2023 Final Exam
COMP2300/ENGN2219/COMP6300

Digital Logic Fundamentals

Q1. During a work assignment, you are asked to design a combinational circuit with a three-bit input, {A, B, C} (A is the most significant bit and C is the least significant bit), and two 1-bit outputs, Factorial and Div3. The value of each output is determined as follows:

- The output Factorial is 1 if the input number is equal to its factorial. Factorial is the product of all positive integers (excluding 0) that are less than or equal to the input number. When the input combinations are all 0, the output is 1

- The output Div3 is 1 only when the input 3-bit number is divisible by 3

Answer the following questions.

A. Draw the truth table for the combinational circuit.

B. Express the output Factorial as the simplest sum of products representation. Show your work step-by-step.

C. Express the output Div3 as the simplest sum of products representation. Show your work step-by-step.

D. Draw a schematic of Factorial and Div3 using basic logic gates.

E. It is well-known that we can implement any Boolean function either using basic logic gates or a look-up table (N to 1 multiplexer). Explain the tradeoff between the two approaches.

Q2. Consider the high-level schematic of a general-purpose hardware circuit shown that computes two outputs from three inputs. Circle the correct answer. Each question carries 4 points. We will subtract 1 point for each incorrect answer and award 0 points for unanswered questions.

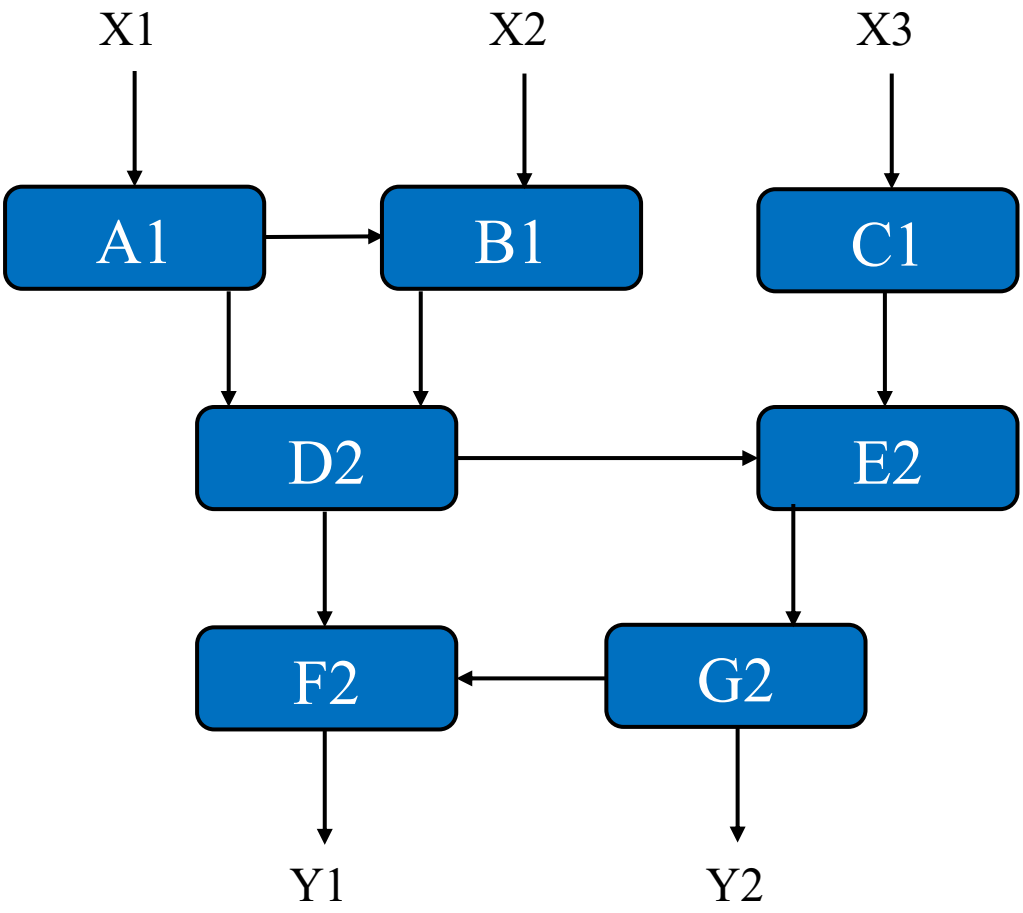
A. Y2 depends on:
   a. X3
   b. X1 and X3
   c. X2 and X3
   d. X1, X2, and X3

