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own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web).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

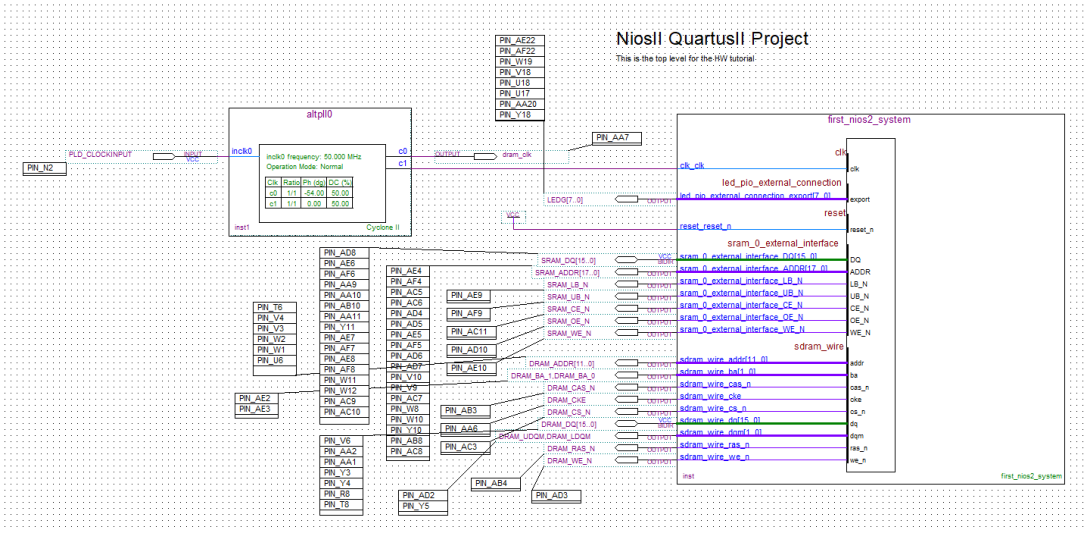


Figure 1: Block diagram of the Nios II system

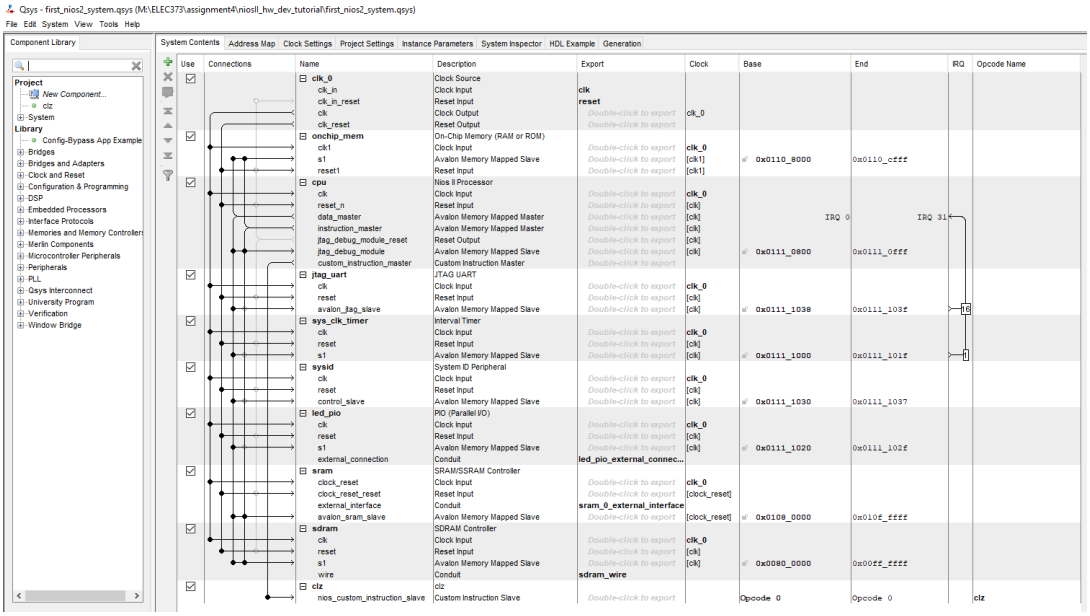


Figure 2: Nios II system contents

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 0×0108000 to 0×010FFFFF. The dynamic RAM was mounted to the address from 0×00800000 to 0×00FFFFFF. 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.

memtest_small.c

Problems | Tasks | Console | Properties | Nios II Console

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 0xFFFFFF
 -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:
>
0x010fffff
0x010fffff

Testing RAM from 0x1080000 to 0x10FFFFF
 -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?

