Project Objective

This project covers three main objectives:

1. Executing the image rotation
2. Measuring the system performance
3. Improving the system performance

Rotation Requirements

a. Use the existing platform designer that you used in Project 4.

b. In your application project, you may comment or delete the previous functions (display, scaling and message printing) that were used in Project 4. In this project, our focus will be only on the image rotation code.

c. No different modes will be run (as previous projects), only rotation.

d. Rotation code is provided in slides (17-21) of L11. You are required to rotate the image for four times (0, 90, 180 and 270 degrees).

e. For testing, you can compare your project operation with the demo-video attached in Module 6.

Performance Requirements

a. In the platform designer, add **the interval timer and** **set its period to 100 and units to µs**, and keep any other settings as default. Connect the component with the processor and clock sources as you connected the JTAG UART (including the interrupt IRQ signal).

b. **Set the priority of the timer component as 0 and the priority of the JTAG UART as 16** as explained in the lecture.

c. Add the **system ID peripheral** and keep its features as default. Connect the component to the processor and the clock sources as you connect the JTAG UART.

d. Then, re-generate the system, recompile quartus project, and configure the board.

e. Open the application project in eclipse. Right-click on the BSP directory, go to Nios II and select Nios II editor. Set the following features, then regenerate the BSP.

Sys_clk_timer	Timestamp_timer	stdin	stdout	stderr
Timer_0	none	jtag	jtag	jtag

f. In the application directory, open the main (or hello_world.c) program. Add the following header

```
#include <sys/alt_alarm.h>
```

g. Measure the time (number of ticks) required to execute the frame time. To do that you will need to use the code below. Also, use ticks_example posted in Module 6 as your reference.

```
int ticks_per_second, ticks_start, ticks_end, frame_ticks;
unsigned long long duration;

ticks_per_second = alt_ticks_per_second();

ticks_start = alt_nticks();


frame_function();

ticks_end = alt_nticks();

frame_ticks = ticks_end - ticks_start;

duration= (unsigned long long) ticks_total / (unsigned long long) ticks_per_second;
printf("Total duration %llu seconds \n\n", duration);
```

Improve-Performance Requirements

a. In the platform designer, **remove the timer component** and **add the performance counter**. Keep its features as default. Connect the counter, as you connected the JTAG UART.

b. Add the **custom floating-point unit (floating-point hardware2)** and connect its slaves (s1 and s2) to the processor custom_instruction_master.

c. Then, re-generate the system, recompile quartus project, and configure the board.

d. Open the application project in eclipse. Right-click on the BSP directory, go to Nios II and select Nios II editor. Reset features, then regenerate the BSP.

Sys_clk_timer	Timestamp_timer	stdin	stdout	stderr
none	none	jtag	jtag	jtag

e. In the application directory, open the main (or hello_world.c) program. Add the header

```
#include <altera_avalon_performance_counter.h>
```

f. Don't forget to remove `#include <sys/alt_alarm.h>` from your code.

g. Measure the cycles required to execute the pixel code by using the following code. Also, use example_cycles posted in Mosule 6 as your reference.

```
unsigned long long start_cycles, end_cycles, total_cycles;
start_cycles=perf_get_total_time((void*) PERFORMANCE_COUNTER_0_BASE);
PERF_START_MEASURING(PERFORMANCE_COUNTER_0_BASE);

pixel_code();

end_cycles=perf_get_total_time  ((void*) PERFORMANCE_COUNTER_0_BASE);
PERF_START_MEASURING(PERFORMANCE_COUNTER_0_BASE);
total_cycles = end_cycles - start_cycles;
printf("Estimate performance cycle = %llu \n", total_cycles);
```

Project Report (90%)

The project report will be graded out of 100, and the points will be distributed as following:

a. **Professional preparation** (10 points):

You are required to submit a typed document with text of the paragraphs in Times New Roman 11 pt font, clear and grammatically well-formed explanations, page numbering and document heading numbering (1.0, 2.0, 3.0, etc to identify the required sections listed below).

b. **Report Content** (90 points):

1.0 (*20 points total, each value is 5 points*) After you compiled and synthesized your system (with the timer), read the summary report from Quartus, and fill out the below table with the numbers from the report.

Logical Elements	Registers	Total Pins	Memory Bits

2.0 (*20 points total, each value is 5 points*) After you compiled and synthesized your system (without the timer, with the performance counter and floating-point custom), read the summary report from Quartus, and fill out the below table with the numbers from the report.

Logical Elements	Registers	Total Pins	Memory Bits

3.0 (*15 points*) Briefly, compare the results of Table.1 and Table.2.

4.0 (*10 points*) Run the application project on case 1 (rotation) and include a picture that shows at least one rotation (either 90, 180 or 270 degree).

5.0 (*5 points*) Run the application project on case 2 and calculate the frame rate in seconds and put the value in the table below. Note, $frame\ rate\ =\ frame\ time\ *\ number\ of\ frames$.

7.0 (*5 points*) Run the application project on case 3 and calculate the estimate number of cycles required to execute pixel code. The code will print the number of cycles in each pixel, you may select only one of them to add in the table below.

8.0 (*5 points*) Calculate number of instructions required to execute the pixel code by using the gdb and put the number in the table.

9.0 (*10 points, 5 for each number*) You are required to use the results that you got in 1st, 2nd, and 3rd columns to calculate the values for the 4th and 5th columns in the table. Please use the information provided regarding the system performance in L12.

Frame rate in seconds	#Cycles per pixel	#Inctructions per pixel	Calculate CPI for pixel	Calculate CPUexecution time

Project Demo (20%)

- The main purpose of the demo is to test your project functionality and execution.
- Demos will be checked and graded by the TA
- Demos will be graded out of 100, but worth 40% of total project grade
- Both partners must show up in that day. If a member didn't show up, he/she receives 0 unless an excused absence was provided.
- Demos will be conducted during the lab time on the following dates:
  - **Section 001**: Wed. April 6th or Wed. April 13th
  - **Section 002**: Fri April 8th or Fri. April 15th
  - Demo dates will be decided by the groups
- Below are how the demo points will be distributed

Tasks	Point
Rotation by 0	/25
Rotation by 90	/25
Rotation by 180	/25
Rotation by 270	/25

Project Submission

1. Save the project report as **r5_username1_username2.pdf**, username of both students in the group.
2. For this project, you are required to submit only the project report (No project submission is required). Submission date is Sunday April 10th by midnight.
3. **Only one attempt** is allowed
4. **Only one group member** can submit the report
5. **Remember:** Any grade dispute must be raised within one week of the grade posting.