ELEX 2117 Verilog Coding Guidelines

Version 2: Follow these revised guidelines for Lab 5 and later.

Mandatory Guidelines

Even if it operates correctly, a design for synthesis will be considered **incorrect** if you do not follow the guidelines in this section.

File-level Comments

As a minimum, each source file must include, near the start of the file, comments that include:

- the file name,
- a line describing the purpose of the file,
- the author's name and the date.

Explanation: These help to quickly identify the origin and purpose of the code.

Synchronous Design

Use one clock signal unless otherwise specified. The same signal must appear in all arguments to the `always_ff@(posedge...)` expression (the "sensitivity list"). Only that signal that should be labelled as a Clock in Compilation Report > Fitter > Resource Section > Control Signals.

Explanation: Programmable hardware and design tools assume synchronous (one clock) design. Using "computed" clocks, such as in ripple counters, leads to inefficient designs that are difficult to verify.

`logic` Type

Use the `logic` type for synthesizable signals.

Explanation: `wire` and `reg` types are not needed in System Verilog and should be avoided in new code. While `integer` and `float` types are useful for simulation, they are inappropriate for synthesis.

Consistent Indentation

An `end` line should be indented the same as the line with the corresponding `begin`. Each level of indentation should increase by the same amount (for example, by 4 spaces).

Explanation: This helps spot errors.

No Sequential Code within `always_ff`

An `always` block that defines a register a may only contain one assignment and it must be in the form x = x`_next` where x is a signal name. You may not include multiple assignments or sequential statements such as `if` or `case` within the `always_ff` .

Explanation: This rule ensures each `always_ff()` statement generates one register and that you have defined the signals at both the input (x`_next`) and output (x) of the register.

Recommended Guidelines

You may ignore the guidelines below where you think it makes your code easier to understand.

Add comments next to port and signal declarations and before non-obvious parts of your design. These should explain why you're doing something rather than repeating what is obvious from the code. They make it easier to understand your code.

Consistent signal naming helps avoid confusion. The following are widely-recognized conventions:

- append `_n` to active-low signals
- append `_t` to type names
- append `_in` to names of input ports that have a corresponding internal signal such as a synchronised or debounced version

- append `_next` to the name of a register output to derive the name of the register input

Use enumerated types for states to simplify your code and help the synthesizer optimize your design.