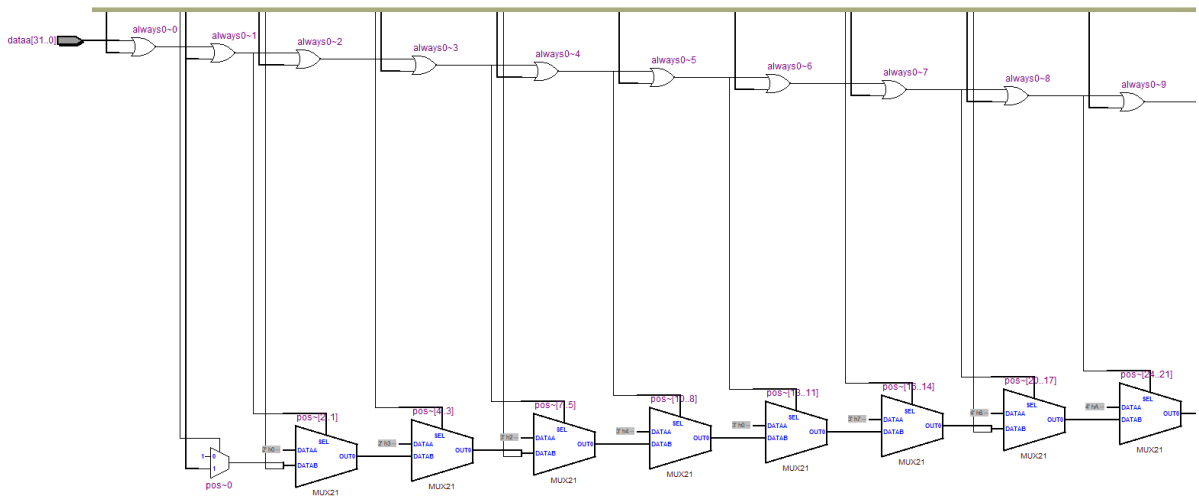


Figure 4: Partial view of the hardware implementation of the clz instruction

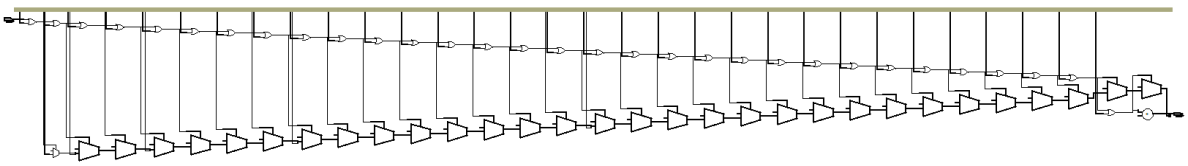
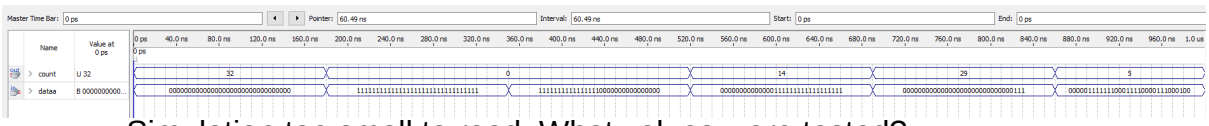


Figure 5: Rough schematic of the hardware implementation of the clz instruction



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.