B. Y1 has no relationship to X3:
   a. True
   b. False

C. If C1 has a delay of 10 units, and all other components have a delay of 3 units, then the critical path of Y1 consists of:
   a. A1, D2, F2
   b. A1, B1, C1, F2
   c. C1, E2, G2, F2
   d. A1, B1, D2, E2, G2, F2
   e. B1, D2, F2
   f. Need more information

D. If C1 has a delay of 5 units, and all other components have a delay of 3 units, then the critical path of Y2 consists of:
   a. C1, E2, G2
   b. A1, D2, E2, G2
   c. A1, B1, D2, E2, G2
   d. B1, D2, E2, G2
   e. A1, B1, E2, G2, F2
   f. Need more information

E. If the input is changed from (X1 =1, X2 = 0, X3 = 0) to (X1 = 1, X2 = 0, X3 = 1), then the following components are triggered to recompute their outputs:
   a. A1, B1, C1, D2, E2, G2, F2
   b. C1, E2, G2
   c. A1, B1, D2, F2
   d. C1, E2, G2, F2
   e. A1, B1, E2, G2, F2


X1
X2
X3
A1
B1
C1
D2
E2
F2
G2
Y1
Y2

Finite State Machines

Q1. In this question, you will design a finite state machine-based video game controller. The video game has the following specification.

- There are two input buttons, B1 and B2. When the player presses any of these buttons, a signal is sent to the controller. Both buttons are never pressed at once. Assume at least one button is pressed every clock cycle.

- The game console has two output LEDs, RED and BLUE. These LEDs are controlled by the controller via two output signals.

- The BLUE LED turns ON when the player presses the buttons in the following sequence, B1, B2, B1. (Not necessarily consecutively.)

- The RED LED turns ON when the player presses the two buttons in the following sequence, B2, B2, B1. (Consecutively.)

- Once the LEDs are turned on, they remain turned on forever.

Design the finite state machine controller by finishing the following tasks. You may find it more convenient to build two separate Moore machines, one for each output.

A. Design the Moore state transition diagram(s) for the above specification. Build the encoded state transition tables. Specify the next state and output equations.

   Note: You do not need to simplify the equations or draw the schematic of the resulting circuit.

B. Explain how you will decide the clock frequency for the state machine controller.

Q2. A specialized pipelined processing unit (PPU) for a high-performance computational problem is shown below. The PU processes an input token (set of inputs) using three stages. Your job is to compute the time it takes to process several tokens for different scenarios discussed below.

First, the timing specification for the combinational and sequential elements of the circuit is given below.

- The propagation delay of Stage 1 is equal to 3.4 nanoseconds.
- The propagation delay of Stage 2 is equal to 3.3 nanoseconds.
- The propagation delay of Stage 3 is equal to 3.7 nanoseconds.
- The clock-to-Q propagation delay (t_{pcq}) is equal to 0.2 nanoseconds.
- The setup time (t_{setup}) is equal to 0.1 nanoseconds.
- The hold time (t_{hold}) is zero.

Now, consider the following scenarios. Assume that in the case of multiple PPUs, tokens are evenly partitioned among the PPUs. How long does it take to process the specified tokens?

A. Twelve tokens and one PPU

B. One billion tokens and one PPU

C. One billion tokens and four PPUs

Assembly and ISA

Q1A. There is no link register in the QuaC ISA. How can we write assembly code with functions without a link register? How would you write code to return from a function in QuAC?

