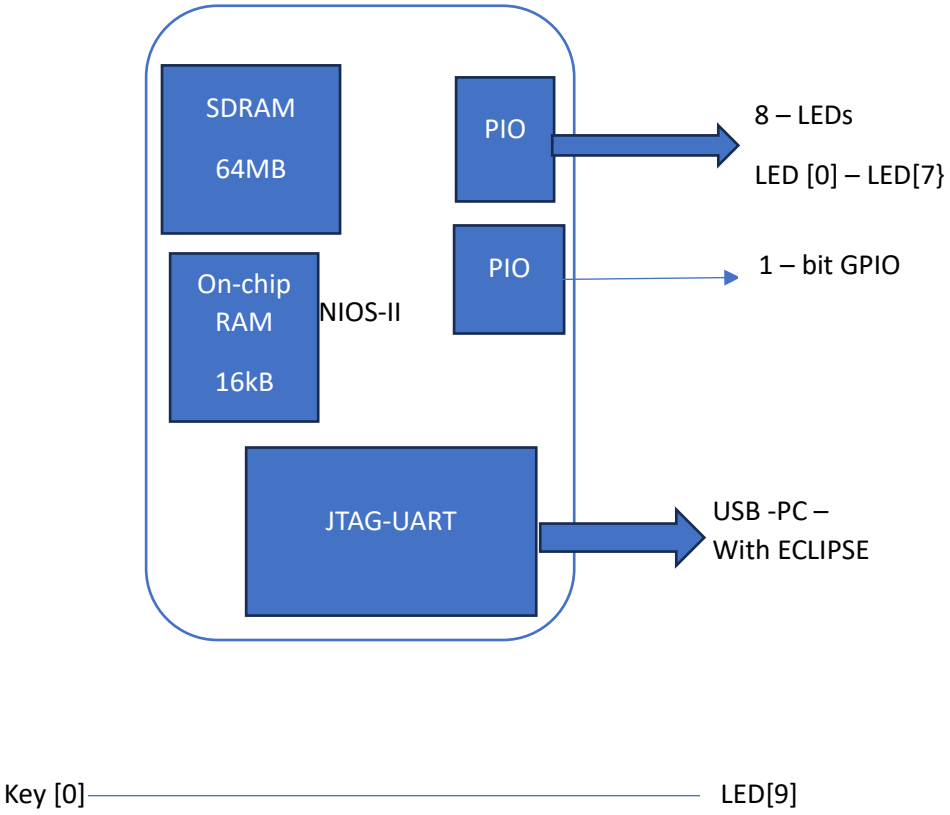
Lab 3 – Submission sheet (2025)
ECE3073 – Computer Systems
Monash University Malaysia
NIOS WITH DRAM AND DEVICE INTERFACE

Task 1:

