NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2 AY2004/2005

CS2271 – Embedded Systems

Apr/May 2005 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FOUR** questions and comprises **SIXTEEN (16)** printed pages, including this page. The weightage for each question is as indicated, and total 50 marks.
2. Answer **ALL** questions within the spaces provided in this booklet.
3. This is an Open Book examination.
4. Please write your Matriculation Number below.

MATRICULATION NO: _____

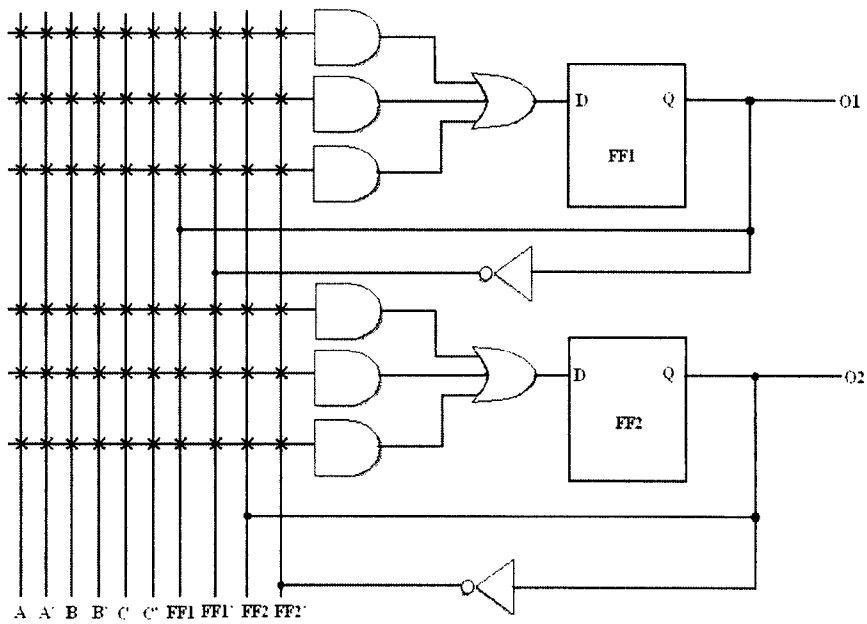
This portion is for examiner's use only

Question	Marks	Remarks
Q1		
Q2		
Q3		
Q4		
Total		

Question 1 Programmable Logic Devices **(8 Marks)**

a. The diagram below shows a section of a PAL with flip-flops on the outputs. This PAL takes 3 inputs A, B and C. Circuits to generate A', B' and C' are not shown.

Explain how this is different from an FPGA with the equivalent number of flip-flops and gates. (4 marks)



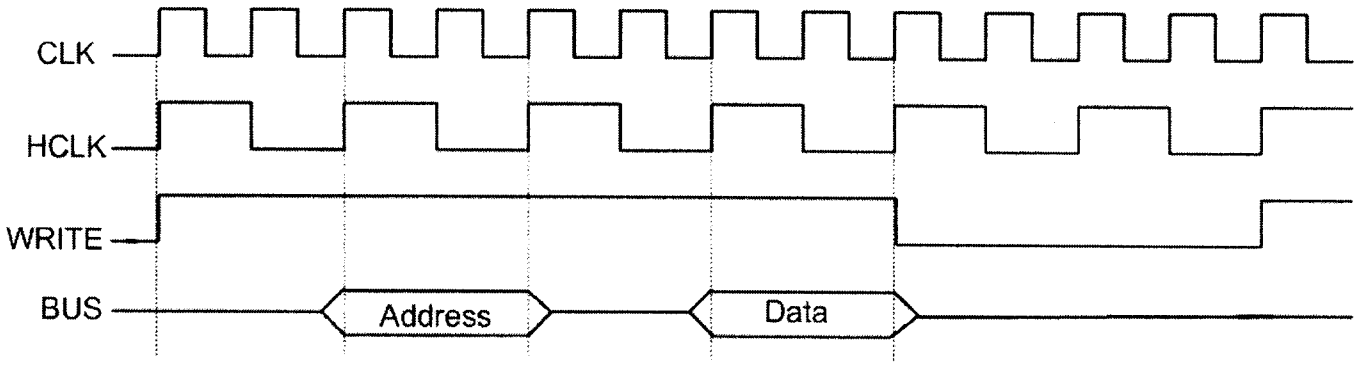
b. State and explain the similarities and differences between a Programmable Logic Array (PLA) (with no flip-flops), and a Programmable ROM (4 marks)

