Case B.

a. Remove the interval timer from your nios_system and add performance counter instead. The component connection is like peripheral connections such as the JTAG connections.

b. Re-generate the system, compile, and synthesize your project. Read the summary report from Quartus and fill out the below table. Save the Table in the report file.

Table B: Custom_Floating_point_performance_counter

Logical Elements	Registers	Total Pins	Memory Bits

c. Open the application project and reset the variables in BSP editor as shown below.

Sys_clk_timer	Timestamp_timer	stdin	stdout	stderr
none	none	jtag	Jtag	jtag

d. In the application directory, open the main (or hello_world.c) program. Add the following header to your application code.

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

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

f. Measure the average clock cycles per instruction (CPI) that required to execute the pixel code by using the following code.

```
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);
```

g. Follow the demo-performance cycles videos that are posted in Module 6 to measure the instruction counts.

h. Record the instruction count and the CPI.