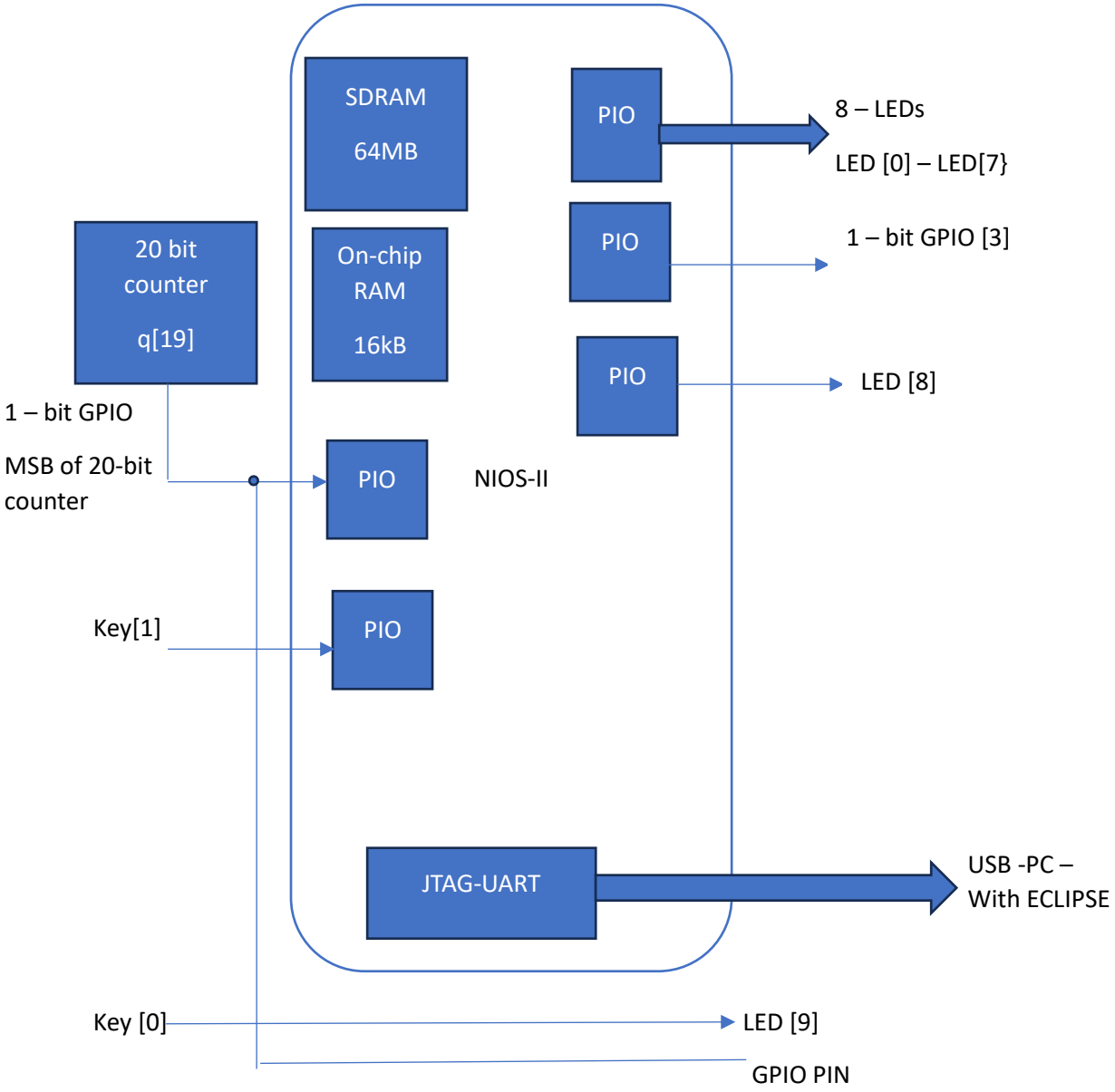
Design and construction of NIOS-II Microcomputer system with SDRAM and JTAG-UART to (1) print a simple hello world in Eclipse console window, (2) Interface 8 LEDs (LED[0] to LED[7]) and 1- bit GPIO (same as previous lab) but we program using C. Demonstrate the display of "Hello from Nios" printed in your console window. Further – 8 LED is HIGH state through C prg. (Also connect Key [0] to LED [9] in schematics – this for checking purpose – refer video)