Question 2 Handel-C Programming **(12 Marks)**

a. The following diagram shows the WRITE timing diagram of a microprocessor with a "split-cycle" bus, which means that the same bus is used to send addresses and to send data. This microprocessor has a 16-bit address and an 8-bit word size. Data written by the microprocessor occupies the lowest order 8-bits of the bus.

The microprocessor is connected to an FPGA, and will assert the WRITE signal to the FPGA when it is writing to it, and asserts the READ signal when it is reading from the FPGA. There is a pause of at least 4 external clock cycles before the microprocessor tries to WRITE again.

CLK is the external clock, while HCLK is the Handel-C clock.

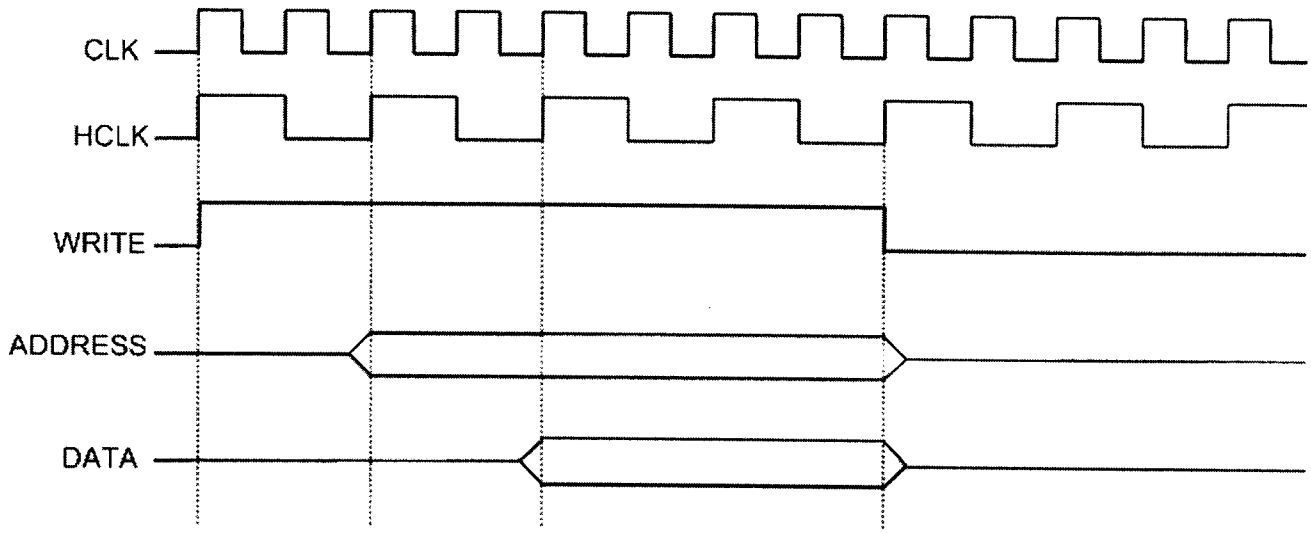


i) Write the `interface` declaration to read this bus. Assume that you can use pins P1 to P64 of the FPGA to interface to this microprocessor (note: You do not have to use all 64 pins). Pin P1 is connected to the LSB of the microprocessor address/data bus.

You should also declare any other variables that you might need to interface with this microprocessor. (3 marks)

ii) Write a Handel-C function `read_cpu` that waits until the microprocessor starts writing to the FPGA. When the microprocessor initiates a write, your code should read the address and data written by the microprocessor, and return them through two global variables `cpuAddress` and `cpuData`. Your function should then wait again until the next microprocessor write (4 marks).

b. Given the following timing diagram of an asynchronous static RAM with separate address and data buses. The FPGA asserts the WRITE signal, then places the address and data onto the respective buses.



