Code Quality

ADIUVO
ENGINEERING AND TRAINING, LTD.

Code Quality

What do we mean by code quality

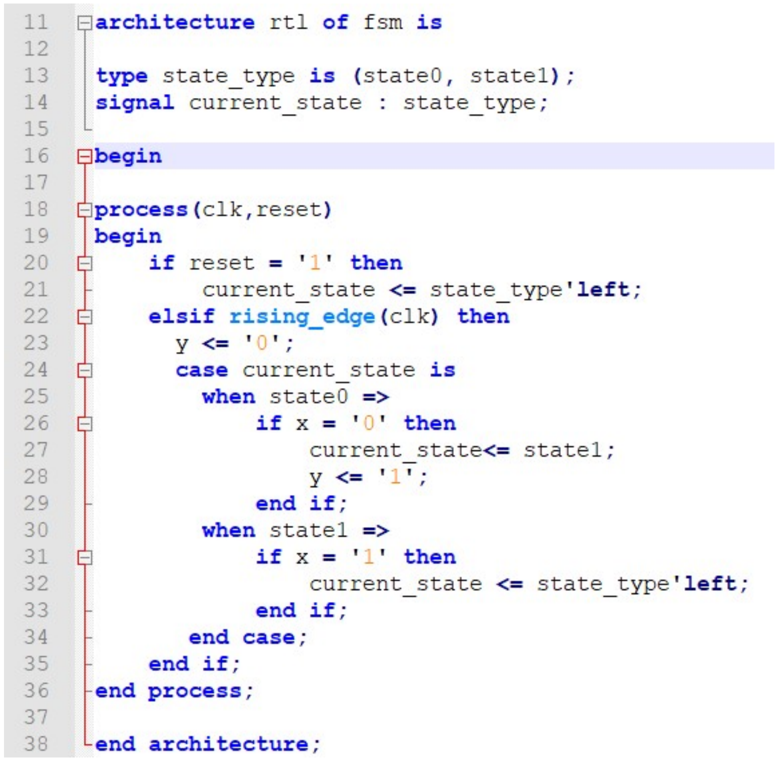
- » Readable and Maintainable
- » Comply with coding standards – either internal or external
- » Ensuring no simulation / synthesis mismatch
- » Enforce RTL clarity and reduce complexity
- » Enable Testability and Traceability of the code
- » Specific checks include
  - FSM without terminal states, unmapped states
  - Long If Then Else (ITE) chains
  - Control Signal test
  - Ensuring Structural correctness – e.g. used, undriven nets.
  - Gated Clocks
  - Allowable Types - e.g. std logic, unsigned etc

How can we achieve code quality?

Stringent set of coding rules – Including

- » Naming conventions (Architectures, Packages, Test Benches, Signals) etc.
  - Signal should indicate bi-directional, active low and synchronisation signals
- » Using Constants in place of hard coded numbers – in common packages if shared
- » Use of enumerated types for FSM
- » Rules on Signal vs Variable use
- » Rules on Synthesis Attributes

Bad coding practice example



```
architecture rtl of fsm is

type state_type is (state0, state1);
signal current_state : state_type;

begin

process(clk,reset)
begin
    if reset = '1' then
        current_state <= state_type'left;
    elsif rising_edge(clk) then
      y <= '0';
      case current_state is
        when state0 =>
            if x = '0' then
                current_state<= state1;
                y <= '1';
            end if;
        when state1 =>
            if x = '1' then
                current_state <= state_type'left;
            end if;
       end case;
    end if;
end process;

end architecture;
```

How can we enforce code quality

Linting tools – able to check the code against language and other rules

RTL Analysis e.g. Blue Pearl – Visual Verification Suite

- » Enables Linting checks against standard
- » Structural Design Checking
- » FSM Checking
- » Clock Domain Crossing Analysis
- » False and Multi Cycle Path Analysis

