Please enter your name, Athena login name, and recitation section above. Enter your answers in the spaces provided below. Show your work for potential partial credit. You can use the extra white space and the backs of the pages for scratch work.

Problem 1. Combinational Circuits (17 points)

A fibbinary number is any number whose binary representation does not contain any sequences of adjacent 1’s. For example, 1 (0b001), 2 (0b010), and 5 (0b101) are fibbinary, but 3 (0b011) is not. Our goal is to design some combinational circuits that check whether a number is fibbinary.

(A) (3 points) You first want to implement a simple 2-bit fibbinary checker in hardware, but Logic Gates ‘R’ Us is sold out of every logic gate except 2-input muxes! Can we still implement a circuit that will output a 1 if a 2-bit input x is a fibbinary number and 0 otherwise using only 2-input muxes? If yes, draw the circuit using the minimum number of muxes. If no, explain why not.
(B) (3 points) Implement the function `isFibbinary3` in Minispec, which returns 1 if its 3-bit input `x` is a fibbinary number and 0 otherwise. For full credit, use only logical operators (NOT (~), AND (&), OR (|), XOR (^)).

```minispec
function Bit#(1) isFibbinary3(Bit#(3) x);
endfunction
```

(C) (5 points) Implement `isFibbinary#(n)` in Minispec using only a for loop and logical operators (NOT (~), AND (&), OR (|), XOR (^)). `isFibbinary#(n)` takes in an n-bit input `x` and returns 1 if it is a fibbinary number and 0 otherwise. Assume `n` is greater than or equal to 2. You may declare variables and return values outside of the for loop.

```minispec
function Bit#(1) isFibbinary#(Integer n)(Bit#(n) x);
    Bit#(1) isFib = 1;
    for (_Integer i = 0_; _ i < n - 1_; _ i = i + 1_) begin
        isFib = isFib & ~(x[i] & x[i + 1])
    end
    return isFib;
endfunction
```

(D) (2 points) How does the delay of `isFibbinary#(n)` grow with input width `n`?

The number of iterations through the for loop is proportional to `n`. 

Delay complexity: \( O(n) \)

(E) (4 points) Your friend points out that there’s a faster way to check for a fibbinary number if we don’t limit ourselves to using only logical operators. Complete the single-line implementation of `isFibbinary#(n)` using any combination of the following binary operators: &, |, ^, ~, <<, >>, ==, !=. *Hint:* Some shifting is involved.

```minispec
function Bit#(1) isFibbinary#(Integer n)(Bit#(n) x);
    return (__((x << 1) & x) == 0__)? 1'b1: 1'b0;
endfunction
```
Problem 2. Combinational and Sequential Logic Timing (15 Points)

Consider the sequential circuit below, as well as the timing specifications. Registers R1, R2, and R3 are driven by a common clock. A, B, and C are combinational circuits.

<table>
<thead>
<tr>
<th>Register (R1, R2, R3)</th>
<th>$t_{pd}$ (ns)</th>
<th>$t_{cd}$ (ns)</th>
<th>$t_{setup}$ (ns)</th>
<th>$t_{hold}$ (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1, R2, R3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Circuit A</td>
<td>3</td>
<td>?</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Circuit B</td>
<td>4</td>
<td>2</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Circuit C</td>
<td>6</td>
<td>2.5</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

(A) (3 points) What are $t_{pd}$ and $t_{cd}$ of this sequential circuit?

- $t_{pd}$ (ns): ______5_______
- $t_{cd}$ (ns): ______1_______

(B) (3 points) What is the smallest value for $t_{cd}$ of Circuit A that will allow this circuit to operate correctly? Show your work.

- $t_{cd,R1} + t_{cd,A} \geq t_{hold,R2}$
- $1 + t_{cd,A} \geq 2$
- $t_{cd,A} \geq 1$ns

- $t_{cd}$ of Circuit A (ns): ______1_______

(C) (4 points) What is the shortest clock period that can be used to drive all of the registers in the circuit? Show your work.