Q1B. Write a recursive function (starting at the label fib) which computes Fibonacci numbers in QuAC assembly. The Code is shown below.

```
int fibon(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
     return 1;
  else
    return fibo(n-1) + fibo(n-2);
}
```

The parameter to the function is provided in `r1` and the result should be returned in `r1`. Add a comment explaining which registers are caller vs callee saved. For this question you can assume the input is a 16-bit unsigned integer and the result will also fit into a 16-bit unsigned integer so do not worry about numerical overflow in your function.

You can use any instruction from the base QuAC ISA specification. In addition, we provide two new instructions from the QuAC isntruction set with better support for writing functions in QuAC assembly. The instructions and their semantics are shown below.

```
push ra
sp := sp - 1
[sp] := ra

pop ra
ra := [sp]
sp := sp + 1
```

We define `r6` to be the stack pointer (`sp`). You may assume that this register has already been appropriately initialized. There is no link register.

Q2. Circle whether each of the following is an aspect of the ISA or the microarchitecture. We will subtract 1 point for each incorrect answer and award 0 points for unanswered questions. Each question carries 2 points.

1. One-bit branch history prediction
   a. ISA
   b. Microarchitecture

2. Location of the bit in the machine instruction that identifies whether a specific memory instruction is a load or a store instruction
   a. ISA
   b. Microarchitecture

3. Number of levels in the cache hierarchy of a processor
   a. ISA
   b. Microarchitecture

4. Number of cores in the processor
   a. ISA
   b. Microarchitecture

5. Width of registers in the register file
   a. ISA
   b. Microarchitecture

6. Number of cycles it takes to execute an ADD instruction in a multi-cycle processor
   a. ISA
   b. Microarchitecture

7. The size of memory addressable by programs
   a. ISA
   b. Microarchitecture

8. Whether memory is byte-addressable or word-addressable
   a. ISA
   b. Microarchitecture

9. Superscalar issue width of a dynamically scheduled out-of-order processor
   a. ISA
   b. Microarchitecture

10. The conditions for stalling the in-order CPU pipeline on a load-use hazard
    a. ISA
    b. Microarchitecture

11. Software-based exploitation of instruction-level parallelism with very simple hardware (e.g., VLIW)
    a. ISA
    b. Microarchitecture

12. The number of ALUs
    a. ISA
    b. Microarchitecture

13. The policy to stall on branch instead of using prediction
    a. ISA
    b. Microarchitecture

14. Register renaming to eliminate false dependences
    a. ISA
    b. Microarchitecture

15. Choosing port-mapped I/O for communicating with peripherals
    a. ISA
    b. Microarchitecture

Q3. Pick all ISA decisions from below that improve code density. You can make one mistake and still get the full points.

A. Complex instructions instead of reduced instructions

B. Sophisticated memory addressing modes

C. Memory-mapped I/O

D. Instruction cache that resides close to the CPU

E. Microprogramming

F. Wide registers

G. Algorithm

H. Conditional execution for all instructions

I. A large programmer-visible register file

J. Clock frequency

Input/Output

Q1. It has been observed that in a large datacenter, during the peak hours of the daytime, the processing load consists of 50% CPU activity and 50% disk activity. The users of the service are complaining that the response time is very slow. As a manager of the service, you have been briefed that an upgrade of disks would cost $4,000 and make them 1.2 times as fast as they are currently. You have also been told that an upgrade of CPU will make it 2 times faster for $6,000. What is the cost per 1% increase in performance with the CPU upgrade and the disk upgrade? Which option would you recommend, and why? Show your work.

Q2. You have a choice of following I/O systems.

- Programmed polling I/O
- Programmed interrupt driven I/O
- DMA interrupt driven I/O

Which one would you choose for each of the below peripherals? Briefly Explain the reason.

A. Slow peripheral that sends one character at a time (e.g., keyboard)
B. Slow storage device that sends an entire 4 KB block when requested
C. Temperature sensor for thermostat

Microarchitecture

Q1. Pick all features from below that impact clock frequency. You are allowed to make one mistake and still get the full points.

