-
o VITIS_LOOP_141_6	-	7.30	-	-	-	-	-	no	-	-	-	-	-
+ Application_Pipeline_VITIS_LOOP_145_7	-	0.18	-	-	-	-	-	no	-	-	426 (1%)	516 (2%)	-
o VITIS_LOOP_145_7	II	7.30	-	-	7	6	-	yes	-	-	-	-	-
o VITIS_LOOP_166_8	-	7.30	-	-	-	-	-	no	-	-	-	-	-
+ Application_Pipeline_VITIS_LOOP_170_9	-	0.79	-	-	-	-	-	no	-	1 (1%)	162 (~0%)	223 (1%)	-
o VITIS_LOOP_170_9	II	7.30	-	-	6	2	-	yes	-	-	-	-	-

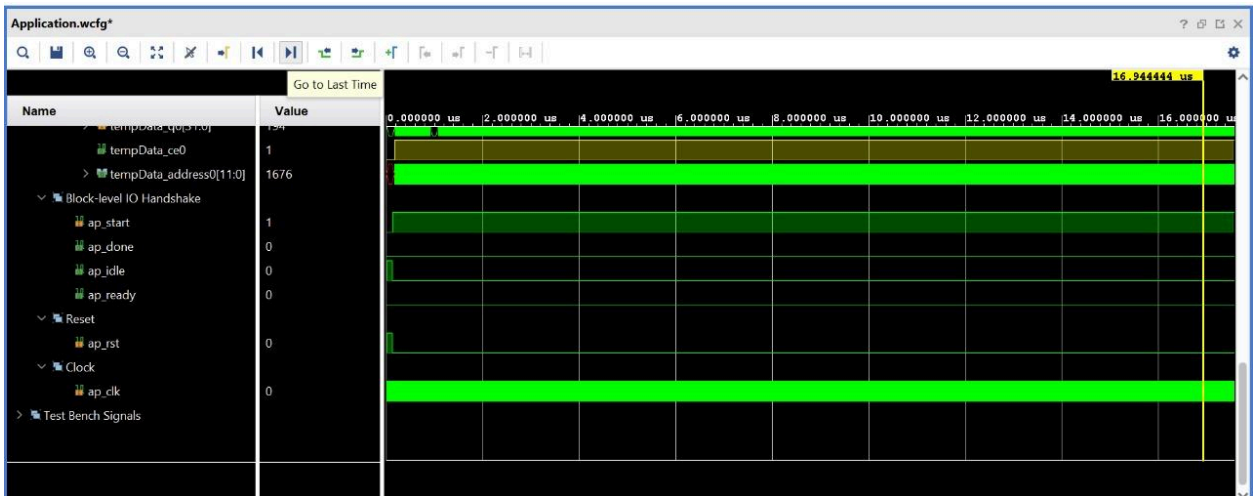
3. **Implemented Layout diagram:**



4. Simulation waveforms:







We can see that out of all the effects, only the value of grayscale is 1 because only that effect has been used for this particular example.