The following variables have been declared and bound to the relevant pins in an `interface` declaration.

```
signal unsigned 2 rw=0; // Writing to "1" LSB asserts the WRITE signal.
                        // Writing "1" to MSB asserts the READ signal.
signal unsigned 16 ramAddr; // Variable for writing to address bus.
signal unsigned 8 ramData; // Variable for writing to the data bus.
```

i) Why are all the variables declared with "signal"? (1 mark)

ii) Write a Handel-C function called `cpu_ram_interface` that will interface between the split-cycle micrprocessor bus and the SRAM with separate address and data buses. You are allowed to modify the read_cpu function you wrote earlier. If you do so, please note down what modifications were made. (4 marks)

Question 3 Real Time System Architectures **(12 Marks)**

a. List and explain the similarities and differences between a non-pre-emptive RTOS and function queue scheduling . (3 marks)

b. Show, on the two (unrelated) C!! code fragments below running on a pre-emptive RTOS, under what circumstances they might fail, and what the effect of the failure(s) is/are going to be. You should also indicate on the code fragment how to correct the problem, if possible. (6 marks)

C!! Code Fragment 1

```
int stack[MAX_SIZE];
int sp = 0;

void push(int data)
{
      if(sp < MAX_SIZE)
      {
            disable_interrupts();
            stack[sp++] = data;
            enable_interrupts();
      }
}

int pop()
{
      if(sp > 0)
      {
            disable_interrupts();
            return stack[--sp];
            enable_interrupts();
      }
}
```

C!! Code Fragment 2

```
int sharedData1, sharedData2;

/* All of these are initialized
in main, which is omitted here */
OSSemaphore sema1, sema2;
OSMailBox mb1, mb2;

void task1()
{
   int data;
   recvmsg(mb2, &data);
   take_sema(sema2);
   !! Do some processing on sharedData2
   sendmsg(mb2, sharedData2);
   take_sema(sema1);
   !! Do some processing on sharedData
   sendmsg(mb1, sharedData);
   release_sema(sema1);
   release_sema(sema2);
}

void task2()
{
   take_sema(sema1);
   !! Do some processing on sharedData
   sendmsg(mb1, sharedData);
   take_sema(sema2);
   !! Do some processing on sharedData2
   sendmsg(mb2, sharedData2);
   recvmsg(mb1, &sharedData1);
   release_sema(sema1);
   release_sema(sema2);
}
```

c. The ECPU compare and conditional branch statements are separate. I.e. to branch to target if R1 is greater than R2, we'd do:

```
CMP R1, R2
BGT target
```

With reference to the SCPU/ECPU hardware specification for CMP and BGT, is this arrangement suitable for programs that run within a pre-emptive or interrupt-driven environment? Why or why not? (3 marks)

Question 4 Real Time System Design **(18 Marks)**

Please read the problem description carefully before attempting this question.

A Full Authority Digital Engine Control (FADEC) system controls the amount of fuel flowing to a jet engine based on the altitude and airspeed of the plane, the outside air temperature and the flight-deck throttle settings.

The airspeed and altitude are computed from the air pressure sensed by the pitot-tube air pressure sensors, while the external air temperature is sensed by an Outside Air Temperature (OAT) probe. Computing altitude and airspeed may take a lot of CPU time. The pitot tube air pressure sensors and the OAT probe are not interrupt driven, and are polled approximately every 300 ms for their readings. The throttle is interrupt driven, and they interrupt the CPU whenever the pilot shifts the throttle.

In addition the FADEC also monitors the rotation speed of the low pressure (N1) and high pressure (N2) compressors, the turbine temperature (EGT) and ten fire sensors located around the engine. The FADEC will sound an alarm and light up a warnings on the flight-deck if the N1, N2 and EGT readings are abnormal, or if the engine is on fire, as detected by one or more of the fire sensors. The N1, N2 and EGT sensor electronics are interrupt driven, and they produce a reading every 100 ms. The ten fire detectors are connected to a single non-maskable interrupt.

