Instructions

- The **skill test** will be conducted **only in the simulator**.
- You will be provided with **a set of testbenches**.
- The exam is **closed book**—you may **only use the code provided in the test**.
- You can use any **Programming language** to generate a file to initialize RAM/ROM value.

Problem Statement

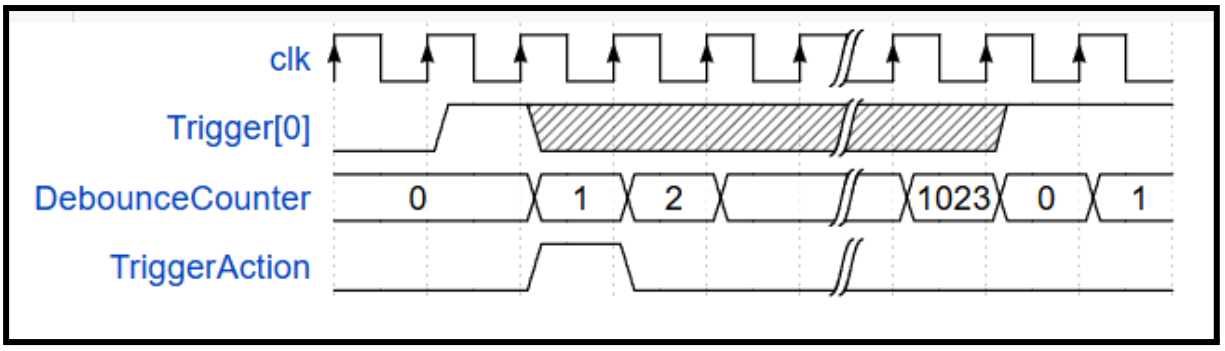
You will implement a **Verilog module** named `skilltest1` with the following functionality:

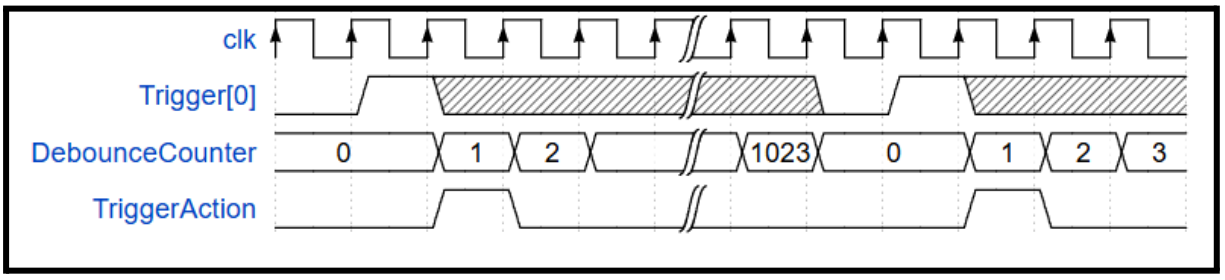
Module Declaration

```
module skilltest1 (
   input  wire        Clk,
   input  wire        Reset,
   input  wire [3:0]  Trigger,
   output wire [3:0]  BCD0,
   output wire [3:0]  BCD1,
   output wire [3:0]  BCD2,
   output wire [3:0]  BCD3
);
```

Inputs

- `Clk`: A well-behaved clock signal.
- `Reset`: A reset signal.
- `Trigger`: A **4-bit signal** that may contain **bouncing noise**. If multiple rising/falling signals occur **within 1024 clock cycles**, they are considered **a single button press**. If the trigger is asserted and remains asserted at the end of the 1024-cycle debounce period—even if it toggles intermittently—it is considered a long press that generates only one trigger action. Conversely, if the trigger is deasserted for at least one clock cycle after the debounce period and then reasserted, it is treated as a new button press, resulting in a separate trigger action.





DebounceCounter : Count how many clock cycles from the first asserted Trigger. (for 1024 range)

TriggerAction : The Trigger Action happens. See the below for what you have to implement for each Trigger[x].

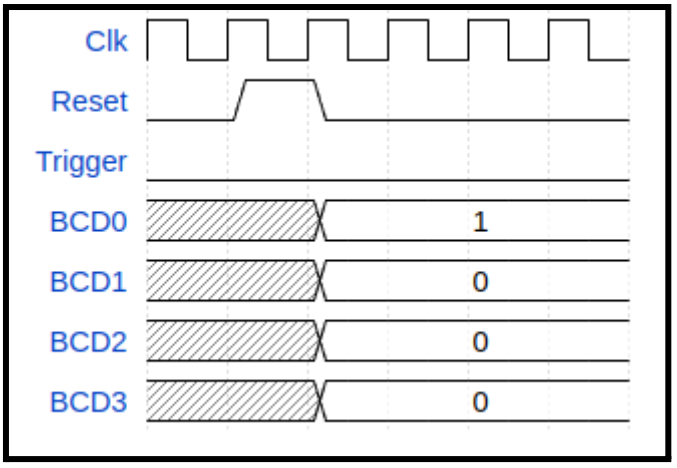
Note : The above timing diagram is only for demonstration purposes, your design may have more delay than this timing diagram.

Outputs

- `BCD3`, `BCD2`, `BCD1`, `BCD0`: **Binary Coded Decimal (BCD) representation** of a number.
  - Example: **1234 (decimal)** →
    - `BCD3 = 4'b0001`
    - `BCD2 = 4'b0010`
    - `BCD1 = 4'b0011`
    - `BCD0 = 4'b0100`

Module Behavior

- **Reset (`Reset = 1`)**:
  - The BCD value is set to **1** (`BCD0 = 1`, `BCD1 = BCD2 = BCD3 = 0`). The reset is **synchronous**.



- **Trigger Actions (the result output BCD should be correct within 4 rising clock edges)**:
  - `Trigger[0]`: **Increment BCD by 1**.
  - `Trigger[1]`: **Increment BCD by 2**.
  - `Trigger[2]`: **Multiply BCD by 2**.
  - `Trigger[3]`: **Multiply BCD by 11**.
  - Also there is going to be only one Trigger asserted at a time.
- **Overflow Handling**:
  - If the **BCD value exceeds 9999**, all of the outputs (`BCD0`, `BCD1`, `BCD2`, `BCD3`) will be set to **4'b1111** until reset.

Grading Criteria

Criteria	Weight
1.Reset behavior is correct	10%
2.Trigger[0] functions correctly	15%
3.Trigger[1] functions correctly	15%
4.Trigger[2] functions correctly	15%
5.Trigger[3] functions correctly	15%
6.Module correctly handles overflow	15%
7.Module handles continuous trigger signals	10%
8.Module handles button noise correctly	5%

- For Criteria 2–6, the trigger signal is free from bouncing noise. Therefore, you may complete this requirement without implementing any debouncing mechanism or single-pulse circuit.

Note

Using RAM/ROM with initialized data

When using the function to initialize RAM/ROM data from files, you must provide the exact (absolute) file path. For example: `C:/Users/JohnDoe/Documents/rom.mem`.

Some useful syntax/hint

- Initializing RAM/ROM data

```
$readmemb(”<absolute path to your .mem file>”, memory_array);
```

- Code for converting decimal integer to binary in Python

```
file = "<file name to write data to>.mem"
with open(file, "w") as f:
    for i in range (10):
        for j in [8,4,2,1]:
            f.write("1" if i & j else "0")
    f.write("\n")
```

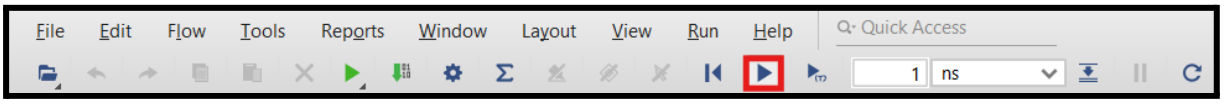
- Generate syntax in Verilog

```
generate
    genvar i; // variable to be use in this scope
    for (i = 0; i < 4; i = i + 1) begin
      Module1 Module1Inst (
          .Clk(Clk),
          .Reset(Reset),
          .DataIn(DataIn[i]),
          .DataOut(DataOut[i])
      );
    end
endgenerate
```

- Observe the criteria weight.

Running testbench

In order to grade the result, you must run a testbench in Vivado which consists of 8 criterias. After running the simulation, **press F3** or **click the run button** shown in the picture below to simulate all criterias (It may take up to 3 minutes). The grading result will be displayed in the TCL console window.



The grading result of testbench is shown in 8 letters, representing each criteria respectively. Each criteria **can also have various test cases**.

- **X** means the code **provides incorrect output** for some of the test cases in that criteria.
- **P** means the output is **correct** in that criteria.

After you correctly implement necessary modules, the final result should be as shown in the picture below.

```
run all
Finished test cases
Test Results:
PPPPPPPP
Total Score: 100/100
```