- $t_{clk} \geq t_{pd,R1} + t_{pd,A} + t_{pd,B} + t_{setup,R3}$
- $t_{clk} \geq 5 + 3 + 4 + 7$
- $t_{clk} \geq 19$ns

- $t_{clk}$ of circuit (ns): ______19_______
The table above shows the original specs of our circuit. We now find that our supplier has the following alternative circuits for A, B and C available with the following specifications.

(D) (5 points) These new circuits aren’t cheap, so you may only replace one circuit in order to minimize clock period. Please indicate which combinational circuit you are replacing, and the resulting minimum clock period. Explain why your selected component is a better choice than the other two. You should explicitly compare all three components in your explanation.

Combinational circuit replaced: _____B_______

t_{clk} of circuit (ns): _____18_______

Explanation:
To minimize $t_{clk} \geq t_{pd,R1} + t_{pd,A} + t_{pd,B} + t_{setup,R3}$ we want to replace A or B. Can’t replace A with A-New because contamination delay of 0.5 would not satisfy the hold time for R2. So, we want to replace B since it is the other component in the critical path. Replacing B reduces the R1-R3 path to $5 + 3 + 2.5 + 7 = 17.5$ns. However, the critical path is now the R2-R1 path which is $5 + 6 + 7 = 18$. 

\[
t_{pd} \quad t_{cd} \quad t_{setup} \quad t_{hold}
\]

<table>
<thead>
<tr>
<th>ORIGINAL</th>
<th>$t_{pd}$</th>
<th>$t_{cd}$</th>
<th>$t_{setup}$</th>
<th>$t_{hold}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register (R1, R2, R3)</td>
<td>5ns</td>
<td>1ns</td>
<td>7ns</td>
<td>2ns</td>
</tr>
<tr>
<td>Circuit A</td>
<td>3ns</td>
<td>?</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Circuit B</td>
<td>4ns</td>
<td>2ns</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Circuit C</td>
<td>6ns</td>
<td>2.5ns</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

\[
| A-New                   | 1ns      | 0.5ns    | --          | --         |
| B-New                   | 2.5ns    | 2ns      | --          | --         |
| C-New                   | 2ns      | 1ns      | --          | --         |
Problem 3. Finite State Machine (15 points)

Rettiwt Complaint Hotline Operator is implementing a new message transfer protocol to help him detect corrupted messages:

- Each message will have 3 bits.
- The first two bits are data bits.
- The third bit is a parity bit that should make the total number of 1’s (including the parity bit) in the message be odd.
- Messages are sent bit by bit.

With this specification, a message is valid if and only if it contains an odd number of 1’s. For example, 111 and 010 are valid messages, but 101 is an invalid message.

The output of the FSM is set to 1 for one clock cycle on the next rising edge of the clock after an incorrect parity bit is received. The FSM should output a 0 at all other times. The bit received immediately after the parity is treated as the beginning of a new message regardless of the correctness of the previous message.

(A) (6 points) We provide a partially complete state transition diagram. Each circle corresponds to the state the FSM is currently in. The Hotline Operator provided annotations for some of the states. Fill in the missing inputs corresponding to each of the arrows connecting the states. The FSM begins at state 0 (Valid Message).
(B) (3 points) Fill in the missing data in the following **truth table** for this finite state machine. Use the integers in the FSM states to identify the current and next state.

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(C) (6 points) Assume your FSM begins at state 0 before processing each of the following messages. What is the state and output of the FSM on the next rising edge of the clock **after receiving the last bit** of each message?

1. 001_011_111_10  
   Output: _____0____  State: _____5_____  
2. 110_010_010_110_011  
   Output: _____1____  State: _____1_____  
3. 110_110_100_111_0  
   Output: _____0____  State: _____2_____
Problem 4. Sequential Circuits in Minispec (20 points)

You are frustrated with the 77 Mass. Ave crosswalk and decide to design a better traffic signal in Minispec. To start, you want to make sure the traffic light will function well in the daytime when there’s lots of traffic. After carefully analyzing traffic patterns, you define the following specification:

- The traffic light should be red for 4 cycles, then green for 10 cycles, then yellow for 1 cycle, and repeat this pattern indefinitely.
- The light starts red (and should stay red for 4 cycles before turning green).
- Pedestrians can only cross when the light is red.

