Lab 05 Pre lab – Worksheet

Name: Shaheer Qureshi, Areeba Izhar	ID: sq09647, ai09625	Section: T1

Exercise

Exercise 1: Identify Synthesizability

```
always @(*) begin
  for (i = 0; i < n; i = i + 1)
    sum = sum + a[i];
end
```

Not Synthesizable (unless n is a fixed constant number).

When writing a code in computer, it can repeat a task n time, where n can be any number.

When we write code for a FPGA, the synthesizer has to physically build the circuit using logic gates.

If n is a variable (like an input signal): The synthesizer needs to know how many adders should it build? 5? 100? 1000?

Since n can change while the FPGA is running, the hardware would have to physically grow or shrink to match n. Hardware cannot shapeshift. Therefore, it fails.

The only exception is if n is defined as a Parameter (fixed constant like parameter n = 8;) before this block, then it is Synthesizable.
This way the synthesizer knows exactly how many copies to make. It loop into a fixed chain of n adders.

If Loop limit is a number (e.g., 8): Good. (Hardware is fixed).
If Loop limit is a variable (e.g., n): Bad. (Hardware size is unknown).

Exercise

Exercise 2: Fix the Code

```
always begin
  #10 clk = ~clk;
end
```

```
module xyz (
   input wire clk   // Ask for the clock as an input!
);
   // Use the 'clk' here, don't create it.
   always @(posedge clk) begin
      // Do work here
   end
endmodule
```

In Simulation: The computer pauses the "game" for 10ns. This is doable
In Real Hardware: Cannot tell electricity to "stop moving" for 10ns. There is no component inside the FPGA that counts time by itself like this so, pass the clock in as an input wire from the board.

Exercise

Exercise 3: What hardware does this code describe?

```
always @(posedge clk) begin
  if (en)
    q <= d;
```

D Flip-Flop with an Enable Signal.

Memory that holds one number.

1. always @(posedge clk) **Trigger**
   - This is the Lock. The memory only opens for a split second when the "Clock" ticks (goes from Low to High). At all other times, it is locked tight.
2. if (en) **Permission**
   - This is "Enable".
   - Even if the Clock ticks, enable checks this signal first.
   - If en is OFF (0): It blocks. The memory ignores the new number and keeps the old number.
   - If en is ON (1): The grants permission.
3. q <= d **Action**
   - d (Data Input): This is the new number to be stored.
   - q (Output): This is the number currently stored inside.
   - Action: If the Clock ticks AND Enable is On, the memory takes the value from d and stores it in q.