VHDL in Practice

Instructor:

Dr. Ahmad El-Banna

DAY#2
SUMMER 2016

Agenda

- **Test bench**
- Modeling the Structure way
- Structural Example
- Behavioral & Structural Example

Test Bench your Model

- Testing a design by simulation
- Use a *test bench* model
  - a Model that uses your Model
  - apply test sequences to your inputs
  - monitors values on output signals
    - either using simulator
    - or with a process that verifies correct operation
    - or logic analyzer

Test Bench of the multiplexer example

```
--------------------------------------------------------------------
-- Test Bench for Multiplexer (ESD figure 2.5)-- by Weijun Zhang, 04/2001
-- four operations are tested in this example.
--------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity Mux_TB is                    -- empty entity
end Mux_TB;
--------------------------------------------------------------------

architecture TB of Mux_TB is
   -- initialize the declared signals
   signal T_I3: std_logic_vector(2 downto 0):="000";
   signal T_I2: std_logic_vector(2 downto 0):="000";
   signal T_I1: std_logic_vector(2 downto 0):="000";
   signal T_I0: std_logic_vector(2 downto 0):="000";
   signal T_O: std_logic_vector(2 downto 0);
   signal T_S: std_logic_vector(1 downto 0);

   component Mux
   port(       I3:             in std_logic_vector(2 downto 0);
               I2:             in std_logic_vector(2 downto 0);
               I1:             in std_logic_vector(2 downto 0);
               I0:             in std_logic_vector(2 downto 0);
               S:              in std_logic_vector(1 downto 0);
               O:              out std_logic_vector(2 downto 0)
   );
   end component;
```

Steps of a testbench:

- entity declaration for your testbench.
- Component Declaration for the Unit Under Test (UUT)
- declare inputs and initialize them
- declare outputs and initialize them
- Clock period definitions
- Stimulus process

© Ahmad El-Banna

Test Bench of the multiplexer example..

```
begin
   U_Mux: Mux port map (T_I3, T_I2, T_I1, T_I0, T_S, T_O);
   process

variable err_cnt: integer :=0;
   begin

                T_I3 <= "001";                  -- I0-I3 are different signals
                T_I2 <= "010";
                T_I1 <= "101";
                T_I0 <= "111";

                -- case select eqaul "00"
                wait for 10 ns;
                T_S <= "00";
                wait for 1 ns;
                assert (T_O="111") report "Error Case 0" severity error;
                if (T_O/="111") then
                   err_cnt := err_cnt+1;
                end if;
                -- case select equal "01"
                wait for 10 ns;
                T_S <= "01";
                wait for 1 ns;
                assert (T_O="101") report "Error Case 1" severity error;
                if (T_O/="101") then
                   err_cnt := err_cnt+1;
                end if;
```

© Ahmad El-Banna

Test Bench of the multiplexer example...

```
-- case select equal "10"
        wait for 10 ns;
        T_S <= "10";
        wait for 1 ns;
        assert (T_O="010") report "Error Case 2" severity error;
        if (T_O/="010") then
           err_cnt := err_cnt+1;
        end if;
        -- case select equal "11"
        wait for 10 ns;
        T_S <= "11";
        wait for 1 ns;
        assert (T_O="001") report "Error Case 3" severity error;

        if (T_O/="001") then
           err_cnt := err_cnt+1;
        end if;
        -- case equal "11"
        wait for 10 ns;
        T_S <= "UU";
```

```
-- summary of all the tests
                if (err_cnt=0) then
                   assert (false)
                   report "Testbench of Mux completed sucessfully!"
                   severity note;
                else
                   assert (true)
                   report "Something wrong, try again!"
                   severity error;
                end if;

                wait;

   end process;

end TB;

-----------------------------------------------------------
configuration CFG_TB of Mux_TB is
                for TB
                end for;
end CFG_TB;
-----------------------------------------------------------
```