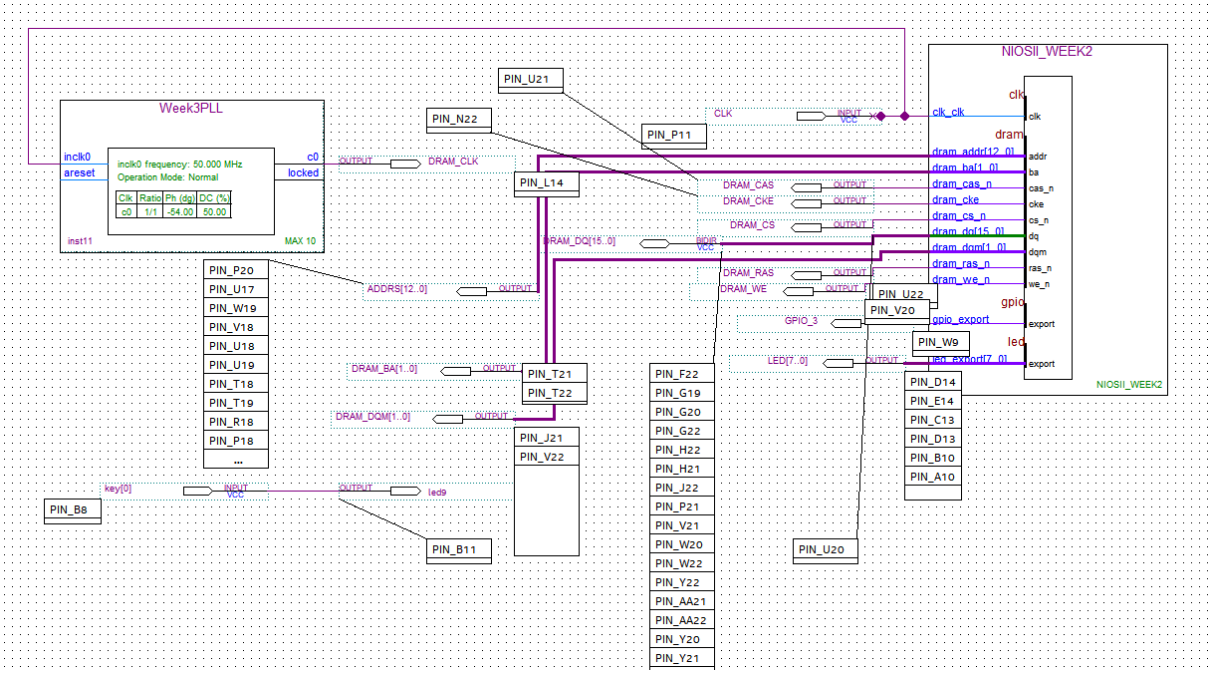
Task 2:

Add, 20-bit counter, where the most significant bit (MSB) of the counter is added to NIOS as input PIO (via 1bit – PIO) – this we designate as device 1. A GPIO pin is next connected to MSB at FPGA schematics (not in NIOS) – this we use for later oscilloscope calculations. Next an additional LED [8] as flag indicators. Finally include key [1] as input to the NIOS via PIO as device 2. Both device 1 and 2 will be interfaced to NIOS via polling loop – Write C-code to display the number of times rising edge occurred in MSB in 8 bit format using the 8 LEDs. Further when Key[1] pressed toggle the LED [8] state. As a subtask you have to build the 20-bit counter.

