



ELEC373 ASSIGNMENT 4

Nios II

Author:
Minghong Xu (201601082)

Assessor:
Prof Jeremy Smith

Declaration of academic integrity

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

18th May 2023

Contents

1	Part A	3
2	Part B	5

List of Figures

1	Block diagram of the Nios II system	3
2	Nios II system contents	3
3	SRAM and SDRAM test passed	4
4	Partial view of the hardware implementation of the clz instruction	5
5	Rough schematic of the hardware implementation of the clz instruction .	5
6	Functional simulation results of the clz instruction	5
7	Timing simulation results of the clz instruction	6
8	Slowest data delay of the system	6
9	Maximum frequency of the system	6
10	O3 enabled for fair comparison	7
11	The hardware implementation is faster than the software implementation	8

1 Part A

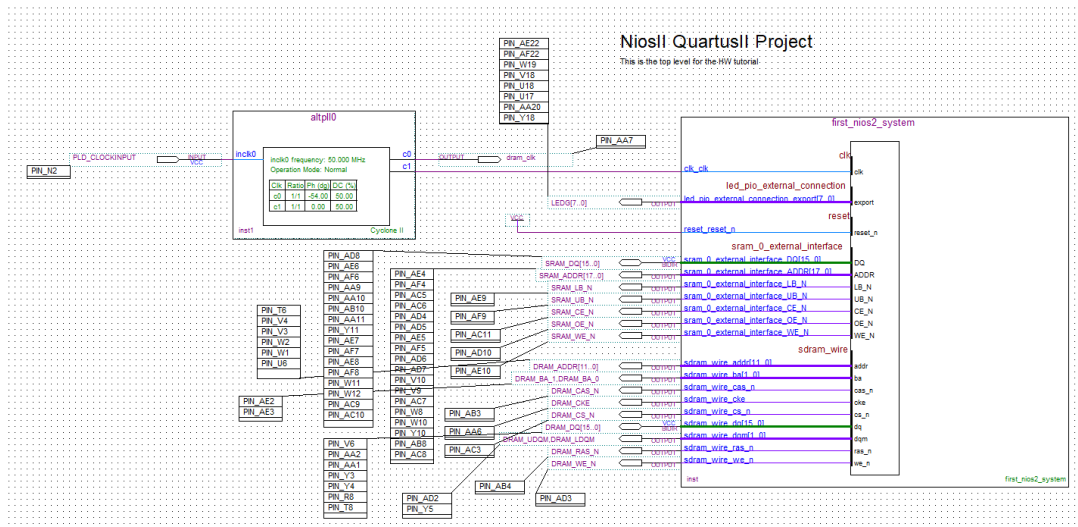


Figure 1: Block diagram of the Nios II system

Qsys - First_nios2_system.qsys (M1:ELEC373.assignment4-Funioil_hw_dev_tutorialfirst_nios2_system.qsys)