(A) (6 points) Fill in the Minispec module on the next page to track the traffic light state as a sequential circuit.

- The pedestriansCanCross method should return True if and only if the light is in a state where pedestrians are allowed to cross.
- The currentLight method should return the current state of the traffic light.
- We have provided a counter register – use this to count down to the next state transition.
typedef enum { Green, Yellow, Red } LightState;

module TrafficLight;

    Reg#(LightState) light(__Red__);  
    Reg#(Bit#(____4____)) counter(____3____);

    method Bool pedestriansCanCross = _light == Red_;  
    method LightState currentLight = ___light____;

rule tick;

    if (light == Green) begin
        if (counter == 0) begin
            light <= ___Yellow____;
        end else begin
            counter <= __counter - 1__;
        end
    end else if (light == Yellow) begin
        light <= ___Red____;
        counter <= _____3____;
    end else if (light == Red) begin
        if (counter == 0) begin
            light <= ___Green___;
            counter <= ___9____;
        end else begin
            counter <= __counter - 1__;
        end
    end
endrule
endmodule
(B) (6 points) To ensure that your module behaves as expected, fill in the timing chart below with the register values and outputs for the first 6 cycles.

<table>
<thead>
<tr>
<th>Cycle</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>light</td>
<td>Red</td>
<td>Red</td>
<td>Red</td>
<td>Red</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>currentLight</td>
<td>Red</td>
<td>Red</td>
<td>Red</td>
<td>Red</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>pedestriansCanCross</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

(C) (8 points) Now you want to add a new feature to your traffic light. During the daytime, you want it to work as in part (A). But during the nighttime, the traffic light should work differently:
- By default, the light should be green.
- When a pedestrian requests to cross the street and the light is green, it should remain green for 3 more cycles, turn yellow for 1 cycle, then red for 4 cycles. Then it should go back to being green indefinitely.
- If a pedestrian requests to cross the street when the light is yellow or red, this request should be ignored and have no effect.
- If a pedestrian requests to cross the street while the light is green, and a pedestrian requests to cross the street in a following cycle when the light is still green, this request should also have no effect.

You also want to add a feature for emergency pedestrian requests. In an emergency, if a pedestrian requests to cross, the light should immediately turn yellow on the next cycle. The pedestrian request is provided as a Maybe#(PedestrianRequest) type – on each cycle it will either be:
- Invalid (no pedestrian request)
- Standard (a standard pedestrian request was made)
- Emergency (an emergency pedestrian request was made)

Note that your implementation should still work when the input transitions from daytime to nighttime, even though in daytime the Green light is 10 cycles and in nighttime it is only 3 cycles following a pedestrian request. Thus, if it is nighttime and our counter variable is too large (because we were counting down from a larger value during the daytime), we should “clamp” it to be no larger than it can be in nighttime. We have provided a currentCounter variable to use for this purpose – i.e. it will be clamped to the maximum value the counter can be during nighttime.

Fill in the Minispec module below to add this functionality. We have provided two inputs – one for whether it is currently nighttime or daytime, and one for whether a pedestrian has requested to cross the street in this cycle.
typedef enum { Green, Yellow, Red } LightState;
typedef enum { Daytime, Nighttime } TimeOfDay;
typedef enum { Standard, Emergency } PedestrianRequest;

module TrafficLight;

Reg#(LightState) light(_<answer from Part A>_);
Reg#(Bit#(_<answer from Part A>_)) counter(_<answer from part A>_);
input TimeOfDay timeOfDay default = Nighttime;
input Maybe#(PedestrianRequest) pedestrianRequest default = Invalid;
method Bool pedestriansCanCross = <answer from part A>;
method LightState currentLight = <answer from part A>;