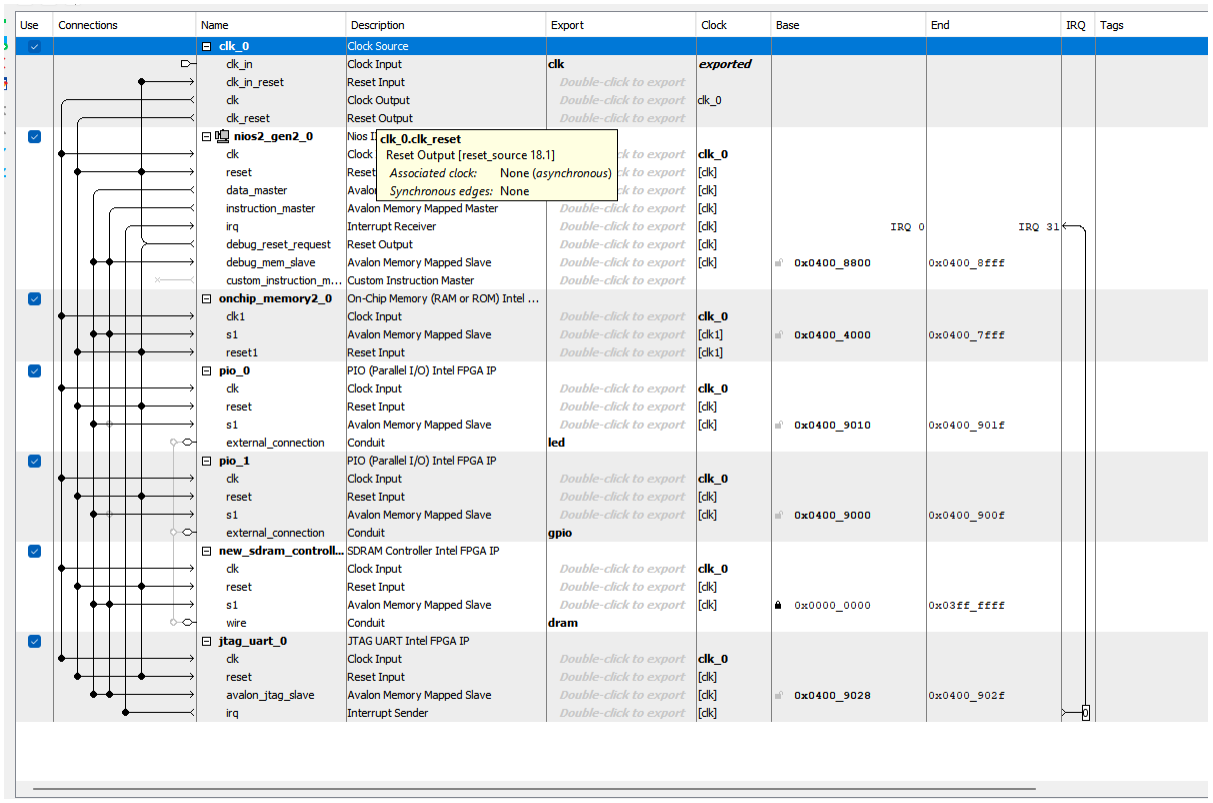
Lab 3 – Submission sheet (2025)
ECE3073 – Computer Systems
Monash University Malaysia
NIOS WITH DRAM AND DEVICE INTERFACE



Lab 3 – Submission sheet (2025)
ECE3073 – Computer Systems
Monash University Malaysia
NIOS WITH DRAM AND DEVICE INTERFACE



Schematic reference



Qysys reference

Lab 3 – Submission sheet (2025)
ECE3073 – Computer Systems
Monash University Malaysia
NIOS WITH DRAM AND DEVICE INTERFACE

Fill up the following

(1) What is the frequency of this MSB rising edge? Your answer .....47.684Hz.....

(2) And what period does this correspond to (in milliseconds)..... 0.020971s .....

PIO	Address (in HEX)
8-bit red LED PIO base address	0x400_9040
1-bit PIO flag base address	0x400_9030
1-bit PIO red LED base address	0x400_9000
1-bit PIO key base address	0x400_9010
1-bit PIO counter MSB base address	0x400_9020
Counter PIO interrupt level	

Task 3: Calculate the latency in polling, using disassembly of C to Assembly calculate theoretical latency. Using oscilloscope and GPIO's used in the previous task calculate practical latency and fill the table below.

Description	Latency (ns)
Expected minimum latency	6T * 3 * 20ns = 360ns
Expected maximum latency	6T * 15 * 20 = 1800ns
Expected increase in latency when KEY[1] is pressed-	6T * 5 * 20ns = 600ns

Description	KEY[1] not pressed -
Minimum observed latency (clock cycles)	98 clocks
Minimum observed latency (ns)	1960ns
Maximum observed latency (clock cycles)	128 clocks
Maximum observed latency (ns)	2560ns
Average observed latency (clock cycles)	113 clocks
Average observed latency (ns)	2260ns

Comment on the change in latency when Key[0] is pressed:

Latency decreased from

Task 4:

