Name: Kerolos Youssef Ghobrial

LAB 1: Linting

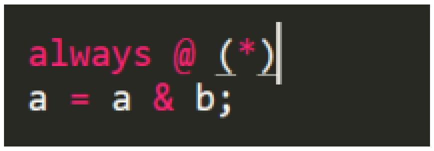
Q1.

The major violations that linting tool can detect are:

- Non-Synthesizable constructs
- Unintentional latches
- Unused declarations
- Multiple drivers and undriven signals
- Race conditions
- Incorrect usage of blocking and non-blocking assignments
- Incomplete assignments in subroutines
- Case statement style issues
- Set and reset conflicts
- Out-of-range indexing

Q2.

A-



```
always @ (*)
a = a & b;
```

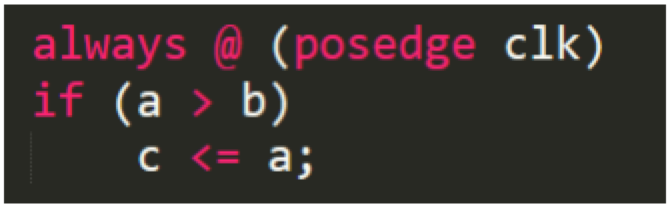
Error: combinational loop

Solution: break the loop by a flip-flop as follows:

```
module Q2_a(output reg a, input b,clk,rst);
always@(posedge clk,negedge rst)
begin
if(~rst)
a <= 0;
```

```
else
a <= a & b;
end
endmodule
```

B-



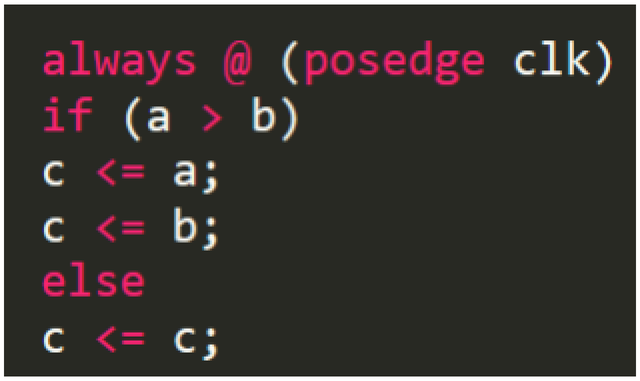
```
always @ (posedge clk)
if (a > b)
    c <= a;
```

Error: flip-flop should have an asynchronous set or reset

Solution: put asynchronous reset for flip-flop

```
module Q2_b(output reg c, input a,b,clk,rst);

always@(posedge clk,negedge rst)
begin
if(~rst)
c <= 0;
else if(a > b)
c <= a;
end

endmodule
```

C-



```
always @ (posedge clk)
if (a > b)
c <= a;
c <= b;
else
c <= c;
```

Error: - Do not assign over the same signal in an always construct for sequential circuits

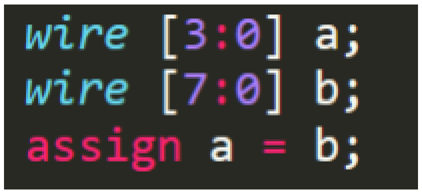
- No begin-end for if statement before else.
- Flip-flop should have an asynchronous set or reset.

Solution: - declare another variable with c variable to assign it to b.

- add begin-end in if statement.
- put asynchronous reset for flip flop

```
module Q2_c(input a,b,clk,rst, output reg c,d);

always@(posedge clk,negedge rst)
begin
if(~rst)
begin
c <= 0;
d <= 0;
end

else if (a > b)
begin
c <= a;
d <= b;
end

else
c <= c;

end

endmodule
```

D-



```
wire [3:0] a;
wire [7:0] b;
assign a = b;
```

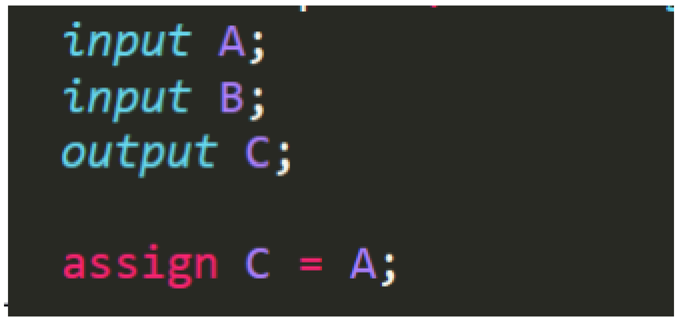
Error: - LHS width is less than RHS width of assignment

- Block should not contain feedthroughs.

Solution: - make size of LHS equal to size of RHS and solve feedthroughs using buffer.

```
module Q2_d(

input [7:0] b,
output [7:0] a
);
buf b1(a,b);

endmodule
```

E-



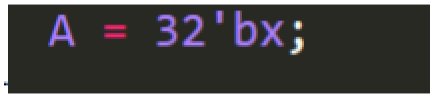
```
input A;
input B;
output C;

assign C = A;
```

Error: An input has been declared but not read

Solution: remove input B.

```
module Q2_e(
input a,
output c);

assign c = a;
endmodule
```

F-



```
A = 32'bx;
```

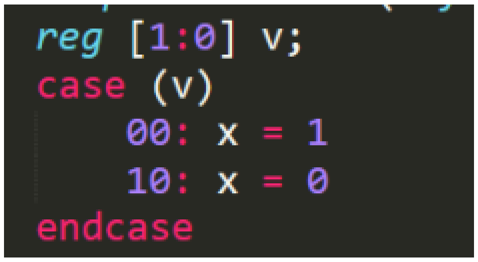
Error: - RHS of the assignment contains X.

