EEE4120F GA2 2025
Preparation Test
for Verilog Assessment Competence

HANDOUT THIS TEST UPSIDE-DOWN

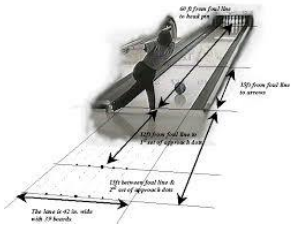
Prep Test Opportunity 1

Please fill in below:

YOUR NAME:	STUDENT NUMBER:

Duration: 45 minutes Total Marks: 100

Bowling flavoured



Instructions:

- Answer all questions.
- Clearly show all your working.
- Assume the clock frequency for clk is 1KHz (1ms period) and for fclk is 1MHz (1us period).

Section 1: Multiple Choice Questions (40 Marks)

NOTE: fill in multiple-choice answers using the answer block on the last page of this test.

Answer the following five multiple-choice questions. Each question is worth 8 marks.

Question 1.1
Which of the following best describes the primary purpose of a testbench in Verilog?

A) To synthesize the Verilog code into a netlist for FPGA implementation.

B) To verify the functional correctness of a Verilog module by providing inputs and checking outputs.

C) To perform static timing analysis on the designed circuit.

D) To define the physical constraints for placing and routing the design on an FPGA.

Question 1.2
In Verilog, what is the effect of using a blocking assignment (=) versus a non-blocking assignment (<=) within a synchronous always @(posedge clk) block?
A) Blocking assignments are for combinational logic, non-blocking for sequential logic.

B) Blocking assignments update variables immediately, while non-blocking assignments schedule updates to happen at the end of the current time step.

C) Non-blocking assignments can cause race conditions, while blocking assignments prevent them.

D) There is no difference in behavior within a synchronous always block.

Question 1.3
Endianness primarily refers to:

A) The order in which bits are transmitted serially over a bus.

B) The order in which bytes (or larger data units) are stored in memory or transmitted over a bus.

C) The direction of data flow in a Verilog module's ports.

D) The phase relationship between multiple clock signals in a system.

Question 1.4
When designing a Finite State Machine (FSM) in Verilog, which part typically describes the conditions under which the FSM moves from one state to another?

A) The output logic.

B) The state register declaration.

C) The next-state logic.

D) The input port definitions.

Question 1.5
Which of the following is a conceptual building block found within a typical FPGA's internal structure, providing configurable logic capability?

A) CPU core

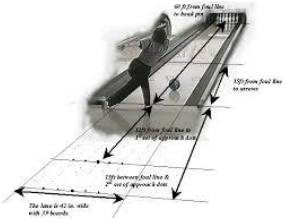
B) Hard disk controller

C) Look-Up Table (LUT)

D) Analog-to-Digital Converter (ADC)

Section 2: Long Answer Questions (60 Marks)