Complete the following and test your understanding with SDRAM interfacing

(1) How would you access data that is stored in the middle of a word? For example, how would you read the front byte of this word (just the AA, and ignore the rest): 0x**AA**000000

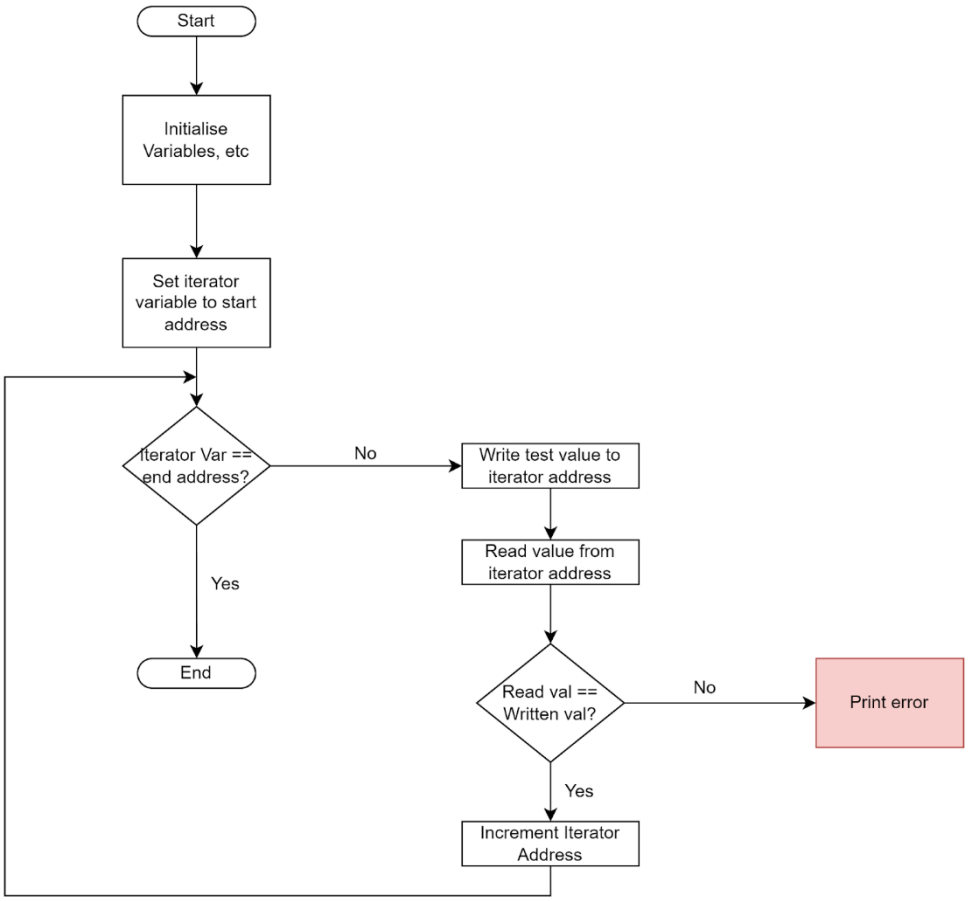
(2) What about if we only want the lower 4 bits of that byte, how would you access “B” in this word: 0x0**B**000000?

(3) Now that you have set up the SDRAM system, we will use it to write a memory tester. A memory tester is a simple program that verifies the proper operation of a region of memory by confirming each storage location in the memory device is working. Your task is to write a C program that writes a value to every memory location, subsequently reads the value stored in each memory location, and verifies that the value stored is the same as the value that was written to that memory location.

You should ensure that every bit of every memory location can have both a 0 and a 1 written to it. To lower the run time of your program, **you can shrink**

your memory range from the full 32MB, down to 1MB, starting at address 0x100000 (or any address of your choice).

The flowchart below shows a basic algorithm you can use to test the SDRAM addresses, keep in mind that you might need to read and write two different values to test that all bits work as expected.



Flowchart for memory tester