- Based number 32'bx has no meaning in synthesis.

Solution: - assign another value for A

```
module Q2_f(
output [31:0] A
);
assign A = 32'b1;
endmodule
```

G-



```
reg [1:0] v;
case (v)
    00: x = 1
    10: x = 0
endcase
```

Error: - case statement(or selected signal assignment) does not have a default or OTHERS clause.

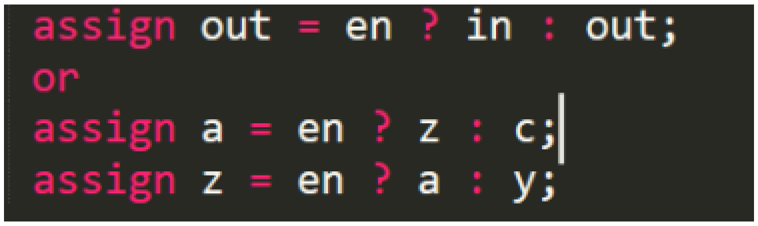
- Latch inferred for signal 'x' in module 'Q2_g'.

Solution: - add default statement

```
module Q2_g(
input [1:0] v,
output reg x
);
always @(*) begin
 case(v)
 2'b00: x = 1'b1;
 2'b10: x = 1'b0;
default: x = 1'b1;
endcase
```

```
end
endmodule
```

H-



```
assign out = en ? in : out;
or
assign a = en ? z : c;
assign z = en ? a : y;
```

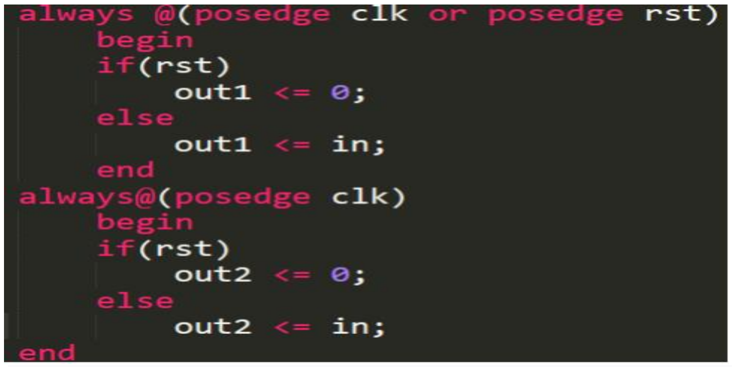
Error: - Combinational loop exists at 'Q2_h.a- Q2_h.z- Q2_h.out'.

- Reserved name 'out'.

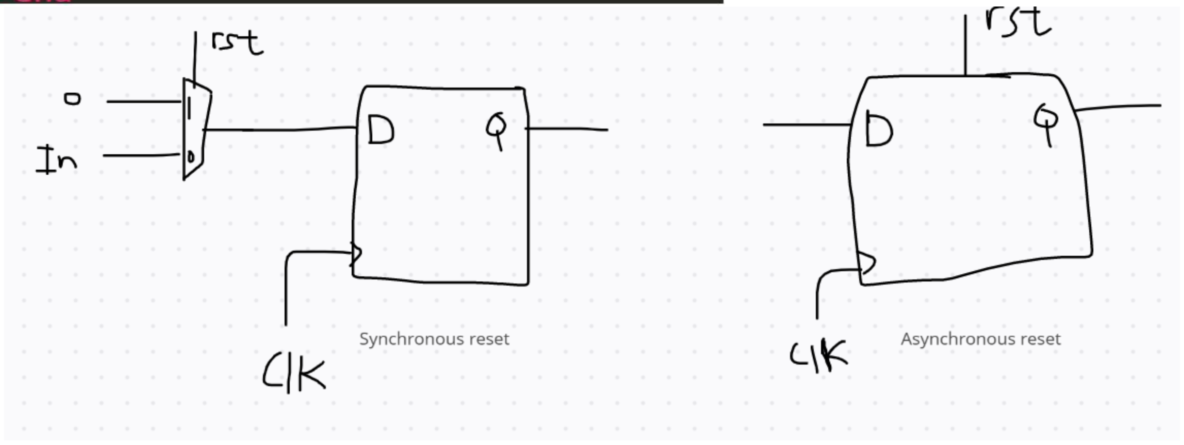
Solution: - change in assign statements to eliminate combinational loops

```
module Q2_h(
input wire en,inp,c,y,
output wire out,a,z
);
assign out = en? inp:c;

assign a = en? y:c;
assign z = en? inp:y;
endmodule
```

I-



```
always @(posedge clk or posedge rst)
    begin
    if(rst)
        out1 <= 0;
    else
        out1 <= in;
    end
always@(posedge clk)
    begin
    if(rst)
        out2 <= 0;
    else
        out2 <= in;
end
```



Error: - Flip-flop 'out2' has neither asynchronous set nor asynchronous reset.
- Asynchronous reset signal 'lint.rst' (flop: 'Q2_i.out1') used as non reset/synchronous-reset at instance 'Q2_i.out2_reg.D'

Solution: - put asynchronous reset for flip flop of out2

```
module Q2_i (
input wire clk,rst,in,
output reg out1,out2
);
always @(posedge clk or posedge rst) begin

if(rst) begin
 out1 <= 1'b0;
end
else begin
 out1 <= in;
end
end

always @(posedge clk or posedge rst) begin
if(rst) begin
 out2 <= 1'b0;
end
else begin
 out2 <= in;
end
end
endmodule
```