DO254 Rules

Number of Rules

Very simple and sensible approach split into 4 sections

- »Coding Practices
- »Clock Domain Crossing
- »Safe Synthesis
- »Design Reviews

Coding Practice

Coding Rule Name	Rule Number	BPS Check
Avoid Incorrect VHDL Type Usage	CP1	Should be a compilation error
Avoid Duplicate Signal Assignments	CP2	PREV_ASSIGN
Avoid Hard-Coded Numeric Values	CP3	HCCC
Avoid Hard-Coded Vector Assignment	CP4	UNIFORM_OTHERS_ASSIGNMENT
Ensure Consistent FSM State Encoding Style	CP5	FSM_NO_HCC
Ensure Safe FSM Transitions	CP6	RST, TERM_STATE, UNREACHABLE_STATE
Avoid Mismatching Ranges	CP7	MOS, MBA
Ensure Complete Sensitivity List	CP8	CSL
Ensure Proper Sub-Program Body	CP9	ACCESS_GLOBAL_VAR
Assign Value Before Using	CP10	UBA
Avoid Unconnected Input Ports	CP11	UNCONNECTED_UNDRIVEN_UNUSED_PORT
Avoid Unconnected Output Ports	CP12	UNCONNECTED_DRIVEN_UNUSED_NET/PORT
Declare Objects Before Use	CP13	UBA
Avoid Unused Declarations	CP14	UNCONNECTED_UNDRIVEN_UNUSED_NET/PIN, UNUSED_VAR, UNUSED_PARAM

ADIUVO
ENGINEERING AND TRAINING, LTD.

Safe Synthesis

Coding Rule Name	Rule Number	BPS Check
Implied Latches	SS1	MCD, MEB, ETB, MCA,MIA,MEA,UBA,LATCH_CREATED
Safe Case Synthesis	SS2	DCI, OCI, MCI, CASE_X, NO_XZ_CASE_ITEMS
Combinatorial feedback	SS3	CLPR
Avoid Latch Inference	SS4	MCD, MEB, ETB, MCA,MIA,MEA,UBA,LATCH_CREATED
Avoid Multiple Waveforms	SS5	NO_DELAY
Avoid Multiple Drivers	SS6	MDR
Avoid Uninitialized VHDL Deferred Constants	SS7	
Avoid Clock Used as Data	SS8	CLK_PROP_DFF_DATA, DIFF_CLK_PROP_DFF_DATA
Avoid Shared Clock and Reset Signal	SS9	RESET_CLOCK
Avoid Gated Clocks	SS10	CKGT
Avoid Internally Generated Clocks	SS11	IGCK
Avoid Internally Generated Resets	SS12	RSTMOD, RST_INTERNAL
Avoid Mixed Polarity Reset	SS13	RESET_POLARITY
Avoid Unresettable Registers	SS14	RST, NO_SR
Avoid Asynchronous Reset Release	SS15	SYNCH_DEASSERT_RST
Avoid Initialization Assignments	SS16	NO_INITIAL
Avoid Undriven and Unused Logic	SS17	UNCONNECTED_UNDRIVEN_UNUSED_NET/PIN,
Ensure Register Controllability	SS18	
Avoid Snake Paths	SS19	COMB_LEVELS, Path Analysis
Ensure Nesting Limits	SS20	ITE_DEPTH
Ensure Consistent Vector Order	SS21	RANGE_CHECK

Design Review

Coding Rule Name	Rule Number	BPS Check
Use Statement Labels	DR1	COMMENT_END_STMNTS, COMMENT_NET_DEC, COMMENT_PORT_DEC
Avoid Mixed Case Naming for Differentiation	DR2	
Ensure Unique Name Spaces	DR3	
Use Separate Declaration Style	DR4	MULT_PORTS
Use Separate Statement Style	DR5	SLCC
Ensure Consistent Indentation	DR6	CHECK_INDENTATION
Avoid Using Tabs	DR7	NO_TABS
Avoid Large Design Files	DR8	MAX_LINES_PER_MODULE
Ensure Consistent Signal Names Across Hierarchy	DR9	
Ensure Consistent File Header	DR10	FILE_HEADER
Ensure Sufficient Comment Density	DR11	Use SLOC report
Ensure Proper Placement of Comments	DR12	
Ensure Company Specific Naming Standards	DR13	NAMING_RULES