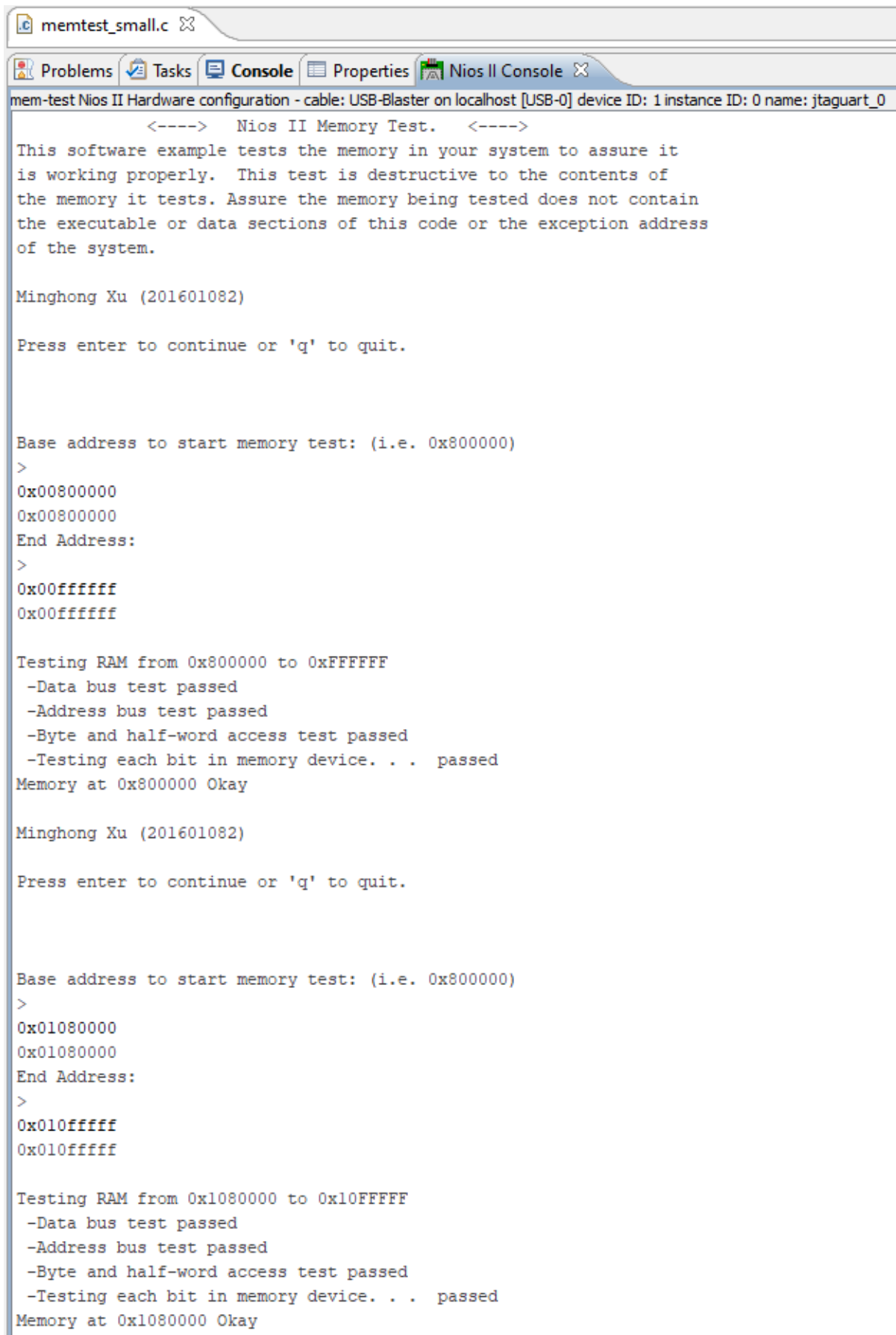
File Edit System View Tools Help

Component Library

Figure 2: Nios II system contents

Figure 1 is the block diagram of the Nios II system. The system has a static RAM and dynamic RAM, as shown in Figure 2. The static RAM was mounted to the FPGA's address from 0x0108000 to 0x010FFFFF. The dynamic RAM was mounted to the address from 0x00800000 to 0x00FFFFFFF. The dynamic RAM's clock leads the CPU clock by 3 nanoseconds to meet the timing requirements by using the phase-locked loop (PLL).

To verify that both memories are working properly, the sample memory test code was run. The results are shown in Figure 3.



The screenshot shows a Nios II console window with the title bar 'memtest_small.c'. The console output displays the 'Nios II Memory Test' results. It starts with a warning that the test is destructive. The user 'Minghong Xu (201601082)' is prompted to press enter to continue. The first test is for SRAM, starting at address 0x800000 and ending at 0x00ffffff. The test results show that the data bus, address bus, and byte/half-word access tests all passed, and the memory at 0x800000 is okay. The second test is for SDRAM, starting at address 0x01080000 and ending at 0x010ffffff. Similar to the SRAM test, the SDRAM tests also passed, and the memory at 0x01080000 is okay.

```
mem-test Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0

<---->  Nios II Memory Test.  <---->
This software example tests the memory in your system to assure it
is working properly.  This test is destructive to the contents of
the memory it tests.  Assure the memory being tested does not contain
the executable or data sections of this code or the exception address
of the system.

Minghong Xu (201601082)

Press enter to continue or 'q' to quit.

Base address to start memory test: (i.e. 0x800000)
>
0x00800000
0x00800000
End Address:
>
0x00ffffff
0x00ffffff

Testing RAM from 0x800000 to 0xffffffff
-Data bus test passed
-Address bus test passed
-Byte and half-word access test passed
-Testing each bit in memory device. . .  passed
Memory at 0x800000 Okay

Minghong Xu (201601082)

Press enter to continue or 'q' to quit.

Base address to start memory test: (i.e. 0x800000)
>
0x01080000
0x01080000
End Address:
>
0x010ffffff
0x010ffffff

Testing RAM from 0x1080000 to 0x10ffffff
-Data bus test passed
-Address bus test passed
-Byte and half-word access test passed
-Testing each bit in memory device. . .  passed
Memory at 0x1080000 Okay
```