A. Instruction complexity

B. Programming language

C. VLIW (very long instruction word)

D. Conditional execution

E. Complex memory addressing modes

F. Compiler

G. Algorithm

H. Number of programmer visible registers

I. Transistor quality

J. Pipeline depth

K. Type of stack (downward / upward)

L. Type of I/O (Interrupts / polling)

M. Code size

N. Microbit colour

Q2. This question requires you to answer questions related to the execution of the following instruction sequence. First, study the instruction sequence below. Then, answer the questions.

```
i1:   MOV R0, #0
i2:   MOV R1, #10
i3:   MOV R2, #200
i4:   MOV R3, #0
i5:   MOV R4, #0
     LOOP:
i6:   CMP R0, R1
i7:   BEQ L2
i8:   LDR R3, [R2], 4
i9:   ADD R3, R3, R1
i10:  ADD R4, R3, R4
i11:  SUB R1, R1, #1
i12:  B LOOP
     L2:
```

A. How many times is i7 executed if the above code runs on a multi-cycle CPU?

B. Consider the ARM 5-stage pipelined CPU without forwarding and hazard detection in hardware. Your task is to insert NOPs in the assembly code so that it executes correctly on the pipelined CPU. Rewrite the code above by inserting NOPs. What is the IPC if the code with NOPs is executed on this pipelined CPU? (Recall that IPC is acronym for instructions per cycle.)

C. Now consider the ARM 5-stage pipelined CPU with forwarding and hazard detection. In the case of conditional branches, this CPU predicts branches are not taken. You can make the following assumptions.

- Conditional branches resolve and update the PC at the end of EX stage.
- Unconditional branches stall fetch until the PC is updated in the WB stage.
- Memory accesses take one cycle.

For one execution of the loop body (i.e., in some iteration before the loop terminates), identify a pair of instructions where forwarding helps avoid stalls altogether. Also identify a pair of instructions which lead to one or

more stall cycles despite hazard detection. What is the IPC if the code executes on the pipelined CPU with forwarding and hazard detection?

Note: You can answer these questions using either text or an illustration.

D. Assume the CPU uses a more sophisticated branch predictor than always untaken. What is the prediction accuracy of the $Smith_2$ predictor (2-bit saturating counters) for the `BEQ` instruction if the above code executes only once? What if the above code executes four times?

Q3. Consider the following ARM assembly code. Answer the following questions.

```
i1: LDR R1, [R0, #16]
i2: LDR R2, [R0, #32]
i3: ADD R1, R1, #32
i4: ADD R2, R2, #32
i5: STR R1, [R0, #16]
i6: STR R2, [R0, #32]
```

A. List all the true and false (anti and output) dependencies in the above ARM assembly code. You can list dependencies using the following format: {`i1` to `i1`, anti, by way of `R0`}.

B. How many independent chains of instructions are there in the code? Write the chains in the following format {i1, i1, i1}.

C. What is the maximum number of instructions that can execute in parallel on a hypothetical machine with no structural hazards (unlimited ALUs, any number of memory accesses in flight at once)? Briefly explain your reasoning.

Q4. This question requires to you show the contents of different structures of an out-of-order processor that uses a scoreboard for resolving read-after-write hazards. Use the following assumptions.

- WAR and WAW hazards stall fetch in the RR stage.
- The CPU has an 8-entry architectural register file.
- All entries of the scoreboard are 1 prior to the execution of the first instruction. A busy register, i.e., value is being computed and dependent instruction must wait, is indicated by {v = 0} in the scoreboard.

- The issue queue is empty. The v bits of all slots are 0. All slots are available. An unavailable slot is indicated by the v bit set to 1.
- The pipeline is fully flushed before the execution of first instruction.
- The contents of the register file are shown below.
- Any memory access results in a cache miss.
- Memory accesses complete in three steps: (1) address generation, (2) data cache access, (3) main memory access. If data is not present in the cache, then the request is sent to memory during the next cycle.
- It takes four cycles for the memory to respond. Therefore, an access that does not find the requested word in the data cache spends 1 cycle computing the address, 1 cycle looking up in the data cache, and four cycles waiting for memory (6 cycles total in the EX-stage).
- The value of data word stored at memory location 16 is 0.

For the four instructions below, answer the following questions.

```
i1: LDR R2, [R1, #0]
i2: ADD R4, R3, #1
i3: ADD R6, R2, R4
i4: SUB R7, R6, #3
```

A. Provide the contents of the issue queue at the end of cycle # 8 and cycle # 12. You can pick any free slot in the issue queue for any instruction. Use the following format to answer this question using text.

   Cycle # X: {i1: v=1, dst tag = R0, rs1 rdy = 0, rs1 tag = R0, rs2 rdy = 0, rs2 val = 0}

   Also draw and attach the cycle-by-cycle pipeline diagram, as shown in the lecture slides.

B. Provide the contents of the scoreboard at the end of cycle #5 and cycle # 9. You can show the scoreboard contents in each cycle by writing a string of 8 binary digits, e.g., 00001111 for each cycle.

C. Provide the contents of the Register File at the end of cycle # 12. Use the format below.

   {R0 = 10, R1 = 10, R2 = 13, and so on for the remaining registers}

D. What is broadcasted on the common data bus (CDB) at the end of cycle # 8?

Initial state of the issue queue, scoreboard, and register file.

Scoreboard

	v
R0	1
R1	1
R2	1
R3	1
R4	1
R5	1
R6	1
R7	1

Register File

	value
R0	#10
R1	#16
R2	#10
R3	#10
R4	#10
R5	#10
R6	#10
R7	#10

Issue Queue (IQ)

v	dst tag	rs1 rdy	rs1 tag/value	rs2 rdy	rs2 tag/value
0					
0					
0					

Pipeline cycle-by-cycle diagram format:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
i1: LDR R2, [R1,#0]																	
i2: ADD R4, R3, #1																	
i3: ADD R6, R2, R4																	
i4: SUB R7, R6, #3																	

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
i1: LDR R2, [R1, #0]	FE	DE	RR	DI	IS	$EX_{@}$	$EX_{D\$}$	MEM	MEM	MEM	MEM	WB					
i2: SUB R3, R2, #6		FE	DE	RR	DI	IS	IS	IS	IS	IS	IS	EX	WB				
i3: ADD R4, R1, #5			FE	DE	RR	DI	IS	EX	WB								
i4: ADD R6, R3, R3				FE	DE	RR	DI	IS	IS	IS	IS	IS	EX	WB			

Bonus Questions

We now provide you an opportunity to attempt two bonus questions. We will not deduct any points for wrong answers. Each bonus question carries 10 points. These points will be added to your final score out of 300.

Q1. Rank the following microarchitectures according to their potential to exploit instruction-level parallelism (ILP). List the highest ILP microarchitecture first. Your answer may look like {A, B, C, D, E, F, G, H}. Briefly explain your reasoning for the top two entries in the ranking.

A. 4-issue superscalar with hardware speculation, but without register renaming

B. 2-issue superscalar in-order with forwarding and hazard detection

C. 4-issue superscalar with hardware speculation

D. Multi-cycle

E. 4-issue superscalar with dynamic scheduling, but without speculation and register renaming

F. Scalar in-order without forwarding and hazard detection (compiler inserts NOPs to resolve hazards)

G. Scalar in-order pipeline with forwarding and hazard detection

Q2. Rename the following instructions to eliminate false dependencies. The true dependencies must still be respected. The physical registers are {T0, T1, T2, T3}. Assume the register file is up-to-date and pipeline is full flushed before the first instruction is fetched.

```
i1: ADD R0, R1, #8
i2: ADD R1, R0, R7
i3: ADD R2, R3, #16
i4: ADD R3, R2, R7
```