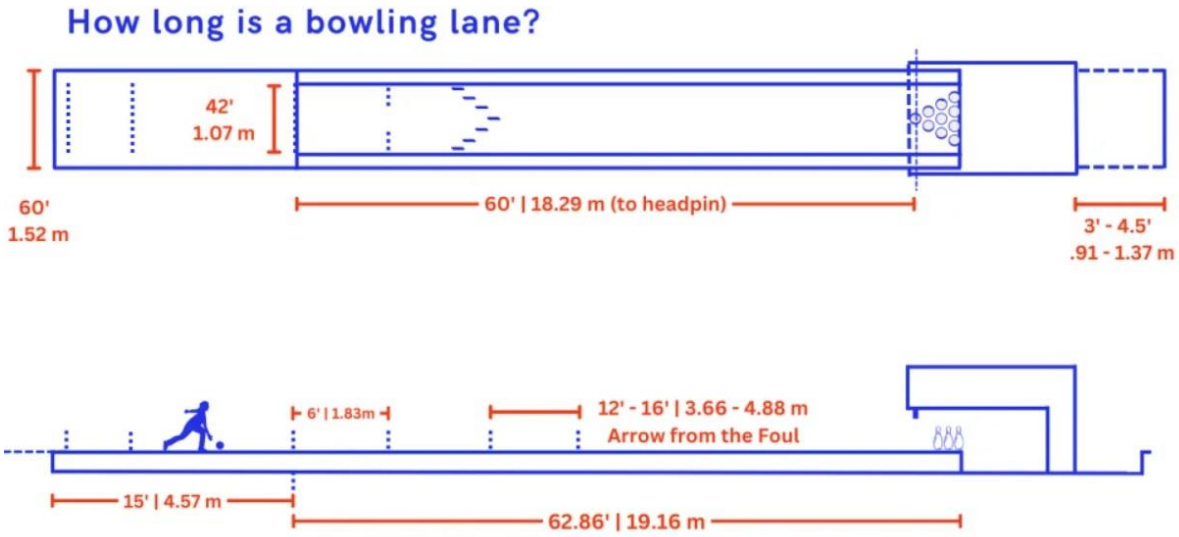
This section has two parts.



Introduction to the Bowling Lane Speed Indicator (BoLaSpI)

The BoLaSpI is a system designed to measure the speed of a bowling ball as it travels down a bowling lane. The system uses two ball detection sensors (bds_1 and bds_2) placed along the lane to time the ball's movement over a known distance. The system provides visual feedback via LEDs and displays the calculated speed.

The diagram below shows information about the dimensions of a bowling lane (that is more just for interest, because you just need to worry about the distance between the first sensor and the second sensor).



Question 2.1: BoLaSpI Verilog Module Implementation [45 Marks]

You are tasked with developing the Verilog code for the BoLaSpI module based on the description provided above, and the functionality of the system specified below. The following is the interface and starting point Verilog code for the module:

Verilog Module Interface:

```
module BoLaSpI (
    input wire clk,         // 1KHz clock (1ms period)
                            // consider using for the 10s display timer
    input wire fclk,        // 1MHz clock signal (1us period) - useful for
                            // more precise timing if needed
    input wire START,       // System activation button (posedge)
    input wire bds_1,       // First ball detection sensor (posedge if ball detected)
    input wire bds_2,       // Second ball detection sensor (posedge if ball detected)
    output reg START_LED,   // Indicates system is ready for a new ball
    output reg APPROACH_LED, // Indicates ball passed bds_1 and approaching bds_2
    output reg CALC_LED,    // Indicates speed calculation is in progress/complete
    output wire display_on, // Controls speed display module (1 turn on, 0 turn off)
    output wire [32:0] speed_out // Output for the calculated speed
);

// You may need to instantiate the udiv32 and display_speed modules here

// Your implementation goes here
endmodule
```

NOTE: Please use CODE ANSWER PAGE towards the end of this test paper to provide your answer to this question (the interface is repeated there for convenience)

Functionality Requirements Summary:

1. Initially, only START_LED is on.
2. When a positive edge on START is detected, the system becomes ready for a ball. START_LED remains on.
3. When a positive edge on bds_1 is detected (while in the ready state):
    - START_LED turns off.
    - APPROACH_LED turns on.
    - A high-resolution timer (using fclk) starts to measure the time until bds_2 is detected.
4. When a positive edge on bds_2 is detected (after bds_1 but before the end of the lane):
    - APPROACH_LED turns off.
    - CALC_LED turns on.
    - The high-resolution timer is stopped.
    - The speed is calculated using the measured time.
    - The distance between bds_1 and bds_2 is **3 feet** (0.91m).
    - Use the provided udiv32 module with the interface module udiv32(output[32:0] result, input[32:0] a, input [32:0] b); for the division. You will need to determine the appropriate values for inputs a and b to calculate speed in feet per second (fps). Remember to scale appropriately to use integer division. Speed in fps = Distance (ft) / Time (s). Time in seconds = Time in fclk cycles / 1,000,000.
    - The calculated speed should be output on speed_out.
    - The display_on signal should be asserted to turn on the external speed display module.
    - A 10-second timer (using clk) starts for the display duration.
5. After the 10-second display timer expires:
    - display_on is de-asserted.
    - CALC_LED turns off.
    - START_LED turns on.
    - The system returns to the initial state, waiting for a new bds_1 detection (after a START has been registered).