Figure 3: SRAM and SDRAM test passed

2 Part B ASM for this? verilog for this?

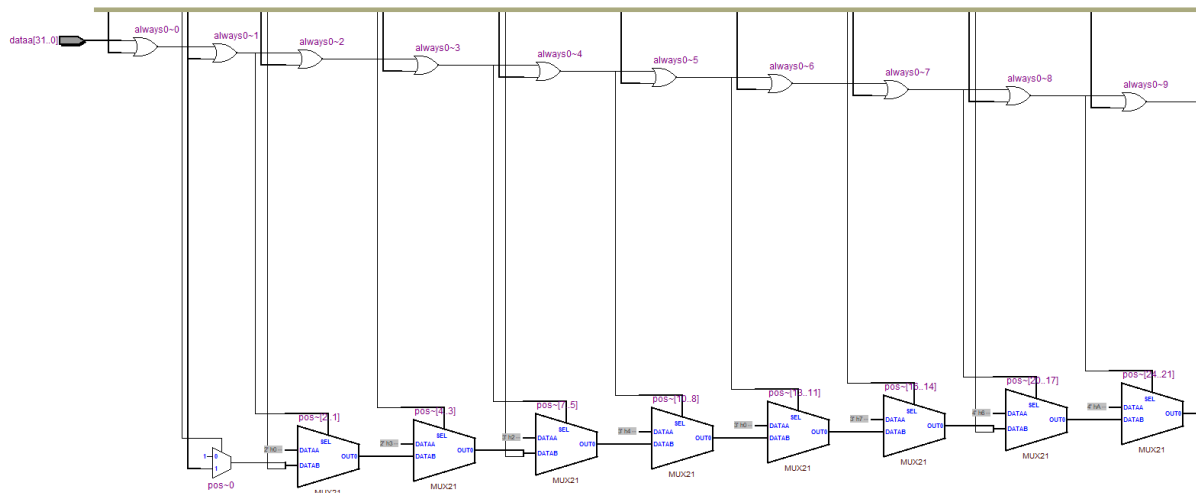


Figure 4: Partial view of the hardware implementation of the clz instruction

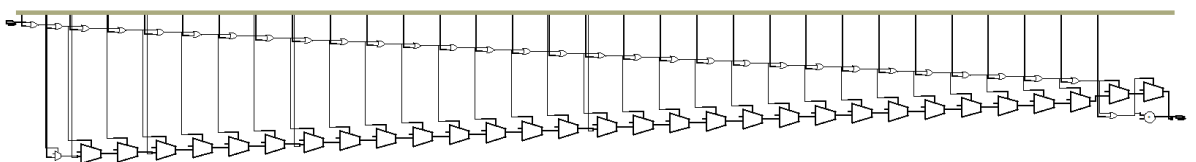
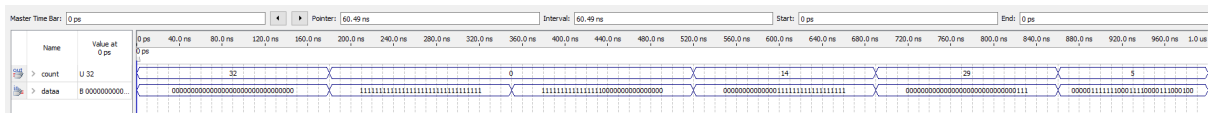


Figure 5: Rough schematic of the hardware implementation of the clz instruction



Simulation too small to read. What values were tested?

Figure 6: Functional simulation results of the clz instruction

Figure 4 and 5 are the schematic of the hardware implementation of the count leading zero (clz) instruction. The implementation is a series of multiplexers and OR gates. Figure 6 and 7 are the simulation test results of the implementation, which shows it is functioning correctly.

After adding the custom instruction, clz, to the system, the slowest data delay is 12 nanoseconds. The maximum frequency the system could reach is 82 MHz, which satisfies the requirement, 50 MHz.

For fair comparison between the hardware implementation and software implementation, the optimisation level was set to O3 for the best performance, as shown in Figure 10. The empirical result, as demonstrated in Figure 11, shows that the hardware implementation of the count leading zero is much faster than the software implementation which is a builtin function of GCC.