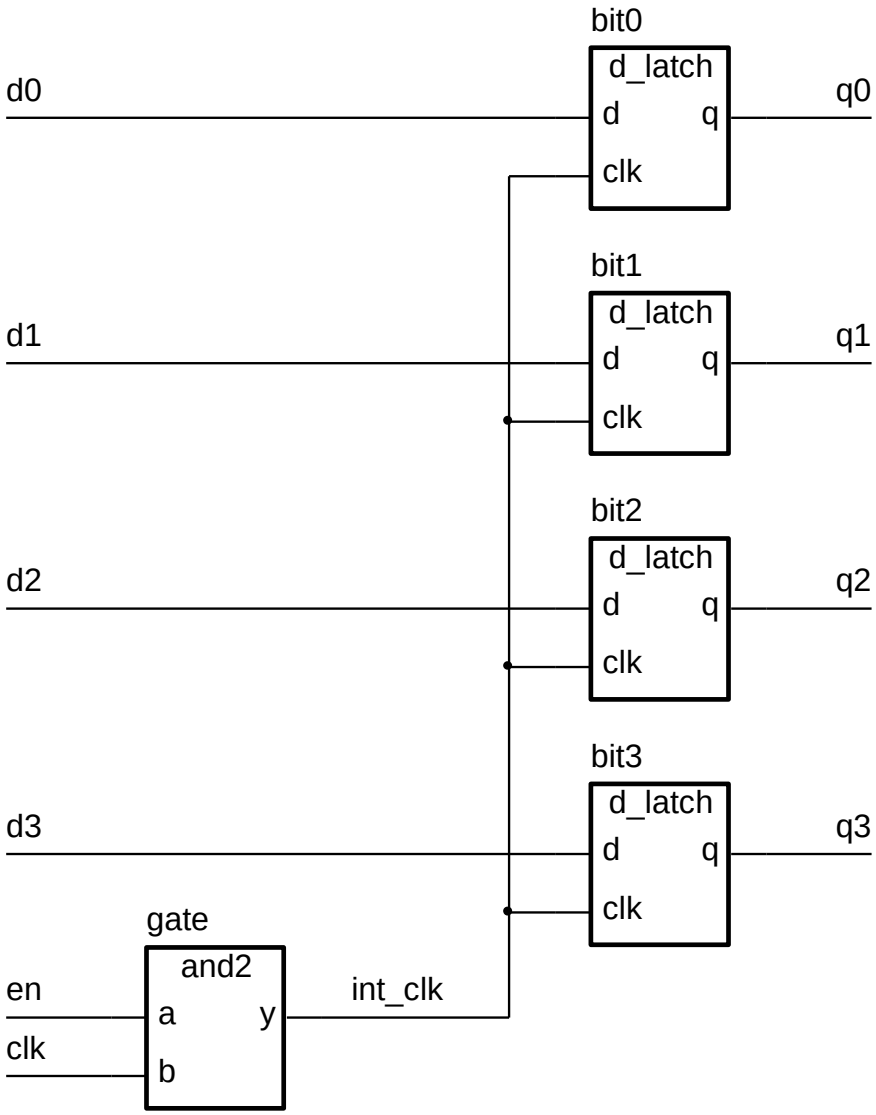
© Ahmad El-Banna

Modeling the Structurural way

- *Structural* architecture
  - implements the module as a composition of subsystems
  - contains
    - *signal declarations*, for internal interconnections
      - the entity ports are also treated as signals
    - *component instances*
      - instances of previously declared entity/architecture pairs
    - *port maps* in component instances
      - connect signals to component ports

Structural way Example

Not complete example, just for concept justification ..



© Ahmad ELBanna

Structural way..

- First **declare D-latch** and **and-gate** entities and architectures

```
entity d_latch is
    port ( d, clk : in bit;  q : out bit );
end entity d_latch;

architecture basic of d_latch is
begin
    process (clk, d)
    begin
        if clk = '1' then
            q <= d after 2 ns;
        end if;
    end process;
end basic;
```

```
entity and2 is
    port ( a, b : in bit;  y : out bit );
end entity and2;

architecture basic of and2 is
begin
    process (a, b)
    begin
        y <= a and b after 2 ns;
    end process ;
end basic;
```

© Ahmad El-Banna

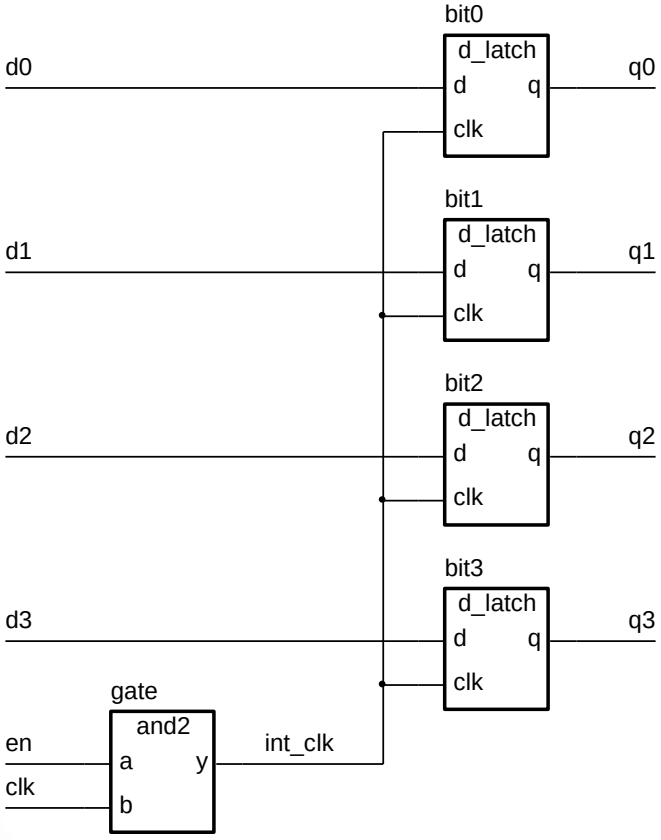
Structural way…

- **Declare** corresponding **components** in register architecture body

```
architecture struct of reg4 is
    component d_latch
        port ( d, clk : in bit;  q : out bit );
    end component;
    component and2
        port ( a, b : in bit;  y : out bit );
    end component;
    signal int_clk : bit;
...
```

Structural way....

- Now **use them** to implement the register




```
...
begin
    bit0 : d_latch
        port map ( d0, int_clk, q0 );
    bit1 : d_latch
        port map ( d1, int_clk, q1 );
    bit2 : d_latch
        port map ( d2, int_clk, q2 );
    bit3 : d_latch
        port map ( d3, int_clk, q3 );
    gate : and2
        port map ( en, clk, int_clk );
end struct;
```

© Ahmad ELBanna

Trace the code & Draw the model structure

```
------------------------------------------------------------
-- Combinational Logic Design
-- (ESD book figure 2.4)
-- by Weijun Zhang, 04/2001
--
-- A simple example of VHDL Structure Modeling
-- we might define two components in two separate files,
-- in main file, we use port map statement to instantiate
-- the mapping relationship between each components
-- and the entire circuit.
------------------------------------------------------------

library ieee;                              -- component #1
use ieee.std_logic_1164.all;

entity OR_GATE is
port(   X:      in std_logic;
        Y:      in std_logic;
        F2:     out std_logic
);
end OR_GATE;

architecture behv of OR_GATE is
begin
process(X,Y)
begin
        F2 <= X or Y;                      -- behavior des.
end process;
end behv;
```

```
------------------------------------------------------------

library ieee;                              -- component #2
use ieee.std_logic_1164.all;

entity AND_GATE is
port(   A:      in std_logic;
        B:      in std_logic;
        F1:     out std_logic
);
end AND_GATE;

architecture behv of AND_GATE is
begin
process(A,B)
begin
        F1 <= A and B;                     -- behavior des.
end process;
end behv;

------------------------------------------------------------
```

© Ahmad EL-Banna

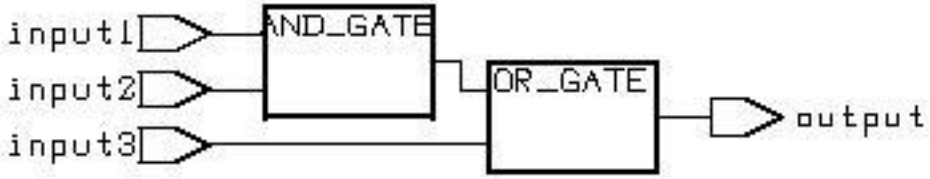
Trace the code & Draw the model structure..

```
------------------------------------------------------------

library ieee;                                   -- top level circuit
use ieee.std_logic_1164.all;
use work.all;

entity comb_ckt is
port(   input1: in std_logic;
        input2: in std_logic;
        input3: in std_logic;
        output: out std_logic
);
end comb_ckt;

architecture struct of comb_ckt is

    component AND_GATE is                       -- as entity of AND_GATE
    port(   A:  in std_logic;
            B:  in std_logic;
            F1: out std_logic
    );
    end component;

    component OR_GATE is                        -- as entity of OR_GATE
    port(   X:  in std_logic;
            Y:  in std_logic;
            F2: out std_logic
    );
    end component;

    signal wire: std_logic;                     -- signal just like wire

begin

    -- use sign "=>" to clarify the pin mapping

    Gate1: AND_GATE port map (A=>input1, B=>input2, F1=>wire);
    Gate2: OR_GATE port map (X=>wire, Y=>input3, F2=>output);

end struct;

------------------------------------------------------------
```

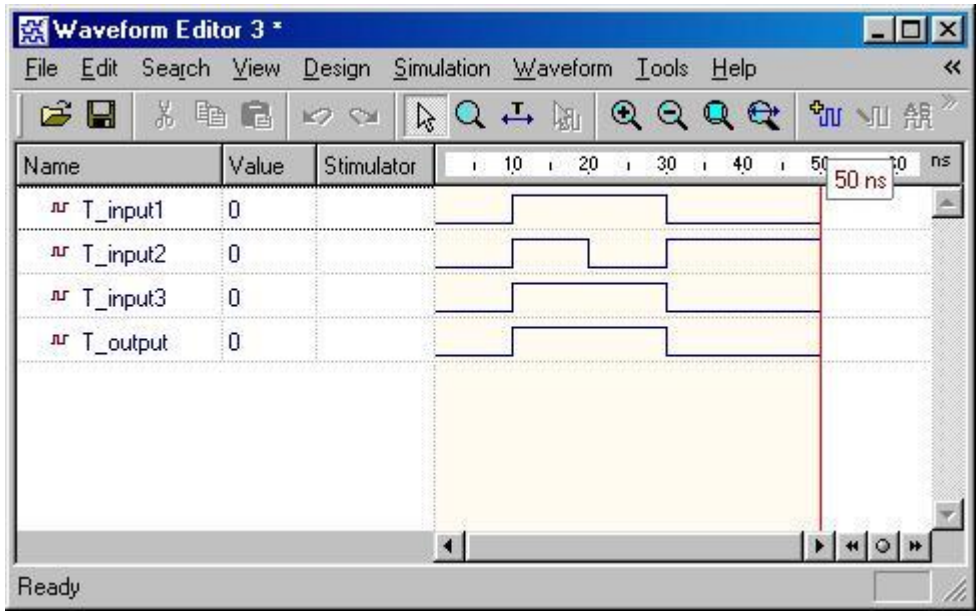
© Ahmad ELBanna

Trace the code & Draw the model structure..

- **the model**



- **Simulation waveform**



© Ahmad El-Banna

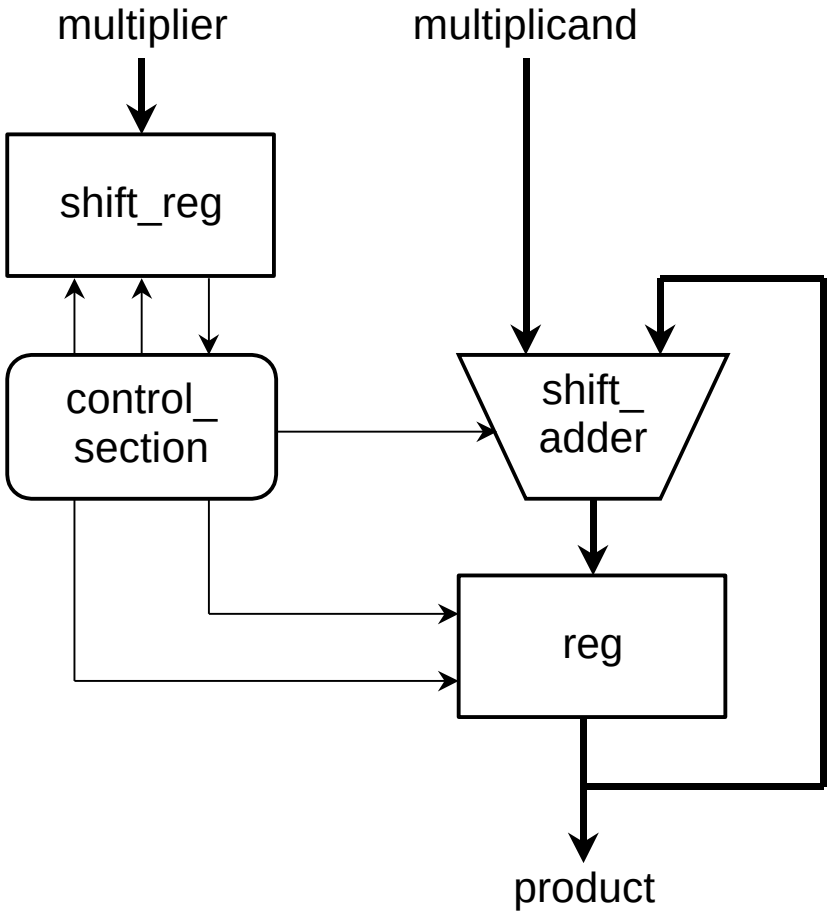
Mixed Behavior and Structure

- An architecture can contain both behavioral and structural parts
  - process statements and component instances
    - collectively called *concurrent statements*
  - processes can read and assign to signals
- Example: register-transfer-level (RTL) Model
  - data path described structurally
  - control section described behaviorally

© Ahmad El-Banna

Mixed Example

Not complete example, just for concept justification ..



© Ahmad El-Banna

Mixed Example

```
entity multiplier is
    port ( clk, reset : in bit;
           multiplicand, multiplier : in integer;
           product : out integer );
end multiplier;

architecture mixed of mulitplier is
    signal partial_product, full_product : integer;
    signal arith_control, result_en, mult_bit, mult_load : bit;
begin
    arith_unit : entity work.shift_adder(behavior)
        port map ( addend => multiplicand,  augend => full_product,
                   sum => partial_product,
                   add_control => arith_control );
    result : entity work.reg(behavior)
        port map ( d => partial_product,  q => full_product,
                   en => result_en,  reset => reset );
    ...
```

Mixed Example..

```
    …
    multiplier_sr : entity work.shift_reg(behavior)
        port map ( d => multiplier,  q => mult_bit,
                   load => mult_load,  clk => clk );
    product <= full_product;

    process (clk, reset)
        -- variable declarations for control_section
        -- …
    begin
        -- sequential statements to assign values to control signals
        -- …
    end process;
end mixed;
```

- For more details, refer to:
  - VHDL Tutorial: Learn by Example *by Weijun Zhang*
    - http://esd.cs.ucr.edu/labs/tutorial/
  - "**Introduction to VHDL**" presentation by Dr. Adnan Shaout, *The University of Michigan-Dearborn*
  - **The VHDL Cookbook**, *Peter J. Ashenden, 1st edition, 1990.*
- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg