Academic Dishonesty

ECE 351

Homework 7

Spr 2026

Gmail Garrison Greenwood <greenwd@pdx.edu>

ECE351 - Possible Academic Dishonesty

Hector Cardenas <hector7@pdx.edu> Wed, 10 Jun at 00:02
To: Garrison Greenwood <greenwd@pdx.edu>

Hi Dr. Greenwood,

While grading I found a couple of discrepancies between two students' submissions that I wanted to bring to your attention. They both involve Saoud Alkhamis and Sophia Peters.

First, the more obvious one: On HW7 they had nearly identical testbenches, I've attached a screenshot of the `diff` here for your own reference.

```
hcardenas@Hectors-MacBook-Pro plagerism % diff -w ./alkhamissaoud_25409_16719844_hw7_tb.v peterssophia_97975_16719828_tb_fifo.v | bat
```

	STDIN
1	1,3d0
2	< //Saoud Alkhamis
3	< //ECE 351
4	< // HW7 Testbench
5	48,49c45
6	< #250;
7	< $finish;
8	---
9	> #250 $finish;
10	82c78
11	< @(posedge clk_sig) begin rd_en = 1; end
12	---
13	> @(posedge clk_sig) begin rd_en = 1; end // Hold read enable for 2nd cycle

Additionally, their FIFO modules also had a high amount of structural similarity, with only a few variable names changed; I ran a script that compared all non-alpha characters and got a 97% similarity score (the same script returned less than ~70% for most cross-student comparisons).

The next discrepancy is in their submission for HW6. Both videos appear to show the same FPGA board and are taken 1 minute and 43 seconds apart. I think this is *technically* possible if you either use a TCL script or only complete the programming part between takes (preparing both bit-streams beforehand). I'll leave it to your discretion whether this is not worth considering, warrants further investigation, or just deserves a 0 score given the plagiarism between the students in HW7.

Best regards,
[Quoted text hidden]

```
hcardenas@Hectors-MacBook-Pro plagerism % diff -w ./alkhamissaoud_25409_16719844_hw7_tb.v peterssophia_97975_16719828_tb_fifo.v | bat
```

```
       STDIN
   1   1,3d0
   2   < //Saoud Alkhamis
   3   < //ECE 351
   4   < // HW7 Testbench
   5   48,49c45
   6   <             #250;
   7   <             $finish;
   8   ---
   9   >             #250 $finish;
  10   82c78
  11   <         @(posedge clk_sig) begin rd_en = 1; end
  12   ---
  13   >         @(posedge clk_sig) begin rd_en = 1; end // Hold read enable for 2nd cycle
```

Gmail Garrison Greenwood <greenwd@pdx.edu>

Re: ECE 351 HW7

Saoud Alkhamis <saoud4@pdx.edu> Wed, 10 Jun at 19:57
To: Garrison Greenwood <greenwd@pdx.edu>
Cc: Sophia Peters <psophia@pdx.edu>

Hi Professor Greenwood,

I hope this email finds you well. I apologize for the inconvenience this homework caused for you. We did collaborate on this homework, but we did not copy from each other. I am trying my hardest to pass this course or to obtain the best grade that I can, because even a small amount of points will have a big effect on my future. Is there a way to fix this problem? I could redo the assignment or complete some extra assignment to get back some points. Thank you for your consideration.

Regards,
Saoud Alkhamis

On Wed, Jun 10, 2026 at 10:24 AM Garrison Greenwood <greenwd@pdx.edu> wrote:

> I have a strict policy regarding copying homework (see syllabus). On HW7 there is a remarkable similarity between your submissions. Specifically
>
> 1. The two testbenchs are virtually identical. See attachment for differences.
>
> 2. Your FIFO source code shows a high degree of structural similarity. Indeed, with the exception of some variable names the similarity score, excluding all non-alpha characters, was about 97%. By comparison, cross checking between other students was around 70%.
>
> Such a high degree of similarity occurring by chance does not seem likely. However, I will give you the benefit of the doubt and allow you to explain how your code—which was supposed to be individually written—turned out to be so similar.
>
> Your HW7 homework score will be 0 until I hear from you.
>
> ———————-
> Garrison Greenwood, Ph.D.
> Associate Professor
> Portland State University

```
//Saoud Alkhamis
//ECE 351
// HW7
`timescale 1ns / 1ps

module fifo (
    input wire [7:0] din,
    input wire w_clk,
    input wire r_clk,
    input wire wen,
    input wire ren,
    input wire rst,
    output reg [7:0] dout,
    output wire empty,
    output wire full
);
    reg [7:0] buffer_mem [0:7];
    reg [3:0] wr_idx;
    reg [3:0] rd_idx;

    always @(posedge w_clk) begin
        if (rst) begin
            wr_idx <= 4'd0;
        end else if (wen && !full) begin
            buffer_mem[wr_idx[2:0]] <= din;
            wr_idx <= wr_idx + 1'b1;
        end
    end

    always @(posedge r_clk) begin
        if (rst) begin
            rd_idx <= 4'd0;
            dout <= 8'd0;
        end else if (ren && !empty) begin
            dout <= buffer_mem[rd_idx[2:0]];
            rd_idx <= rd_idx + 1'b1;
        end
    end

    assign empty = (wr_idx == rd_idx);
    assign full  = (wr_idx[2:0] == rd_idx[2:0]) && (wr_idx[3] ≠
rd_idx[3]);

endmodule
```

HW7, V

hw7_tb.v

```
//Saoud Alkhamis
//ECE 351
// HW7 Testbench
`timescale 1ns / 1ps

module tb_fifo;

    reg [7:0] tb_din;
    reg clk_sig;
    reg wr_en;
    reg rd_en;
    reg rst_sig;
    wire [7:0] tb_dout;
    wire f_empty;
    wire f_full;

    fifo uut (
        .din(tb_din),
        .w_clk(clk_sig),
        .r_clk(clk_sig),
        .wen(wr_en),
        .ren(rd_en),
        .rst(rst_sig),
        .dout(tb_dout),
        .empty(f_empty),
        .full(f_full)
    );

    task setup_sys;
        begin
            rst_sig = 1;
            wr_en = 0;
            rd_en = 0;
            tb_din = 8'h00;
            #15 rst_sig = 0;
        end
    endtask

    task make_clk;
        begin
            clk_sig = 0;
            forever #5 clk_sig = ~clk_sig;
        end
    endtask
```

```
task end_sim;
    begin
        #250;
        $finish;
    end
endtask

task SIM;
    begin
        fork
            setup_sys;
            make_clk;
            end_sim;
        join
    end
endtask

initial begin
    SIM;
end

initial begin
    #20;

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
    @(posedge clk_sig) begin wr_en = 0; end

    @(posedge clk_sig) begin rd_en = 1; end
    @(posedge clk_sig) begin rd_en = 0; end

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hFF; end
    @(posedge clk_sig) begin wr_en = 0; end

    @(posedge clk_sig) begin rd_en = 1; end
    @(posedge clk_sig) begin rd_en = 1; end
    @(posedge clk_sig) begin rd_en = 0; end

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hFF; end
```

```
        @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
        @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
        @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
        @(posedge clk_sig) begin wr_en = 0; end

        @(posedge clk_sig) begin rd_en = 1; end
        @(posedge clk_sig) begin rd_en = 0; end
    end

endmodule
```

fifo.v

peters

```
`timescale 1ns / 1ps

module fifo (
    input wire [7:0] din,
    input wire w_clk,
    input wire r_clk,
    input wire wen,
    input wire ren,
    input wire rst,
    output reg [7:0] dout,
    output wire empty,
    output wire full
);

    reg [7:0] mem [0:7];
    reg [3:0] w_ptr;
    reg [3:0] r_ptr;

    always @(posedge w_clk) begin
        if (rst) begin
            w_ptr <= 4'b0000;
        end else if (wen && !full) begin
            mem[w_ptr[2:0]] <= din;
            w_ptr <= w_ptr + 1;
        end
    end
    always @(posedge r_clk) begin
        if (rst) begin
            r_ptr <= 4'b0000;
            dout <= 8'h00;
        end else if (ren && !empty) begin
            dout <= mem[r_ptr[2:0]];
            r_ptr <= r_ptr + 1;
        end
    end
    assign empty = (w_ptr == r_ptr);
    assign full  = (w_ptr[2:0] == r_ptr[2:0]) && (w_ptr[3] ≠
r_ptr[3]);

endmodule
```

```
`timescale 1ns / 1ps

module tb_fifo;

    reg [7:0] tb_din;
    reg clk_sig;
    reg wr_en;
    reg rd_en;
    reg rst_sig;
    wire [7:0] tb_dout;
    wire f_empty;
    wire f_full;

    fifo uut (
        .din(tb_din),
        .w_clk(clk_sig),
        .r_clk(clk_sig),
        .wen(wr_en),
        .ren(rd_en),
        .rst(rst_sig),
        .dout(tb_dout),
        .empty(f_empty),
        .full(f_full)
    );

    task setup_sys;
        begin
            rst_sig = 1;
            wr_en = 0;
            rd_en = 0;
            tb_din = 8'h00;
            #15 rst_sig = 0;
        end
    endtask

    task make_clk;
        begin
            clk_sig = 0;
            forever #5 clk_sig = ~clk_sig;
        end
    endtask

    task end_sim;
        begin
```

tb_fifo. ✓

Peters

```
        #250 $finish;
    end
endtask

task SIM;
    begin
        fork
            setup_sys;
            make_clk;
            end_sim;
        join
    end
endtask

initial begin
    SIM;
end

initial begin
    #20;

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
    @(posedge clk_sig) begin wr_en = 0; end

    @(posedge clk_sig) begin rd_en = 1; end
    @(posedge clk_sig) begin rd_en = 0; end

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hFF; end
    @(posedge clk_sig) begin wr_en = 0; end

    @(posedge clk_sig) begin rd_en = 1; end
    @(posedge clk_sig) begin rd_en = 1; end // Hold read enable
for 2nd cycle
    @(posedge clk_sig) begin rd_en = 0; end

    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hFF; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h00; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'h55; end
    @(posedge clk_sig) begin wr_en = 1; tb_din = 8'hAA; end
```

```
        @(posedge clk_sig) begin wr_en = 0; end

        @(posedge clk_sig) begin rd_en = 1; end
        @(posedge clk_sig) begin rd_en = 0; end
    end

endmodule
```