3. Introduction to Design With Verilog

3.4 Exercises

Dr. Paul D. Franzon
Exercise: Three Timing Examples (from Timing Notes)

- What do these look like in Verilog?

```verilog
always@(A or B or C)
begin
  if (A>B) then E = A; else E = B;
  if (C>E) then F = C; else F = E;
end
always@(posedge clock)
  if (D>F) then G <= D; else G <=F;
```

Why not move E, F assignments down to here?
E, F would now be outputs of flip-flops.
Three timing examples

- Produce a Verilog code fragment for ...
  - Use continuous assignment

```verilog
assign E = (A>B) ? A : B;
assign F = (C>D) ? C : D;

always@(posedge clock)
    if (E>F) then G <= E; else G <=F;
```
... Three Timing Examples

And for this...

```
always@(posedge clock)
begin
   if (A>B) then E <= A; else E <=B;
   C1 <= C;
   D1 <= D;
   if (E>C1) then F <= E; else F <= C1;
   D2 <= D1;
   if (F>D2) then G <= F; else G <= D2;
end
```
Sample Problem

Accumulator:

- Design an 8-bit adder accumulator with the following properties:
- While ‘accumulate’ is high, adds the input, ‘in1’ to the current accumulated total and add the result to the contents of register with output ‘accum_out’.
  - use absolute (not 2’s complement) numbers
- When ‘clear’ is high (‘accumulate’ will be low) clear the contents of the register with output ‘accum_out’
- The ‘overflow’ flag is high is the adder overflows

Hint:

8-bit adder produces a 9-bit result:
\[
\{\text{carry}_{\text{out}}, \text{sum}\} = A+B;
\]
Sketch Design

1. Determine and name registers.
2. Determine combinational logic
3. Hand generate a timing diagram to verify

Clear

accumulate

0

in1

+ 

overflow

accum_out
module accum (clock, accumulate, clear, in1, accum_out, overflow);

input clock, accumulate, clear;
input [7:0] in1;
output [7:0] accum_out;
output overflow;

reg [7:0] accum_out;
wire [7:0] accum_in;
wire overflow;

always@(posedge clock)
  begin
    if (clear) accum_out <= 8'b0;
    else if (accumulate) accum_out <= accum_in;
  end

assign {overflow, accum_in} = accum_out + in1;
endmodule /* counter */