The code continues on the next page.
rule tick;
    if (timeOfDay == Daytime) begin
        <Your answer from Part A>
    end else begin
        if (light == Green) begin
            Bit#(<answer from part A>) currentCounter;
            // Clamp currentCounter to the maximum value counter
            // can be for a Green light at night
            currentCounter = counter > ___3__ ? ___3__ : counter;
            if (currentCounter == 0) begin
                light <= ___Yellow___;
            // Check if received pedestrian request this cycle
            end else if (___isValid(pedestrianRequest)___) begin
                // Handle emergency request
                if (fromMaybe(?., pedestrianRequest) == Emergency) begin
                    light <= ___Yellow________________
                end else begin
                    counter <= ___currentCounter - 1___;
                end
                end else if (currentCounter < ___3____) begin
                    counter <= ___currentCounter - 1____;
                end else begin
                    counter <= ___currentCounter_(or 3)___;
                end
            end else if (light == Yellow) begin
                <Your answer from Part A>
            end else if (light == Red) begin
                <Your answer from Part A>
            end
        end
        endrule
endmodule
Problem 5. Arithmetic Pipelines (17 points)

You are given a module, named “F.” This module has two inputs, X and Y, and two outputs, A and B. You are told that the circuit functions, but its throughput is too low. You decide to take a look and try to pipeline the circuit.

For each of the questions below, please create a valid K-stage pipeline of the given circuit. Each component in the circuit is annotated with its propagation delay in nanoseconds. Show your pipelining contours and place large black circles (●) on the signal arrows to indicate the placement of pipeline registers. Give the latency and throughput of each design, assuming ideal registers (t_{PD}=0, t_{SETUP}=0). Remember that our convention is to place a pipeline register on each output.

(A) (1 point) What is the propagation delay of the whole circuit shown below as-is without pipelining?

\[ t_{PD} \text{ (ns)}: \boxed{26} \]

(B) (4 points) Show the maximum-throughput 3-stage pipeline using a minimal number of registers. What are the latency and throughput of the resulting circuit? Pay close attention to the direction of each arrow. In case you need them, extra copies of the circuit are available on the last page of the exam.

\[ \text{Latency (ns)}: \boxed{33} \]
\[ \text{Throughput (ns}^{-1}\text{)}: \boxed{1/11} \]
(C) (4 points) Show the maximum-throughput pipeline using a minimal number of registers. What are the latency and throughput of the resulting circuit? In case you need them, extra copies of the circuit are available on the last page of the exam.

Latency (ns): __32____

Throughput (ns⁻¹): __1/8____

(D) Now, you are given two new modules: module “G” takes in input U and produces output V, and module “H” takes in input S and produces output T. You are given module “G” implemented with a 2-stage pipeline, with registers denoted by the large black circles (●), and module “H” implemented with a single stage pipeline as shown below.
(i) (2 points) Given the implementation of the modules above, what are the throughputs of the modules?

Throughput of G(ns⁻¹): ___1/15__

Throughput of H(ns⁻¹): ___1/8____

You want to connect these two modules with the module “F”, from the previous parts of this problem, so the output A of Module “F” is connected to the input U and output B is connected to the input S as shown below.

![Diagram of modules F, G, H, and V connected with inputs X, Y, and outputs Out1 and Out2.]

(ii) (2 points) Do any changes need to be made to modules G or H to ensure that the combined circuit above behaves as a proper pipeline? If so, draw any updated modules below. If no changes are required, say “No changes required”.

Add a pipeline stage to H so that both G and H have 2 pipeline stages. (Preferred)

OR

Remove the first pipeline stage on G (between 6 and 3/7)

(iii) (4 points) When connecting module F to module G and module H, you have two options for module F: your 3-stage pipeline, or your maximum-throughput pipeline. If you want to maximize throughput, while minimizing latency and the number of registers used, which implementation of module F would you use? What would the latency and throughput of the combined device be? Assume that any required changes to modules G or H have been made.