NB: First look at "Your Task" on next page, then (a hint) jot down rough code thoughts for this operation as you read through them to save time

Helper Modules (provided conceptually):
You will need to instantiate these modules in your BoLaSpI module, they will be useful in calculating and displaying the bowling ball speed.

```
// Unsigned Integer Divide Module
// result = a / b
module udiv32(
    output [32:0] result,
    input  [32:0] a,
    input  [32:0] b
);
    // Assume this module's internal implementation is correct
endmodule

// Speed Display Module
// on = 1 enables display, displays 'speed' value
// on = 0 disables display
module display_speed(
    input on,
    input [32:0] speed
);
    // Assume this module's internal implementation drives physical display
endmodule
```

Your Task – this is what to do for Question 2.1:

Read over the description and module interface.

1. **Approach, Calculations, and Pseudocode:** Briefly explain your approach to implementing the BoLaSpI logic. Indicate any distances and calculations you will use to determine the speed in feet per second (fps), including how you will handle the division using udiv32. Provide a short snippet of pseudocode outlining the main states and transitions of your system. This section can help demonstrate your understanding even if your Verilog code is incomplete.
2. **Verilog Code:** Provide the complete Verilog code for the BoLaSpI module, implementing the functionality described above. Make sure to include comments to explain your logic, as well-commented code will improve your mark.

Provide your code in the CODE ANSWER PAGE below.

Question 2.2: BoLaSpI Enhancement [15 Marks]

Consider an enhancement where the system checks if bds_2 is triggered *before* bds_1 in a given cycle. If this invalid sequence occurs, the system should indicate an error by displaying "XXX" for 1 second, using a hypothetical module call like display_txt("XXX"). After displaying "XXX" for 1 second, the system should return to the initial state (only START_LED on, waiting for START).
Explain conceptually how you would modify your BoLaSpI design to incorporate this error detection and indication. You do not need to provide Verilog code for this answer, but you can use code snippets to illustrate your explanation if helpful. Your answer should be a written explanation in English.

(answer Q2.2 on next page)

Q2.2:

end of test

Prep Test Opportunity 1

Q2.1 CODE ANSWER PAGE

Verilog Module Interface:

```
module BoLaSpI (
    input wire clk,         // 1KHz clock (1ms period)
                            // consider using for the 10s display timer
    input wire fclk,        // 1MHz clock signal (1us period) - useful for
                            // more precise timing if needed
    input wire START,       // System activation button (posedge)
    input wire bds_1,       // detection sensor 1 (posedge if ball detected)
    input wire bds_2,       // detection sensor 2 (posedge if ball detected)
    output reg START_LED,  // Indicates system is ready for a new ball
    output reg APPROACH_LED,  // Indicates ball passed bds_1 and approaching bds_2
    output reg CALC_LED,   // Indicates speed calculation is in progress/complete
    output wire display_on, // Controls speed display module (1=on, 0=off)
    output wire [32:0] speed_out // Output for the calculated speed
);

// Add your code below in the lines that follow!!
```

// continuation of Q2.1 answer ...

Prep Test Opportunity 1

MULTIPLE CHOICE ANSWER PAGE

Please put the answers for all the multiple choice questions of this test using the table below.

Put a tick in just one of the a, b, c, or d columns for each row corresponding a multiple choice question number.

Question #	A	B	C	D
1				
2				
3				
4				
5				