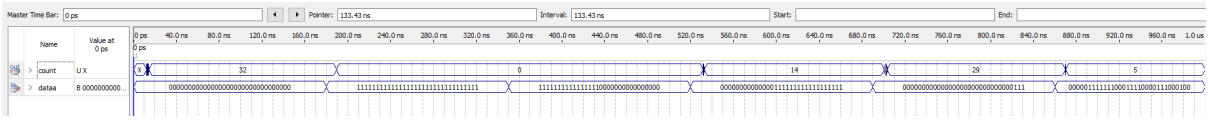


Figure 7: Timing simulation results of the clz instruction

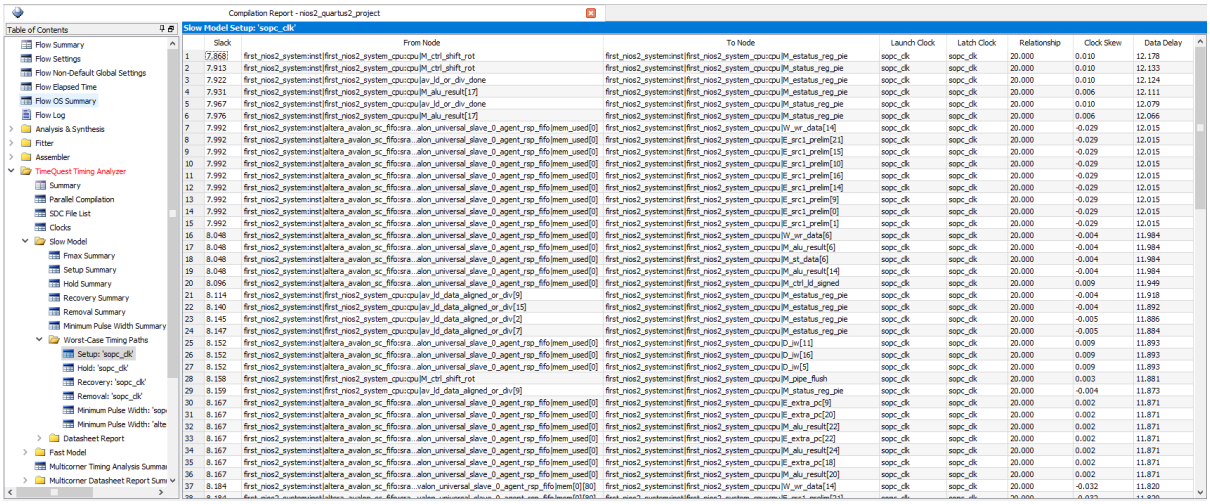


Figure 8: Slowest data delay of the system

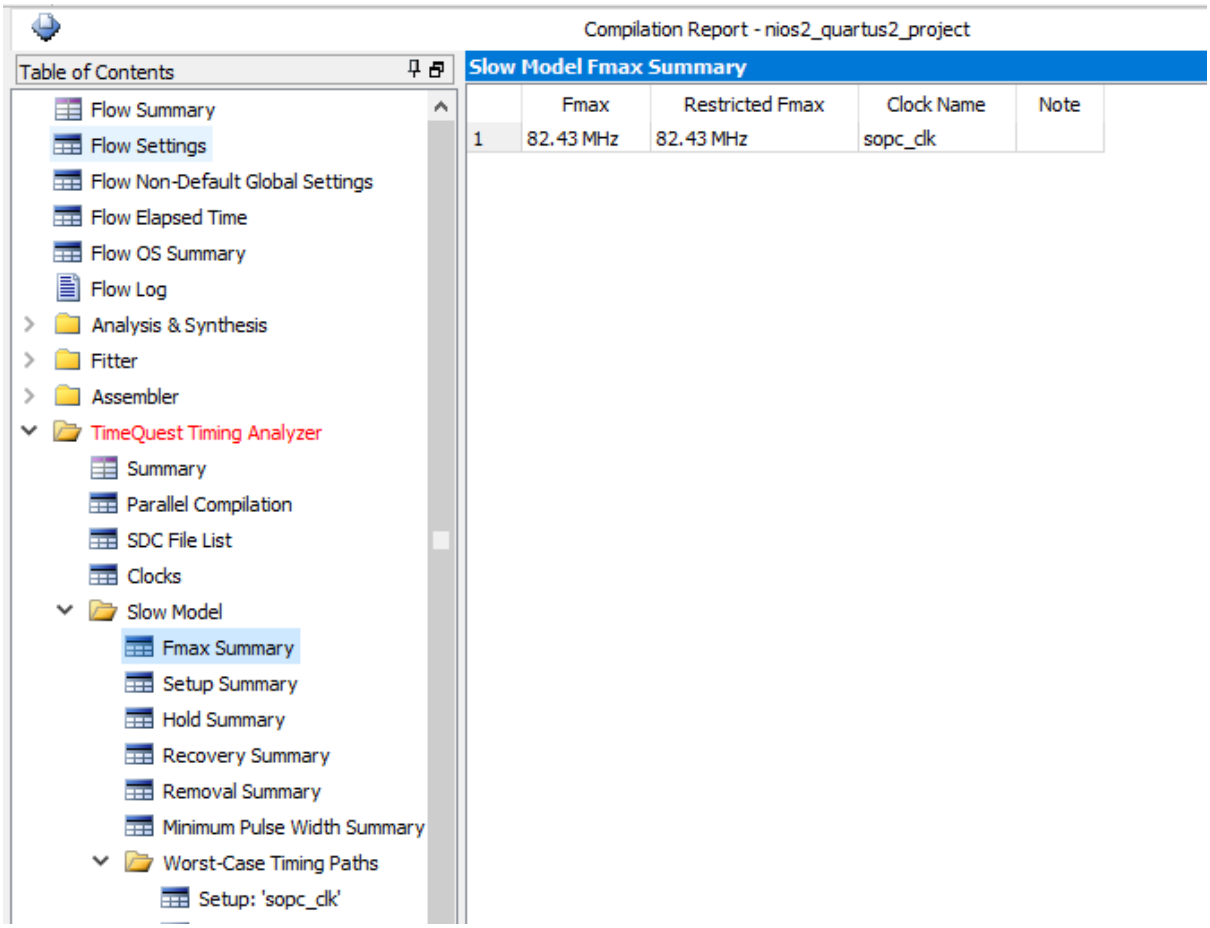


Figure 9: Maximum frequency of the system