Module F implementation (circle one): 3-stage pipeline  Maximum-throughput pipeline

5 Stages at 15ns/stage (if gave 1st answer to dii) or 4 Stages at 26ns/stage (if gave 2nd answer to dii) Latency (ns): ___75 or 104____

Throughput (ns⁻¹): ___1/15 or 1/26____
Problem 6. A RISCier processor (16 points)

Consider the single-cycle processor implementation we saw in lecture:

The timing characteristics of all components are listed below:

<table>
<thead>
<tr>
<th>Component</th>
<th>Propagation delay (trD)</th>
<th>Setup/hold times for clocked inputs (registers and writes to RegFile and data memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>1ns</td>
<td>Setup time (t_{SETUP}) 2 ns</td>
</tr>
<tr>
<td>Decoder</td>
<td>2ns</td>
<td>Hold time (t_{HOLD}) 0 ns</td>
</tr>
<tr>
<td>RegFile read</td>
<td>3ns</td>
<td></td>
</tr>
<tr>
<td>MUX</td>
<td>1ns</td>
<td></td>
</tr>
<tr>
<td>ALU</td>
<td>4ns</td>
<td></td>
</tr>
<tr>
<td>Adder</td>
<td>3ns</td>
<td></td>
</tr>
<tr>
<td>Memory read</td>
<td>4ns</td>
<td></td>
</tr>
</tbody>
</table>

Assume that any components not listed have a delay of 0 ns.

(A) (3 points) What is the minimum clock cycle time of this processor? (For partial credit, draw the critical path in the diagram above.)

Critical path is PC -> IMem -> Decoder -> RF(read) -> BSEL mux -> ALU -> DMem (read) -> WDSEL mux -> RF(write) => t_{CLK} >= 1 + 4 + 2 + 3 + 1 + 4 + 4 + 1 + 2 (RF setup)

Minimum t_{CLK}: 22 ns
Ben Bitdiddle is unhappy with the performance of this processor. After a Ouija session with legendary CPU designer Seymour Cray, Ben implements this alternative datapath (the control logic stays the same), where the data memory’s Adr input comes from a different place:

(B) (3 points) What is the minimum clock cycle time of Ben’s new processor? (For partial credit, draw the critical path in the diagram above.)

Critical path is PC -> IMem -> Decoder -> RF(read) -> BSEL mux -> ALU -> WDSEL mux -> RF(write) => t_{CLK} >= 1 + 4 + 2 + 3 + 1 + 4 + 1 + 2 (RF setup)

Minimum t_{CLK}: ___18____ ns

(C) (2 points) Ben’s processor executes some instructions incorrectly according to the RISC-V ISA. Give an example of one such instruction, and write the equivalent RISC-V instruction that is actually executed by the processor.

Example of incorrect instruction: ___lw a0, 8(a1)_________

(any lw or sw instruction with a non-zero offset)

RISC-V instruction that produces the same behavior as executing the above incorrect instruction: ___lw a0, 0(a1)_________

(the same lw or sw instruction with the offset set to 0)
(D) (5 points) The program below takes 90 instructions to execute in the original processor. However, it produces incorrect results in Ben’s new processor. Modify the program so that it runs correctly on the new processor. For full credit, the number of executed instructions should not increase compared to the original code. How many instructions does your assembly code execute?

**C code**

```c
int x[16];
for (int i = 0; i < 15; i++)
    x[i+1] = x[i] + x[i+1];
```

**Assembly code**

```assembly
# Initial values:
# a0: address of x[0]
# a7: address of x[15]
loop:  lw a1, 0(a0)
       lw a2, 4(a0)
       add a3, a2, a1
       sw a3, 4(a0)
       addi a0, a0, 4
       blt a0, a7, loop
```

**Modified assembly that executes correctly on new processor:**

```assembly
# Initial values:
# a0: address of x[0]
# a7: address of x[15]
loop:  lw a2, 0(a0)
       add a1, a1, a2
       sw a1, 0(a0)
       addi a0, a0, 4
       ble a0, a7, loop
```

Number of executed instructions of new program: \(_{15\times5 + 2 = 77}\)  
(other solutions are possible; any solution with fewer instructions than the original one earns full credit, and any correct solution earns at least 2 points of partial credit)

(E) (2 points) What is the execution time of the above program in the original and new processors? (Use the appropriate variant of the program for each processor.)

Execution time on original processor: \(_{77\times22 = 1694\text{ ns}}\)

Execution time on Ben’s new processor: \(_{77\times18 = 1386\text{ ns}}\)

(or consistent with D)

END OF QUIZ 2!
Extra copies of pipeline diagram for problem 5: