Enumerate Data Types:

type stoplight\_state is (red, green, yellow);

signal next\_state, current\_state : stoplight\_state;

Case Statements:

process (current\_state,–- Other sensitivity list inputs)

begin

case current\_state is

when state1=>

-- Sequential Statements

when state2 =>

-- Sequential Statements

when state3 =>

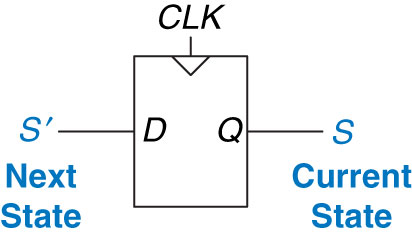
-- Sequential Statements

when others =>

-- Sequential Statements

end case;

end process;



Basic Next State Flip Flop:

process (clk, reset)

begin

if reset = '1' then

current\_state <= Reset\_State; --Reset state

elsif (rising\_edge(clk)) then

current\_state <= next\_state; --next state becomes current state

end if;

end process;

Testbenches:

* Assert (check output)
* Report (report message)
* Severity (Error, Warning, Note, Failure)

A <= '0'; B <= '0'; C <= '0'; D <= '0' ; S <= "00";

wait for 10 ns;

ASSERT Y = '1' REPORT "Test 0000 00 Failed" SEVERITY ERROR;

-- Can use Error, Warning, Note Failure

Be Careful with its use: What could be an issue?



Component Declaration:

Declaration: In Architecture…Before Begin

component Component\_Name

port ( -- signal\_name : mode signal\_type;

Input0, Input1 : in std\_logic;

Output :out std\_logic);

end component;.

Instantiation: In Architecture…After Begin

Component\_Label: Component\_Name

port map ( -- Port\_name => Signal name,

Input0 => a,

Input1 => b,

Output => Y);