N26F300
VLSI SYSTEM DESIGN
(GRADUATE LEVEL)

Coding Guideline (I)

2 Coding Guideline for Design

Severity Levels Definition	
Mandatory 1 (M1)	Rules must be followed. If not, design must be fixed.
Mandatory 2 (M2)	Rules should be followed. If not, documentation must be provided.
Recommended (R)	Recommended deliverable items.

Ref : Reuse Methodology Manual (RMM) Keating & Bricaud, 3rd ed., 2002.

Basic Coding Practices

- Naming convention
  - Lower case for signals, variables, ports
  - Upper case for constants (define directive)
  - Meaningful names

    Ex: `ram_addr` for RAM address, `sys_bus` for system bus.
  - Short but descriptive parameters (for synthesis)
  - Prefix `clk` for clocks, prefix `rst` for reset.
  - `*_x` for active low, e.g., rst_n, act_b.
  - Use `[x:0]` rather than `[0:x]`
  - Use `*_r` for output of a register, e.g., count_r
  - Use `*_z` for tristate signal

- Do not use HDL reserved words.

Basic Coding Practices (cont.)

- State variables: `<name>.cs` for current state, `<name>.ns` for next state
- Use informational header for each source file:
  - Legal information (confidentiality, copyright, restriction, etc.), filename, author, date, version history, main contains.
- Use concise but explanatory comments where appropriately.
- Avoid multiple statements in one line.
- Use indentation to improve readability.
  - Use indentation of 2 spaces; do not use tab
- Port (core I/O) ordering
  - One port per line; a comment followed each port declaration
  - Follow the following order:
    - clocks, resets, enables, other control signals, data & Address lines

Basic Coding Practices (cont.)

- Port mapping
  - Use explicit mapping
  - Use named association rather than positional association

Ex:

```
 TagRam TagRam (
.Address    (PAddress[`INDEX]),
.TagIN       (PAddress[`TAG]),
.Tagout      (TagRAMTag[`TAG]),
.Write        (Write),
.Clk          (Clk)
);
```

Basic Coding Practices (cont.)

- Use functions wherever possible (for reusability and avoid repetition).
- Use vector operations (array) rather than loops for faster simulation.

Ex:

Poor:
```
for (i = 1, i < k, i ++)
    c_vec[i] = a_vec[i] ^ b_vec[i];
```

Better:
```
c_vec = a_vec ^ b_vec;
```

Clocks & Resets

- Preferably use a single global clock and positive edge-triggered flip-flops only.
- Avoid both positive and negative edge-triggered flip-flops.
- Separate positive-edge and negative-edge triggered flip-flops into different modules **--- make scan design easier.**
- Avoid clock buffers **--- leave it to clock insertion tool.**
- Avoid gated clock **--- to avoid false clock or glitch, and improve** testability.
- Avoid internally generated clocks **--- for testability.**
- If internally generated clocks are necessary, separate it in a top-level module.
- Avoid internally generated resets.

Coding for Synthesis

- Registers (flip-flops) are preferred.
- Latch should be avoided.
- Use design tools to check for latches.
- Poor coding may infer latches:
  - Missing **`else`** statement in **`if-then-else`** structures.
  - Missing assignments or conditions in **`case`** structures.
- To avoid the undesired inferred latches:
  - Assign default values at the beginning of a process.
  - Assign outputs for all input conditions.
  - Use default statements for case structures.
  - Always consider the else case in a if-then-else structure.
- If a latch must be used, well document it and prepare to make it testable via a mux.

Coding for Synthesis (cont.)

- Avoid combinational loops.
- Specify complete sensitivity lists.
- Always use nonblocking assignments in **`always @(posedge clk)`** blocks.
- Use **`case`** instead of nested **`if-then-else`** statements.
- Separate FSM and non-FSM logic in different modules.
- Keep late-arriving signals with critical timing closest to the output of a module, e.g., earlier in if-then-else structures.
- Do not use delay constants in RTL code to be synthesized.

Partitioning for Synthesis

- Partitioning can result in better synthesis results, faster compile and simulation time, and enable simpler synthesis strategy to meet timing requirement.
- Locate related combinational logic in a single module
- Separate modules that have different design goals.
- Avoid asynchronous design except reset.
- If asynchronous is required, put it in a separate module.
- Put relevant resources to be shared in the same module.

Coding Style (1/2)

- **[M1] Use a separate line for each HDL statement**

```
// Use:
  out1 = a1 & b1; // AND operation
  out2 = a2 | b2; // OR operation
// Do not use:
  out1 = a1 & b1; out2 = a2 | b2;
```

- **[M1] Avoid expression in port connections**

```
foo u_foo ( .bad ((m & n) ^ ((p | q) > 4'hc)),
            .good (good)); // expressions in port connections
```

- **[M1] Wire must be explicitly declared**

```
wire int_signal_a; // internal signal for block signal_a
wire int_signal_b; // internal signal for block signal_b
block u_block( .signal_a (int_signal_a), .signal_b (int_signal_b) );
```

Coding Style (2/2)

- **[M1] Operand size must match**

```
wire [32:0] signal_a; // internal signal_a with 33 bit width
wire [7:0] signal_b;  // internal signal_b with 8 bit width
// Do not use:
signal_x <= signal_a & signal_b;
signal_a <= 1;
// Use:
signal_a <= 33'h0_0000_0001;
```

- **[R] Expression in condition should be 1- bit value**

```
// Do not use:
if (bus)
  data_enable = 1; /*  The signal data_enable is set to 1 whenever the
multi-bit condition bus has the known value other than 0 */
// Use:
if (bus > 0)
  data_enable = 1;
/* This is the condition represents in 1-bit value expression, which is
instinctively easier to understand and avoid confusion */
```

Coding for Synthesis (1/3)

- **[M1] Use only synthesizable statement**
  - NOT waveform statements (initial)
  - NOT system task and function ($display, $monitor, ..)
  - NOT wait statement and # delay statement
  - NOT real, time, and event
  - ONLY one clock per always sensitivity list
  - ONLY loop with a static range

Coding for Synthesis (2/3)

- [M2] NO full_case and parallel_case directive statements
- [M2] Avoid unexpected latch inference
- [M1] NO User-Defined-Primitive (UDP)
- [M1] All unused module inputs must be driven

  Either by some other signal or by a fixed logic zero or one

Coding for Synthesis (3/3)

- [M2] Avoid top-level glue logic