How not to write a bad report

T J Kazmierski
Report is written to be read!

• Always write for your readers – you know who they are: your examiners
• Know the purpose of your report: academic assessment
• Hence
  • No need to repeat material from notes, data sheets
  • But you must
    • Demonstrate understanding and knowledge
    • Demonstrate results
      • State what you achieved in the Introduction as well as in the Conclusion
Structure

• **Introduction (short)**
  - What is the problem?
  - How have you solved the problem?
  - What remains unsolved
  - What are your main results?
  - How is the rest of the report organized?
Structure (cont)

• **Technical sections and results**
  - What are the results?
  - Why do they look the way they do? – Analysis!
    - Saying “Modelsim waveforms in fig X show that the sequencer works correctly” is not explaining anything.
  - Be organised: introduce-explain-summarise
  - Be succinct: make your point, then move on.

• **Presentation matters**
  - High-quality technical work requires high-quality presentation
    - Clear figures
    - Annotated code
    - Do not leave much blank space, especially around figures; use effectively the space you have
Structure (cont)

• **Conclusions and further work**
  • Remember: examiners usually read the Introduction and Conclusion - FIRST.
    • In Introduction: tell what you are going to tell (set the scene)
    • In Conclusion: tell what you have told (summarise)
  • State the main points you took away from your work. Show what have you learned. Do more analysis!
The ModelSim simulation (Figure 7) shows the register being reset on the first rising clock edge whilst 'reset' is high, then the register does not change even when 'Sum' is assigned a value until 'shift' goes high, the shifted values are as predicted above.

Figure 7: ModelSim simulation of register module
Example of efficient use of space

```
initial
begin
    add = 0;
    shift = 0;
    C = 0;
    Sum = 4'b0000;

    // testing reset
    reset = 0;
    Creg = 1;
    Qin = 4'b1011;
    #50ns reset = 1;
    #100ns reset = 0;

    // testing add
    Creg = 0;
    C = 1;
    Sum = 4'b1111;
    #50ns add = 1;
    #100ns add = 0;

    // testing shift
```

Figure 5: Register simulation waveform
Example of bad figure:
unclear, black background, no explanation what it shows

The following waveform suggests appropriate behaviour (figure 4).

Figure 4: Modelsim simulation of the register module.
Fig 5: The code for an n-bit register with add and shift in one cycle

The testbench and simulation results: (fig6 and fig7)

Fig 6: The code for the n-bit register (fig5)

Fig 7: The simulation results for the n-bit register (n=8)
always_comb is for combinational logic only!

// correct counter implementation:
logic[2:0] n;
always_ff @ (posedge clk)
if (present== idle)
  n <= 4;
else if (present == adding)
  n <= n-1;