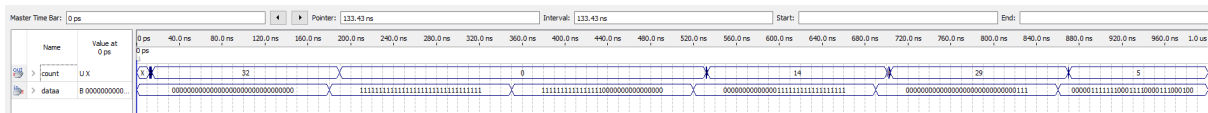


Figure 7: Timing simulation results of the clz instruction

Compilation Report - nios2_quartus2_project									
Slow Model Setup: 'sopc_clk'									
Table of Contents	Stack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay	
Flow Summary	1 7.866	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.010	12.178	
Flow Settings	2 7.913	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.010	12.133	
Flow Non-Default Global Settings	3 7.922	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.010	12.124	
Flow Elapsed Time	4 7.931	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.006	12.111	
Flow OS Summary	5 7.967	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.010	12.079	
Flow Log	6 7.976	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	0.006	12.066	
Analysis & Synthesis	7 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Fitter	8 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Assembler	9 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Summary	10 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Parallel Compilation	11 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
SDC File List	12 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Clocks	13 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Slow Model	14 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Flow Summary	15 7.992	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.029	12.015	
Setup Summary	16 8.048	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.984	
Hold Summary	17 8.048	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.984	
Recovery Summary	18 8.048	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.984	
Removal Summary	19 8.048	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.984	
Minimum Pulse Width Summary	20 8.048	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.984	
Worst-Case Timing Paths	21 8.114	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.009	11.949	
Setup: 'sopc_clk'	22 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Hold: 'sopc_clk'	23 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Recovery: 'sopc_clk'	24 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Removal: 'sopc_clk'	25 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Minimum Pulse Width: 'sopc_clk'	26 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Datasheet Report	27 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Fast Model	28 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Multicorner Timing Analysis Summary	29 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	
Multicorner Datasheet Report Summary	30 8.152	first_nios2_systeminstfirst_nios2_system_cocpuM_ctl_shift_rot	first_nios2_systeminstfirst_nios2_system_cocpuM_status_reg_pe	sopc_clk	sopc_clk	20,000	-0.004	11.919	

Figure 8: Slowest data delay of the system

Compilation Report - nios2_quartus2_project				
Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	82.43 MHz	82.43 MHz	sopc_clk	

Figure 9: Maximum frequency of the system

Nios II BSP Properties

Sopclnfo: M:\ELEC373\assignment4\niosII_hw_dev_tutorial\first_nios2_syst

Flags

Defined symbols:	none
Undefined symbols:	none
Assembler flags:	-Wa,-gdwarf2
Warning flags:	-Wall
User flags:	none

Debug level: Off

Optimization level: Level 3

- ☒ Reduced device drivers
- ☒ Support C++
- ☐ GPROF support
- ☒ Small C library
- ☐ ModelSim only, no hardware support

Figure 10: O3 enabled for fair comparison

The screenshot shows a code editor with a file named `hello_world.c` and a tab for `system.h`. The code in `hello_world.c` includes `<stdint.h>`, `<stdlib.h>`, and `<sys/alt_timestamp.h>`. The `main` function initializes a variable `num` to `0x00000FFF` and prints it. It then checks if an `alt_timestamp_start()` device is available. If not, it returns `EXIT_FAILURE`. Otherwise, it measures the time taken by hardware (`hw_start`, `hw_cnt`, `hw_end`) and software (`sw_start`, `sw_cnt`, `sw_end`) to count the leading zeros of `num`. The hardware implementation takes 148 ticks, while the software implementation takes 338 ticks.

```
#include <stdint.h>
#include <stdlib.h>
#include <sys/alt_timestamp.h>

int main()
{
    uint32_t num = 0x00000FFF;
    printf("The num to be counted for leading zero is %08x\n\n", num);

    if (alt_timestamp_start() < 0)
    {
        printf ("No timestamp device available\n");
        return EXIT_FAILURE;
    }

    alt_u32 hw_start = alt_timestamp();
    uint32_t hw_cnt = ALT_CI_CLZ(num);
    alt_u32 hw_end = alt_timestamp();
    printf("The count by hardware is %d\n", (int)hw_cnt);
    printf("The hardware implementation takes %u ticks\n\n", (unsigned int) (hw_end - hw_start));

    alt_u32 sw_start = alt_timestamp();
    int sw_cnt = __builtin_clz(num);
    alt_u32 sw_end = alt_timestamp();
    printf("The count by software is %d\n", sw_cnt);
    printf("The software implementation takes %u ticks\n\n", (unsigned int) (sw_end - sw_start));
}
```

The console output shows the following results:

```
part-b Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
The num to be counted for leading zero is 00000fff

The count by hardware is 20
The hardware implementation takes 148 ticks

The count by software is 20
The software implementation takes 338 ticks
```

Figure 11: The hardware implementation is faster than the software implementation

You should have considered how long the call to `alt_timestamp()` took. The CI should take 1 clo

Also test the time for different values the software version would depend on the number