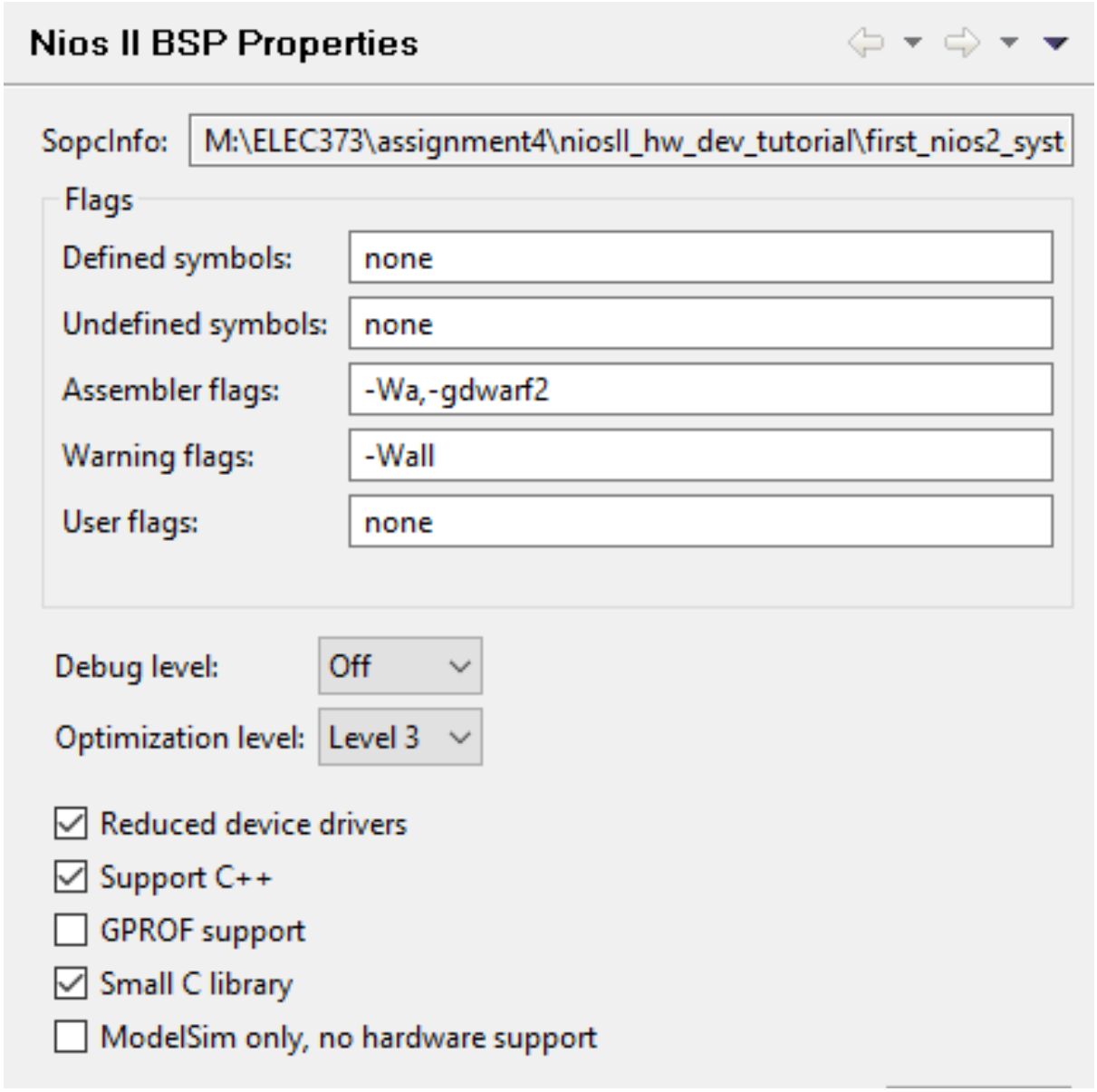


Figure 10: O3 enabled for fair comparison

hello_world.c | system.h

```c
#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));
}
```

Tasks | Console | Properties | Nios II Console

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