The FADEC is implemented with the ECPU, with 32 bit instruction words and 16 bit data words.

a. Evaluate the Round Robin with Interrupts, Function Queue Scheduling and Real Time Operating System architectures for their suitability to this application. (3 marks)

b. Since computing airspeed and altitude from the pitot tube air pressure sensors takes a substantial amount of computing time, would Rate Monotonic Scheduling be suitable for this application? Why or why not? (2 marks)

c. Your project manager has given you the following specifications for the FADEC (All addresses are in decimal). Use these specifications to position your ISRs and task routines properly in memory, as well as to read from the correct ports and write to the correct memory locations after reading the ports.

Memory Range (Decimal)	Purpose
0 - 7	Interrupt Vector Table. IRQ0: Throttle IRQ1: N1 speed IRQ2: N2 speed IRQ3: EGT IRQ7 (NMI): Fire sensors
8-30000	Reserved for RTOS
30100	N1 rotation speed (in %age maximum rotation speed)
30101	N2 rotation speed (in %age maximum rotation speed)
30102	EGT
30103	Throttle Setting (in %age power. 100% = full power, 0% = idle power)

Table 1. Data Memory Usage

Address Range (Decimal)	Purpose
0-29999	Reserved for RTOS
30000	Interrupt service routine (ISR) for N1
30800	ISR for N2
31600	ISR for EGT
32400	ISR for throttles
33200	Polling routine for OAT probe
34000	Polling routine for pitot tube pressure sensor
34800	ISR for fire sensor
36000	FADEC application entry point
37000 to 65535	Task Routines

Table 2. Program Memory Usage

I/O Port (Decimal)	Purpose
3210	Throttle settings (in %age power. 100% = full power, 0% = idle power)
3211	N1 (in %age maximum rotation speed)
3212	N2 (in %age maximum rotation speed)
3213	EGT
3214	OAT probe
3215	Pitot tube pressure sensor
3216	Alarm Switch on by writing a non-zero value. Switch off by writing "0".
3217	N1 overspeed light. Switch on by writing a non-zero value. Switch off by writing "0".
3218	N2 overspeed light. Switch on by writing a non-zero value. Switch off by writing "0".
3219	Engine overheat light. Switch on by writing a non-zero value. Switch off by writing "0".

Table 3. I/O Port Assignments

i) Write the interrupt service routine for the throttle in ECPU assembly (3 marks)

Given that the RTOS manual lists the following information:

RTOS Entry Point	RTOS Function	Input Parameters
0	Initialize the RTOS	R0 = Scheduling Mode: 0 = Fixed 1 = Rate Monotonic 2 =Earliest Deadline First
4	Register an ISR	R0 = Address of ISR R1 = IRQ number
8	Register a task	R0 = Address of task R1 = Task priority
12	Start the RTOS	Nil
14	Block on a memory location. Task remains in BLOCKED state until an interrupt handler writes to that memory location.	R0 = Program address of task to block R1 = Data address of memory location to block on.
16	Block a task for a fixed time	R0 = Program address of task to block R1 = # of milliseconds to block
... ..	.....	.....

To call an RTOS function, set up the appropriate registers, then JSR to the entry point for that function.

ii) Write the task that reads the OAT probe and pitot-tube air pressure sensor in ECPU assembly (4 marks). Ensure that it complies with the specification given earlier.

Given the following information for a CFM56-7 high-bypass turbofan engine:

Throttle Setting	Maximum N1	Maximum N2
0%	45%	50%
25%	65%	75%
50%	80%	85%
75%	92%	95%
100%	100%	100%

iii) Write, in ECPU Assembly, a task routine for N1 that sounds an alarm and lights the N1 warning light when N1 exceeds maximum limits. Make this the first task routine in memory, subject to the Data Memory Usage shown in Table 1. (6 marks)

End of paper