Filename=”ch4.doc”

4.0 Structural Modeling

Describing entity in terms of its subcomponents and how they are connected.
Hierarchical arrangement of interconnected components.
Ports and their interconnection are central focus.
Has to be supported at some level by behaviorial description.

Structural Steps
Declare component
Instantiate component
Identify which component

4.1 Component Declaration

- Describes local interface
  name
  ports and generics
- Appears in declarative areas.
- Format:
  COMPONENT identifier_name
  [GENERIC(local_generic_list);]
  [PORT(local_port_list);]
  END COMPONENT;

Example:

COMPONENT and2
PORT(a,b: IN qsim_state;c: OUT qsim_state);
END COMPONENT;

4.2 Component Specification

- Identifies which design entity is to be used as component.
- States:
  name of entity
  name of architecture
- Format:
  FOR component_instantiation_label:component_name
  USE binding_indication;
- Binding_indication:
  ENTITY library.entity_name[(architecture_name)]

Example:
FOR U1: and2 USE ENTITY work.and_gate(behav);

4.3 Component Instantiation

- Places component in relationship to other components.
- Identifies signals connected to component ports.
- Associates values to generics.

- Format:
  label: component_declaration_name --label required
  [GENERIC MAP (association_list)] -- Note no ";"
  [PORT MAP (association_list)];

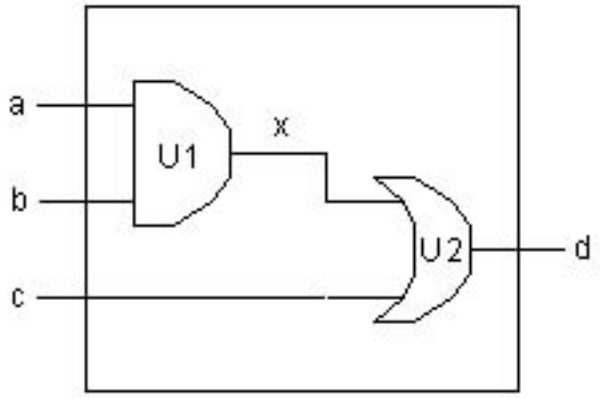
Example:

```
u2: and2
    PORT MAP(signal1,signal2,signal3);
```

4.4 Structural Example

Specification :

Create model of entity with three input ports(a,b,c) and one output port (d). Signals a and b are ANDed and the result(x) is ORed with signal c to produced output signal d.



Analysis:

Two components will be declared, specified, and instantiated. One for the and_gate and the other for the or_gate. In addition an internal signal x which represents the output of the and_gate and input of the or_gate will be declared.

```
--Module 1
  LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;

  ENTITY and_gate IS
        PORT(x,y: IN STD_LOGIC; z: OUT STD_LOGIC);
  END and_gate;

  ARCHITECTURE behav OF and_gate IS
  BEGIN
        z <= x AND y;
  END behav;

--Module 2

  LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY or_gate IS
        PORT(x,y: IN STD_LOGIC; z: OUT STD_LOGIC);
END or_gate;

ARCHITECTURE behav OF or_gate IS
BEGIN
        z <= x OR y;
END behav;

--Module 3

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY my_thing IS
        PORT (a,b,c: IN STD_LOGIC;d: OUT STD_LOGIC);
END my_thing;

--Structural modeling using position association

ARCHITECTURE structure OF my_thing IS
        COMPONENT and2
                PORT(x,y: IN STD_LOGIC; z: OUT STD_LOGIC);
        END COMPONENT and2;

        COMPONENT or2
                PORT(x,y: IN STD_LOGIC; z: OUT STD_LOGIC);
        END COMPONENT or2;

FOR u1: and2 USE ENTITY work.and_gate(behav);
FOR u2: or2 USE ENTITY work.or_gate(behav);

        SIGNAL x: STD_LOGIC;
BEGIN
        u1: and2 PORT MAP(a,b,x);               --position association
        u2: or2 PORT MAP(x,c,d);                --position association
END structure;

--Another structural modeling using name association

ARCHITECTURE structure2 OF my_thing IS
        COMPONENT and2
                PORT(In1,In2: IN STD_LOGIC; Out1: OUT STD_LOGIC);
        END COMPONENT and2;

        COMPONENT or2
                PORT(In1,In2: IN STD_LOGIC; Out1: OUT STD_LOGIC);
        END COMPONENT or2;

FOR u1: and2 USE ENTITY work.and_gate(behav);
FOR u2: or2  USE ENTITY work.or_gate(behav);



        SIGNAL x: STD_LOGIC;
```

```
BEGIN
        u1: and2 PORT MAP(a=>In1,b=>In2,x=>Out1);       --name association
        u2: or2 PORT MAP(x=>In1,c=>In2,d=>Out1);        --name association
END structure2;
```

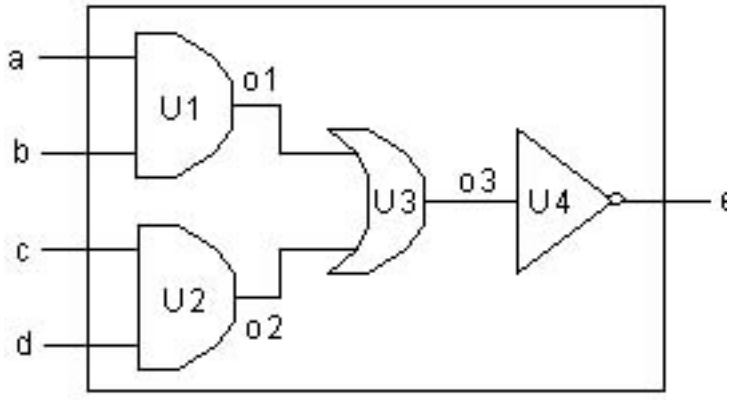
4.5 Behavioral, Data Flow, and Structural Modeling Example

A VHDL data flow description and a register transfer language description are similar in that they describe the function of a design by defining the flow of information from one input or register to another register or output.

A data flow description uses a large number of concurrent signal assignment; concurrent statements used in data flow descriptions include the following:

- BLOCK statement
- Concurrent PROCEDURE call
- Concurrent ASSERT statement
- Concurrent SIGNAL assignment statement

The given circuit will be described using behavioral, data flow and structural.




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY my_ckt IS
        PORT( a, b, c, d : IN STD_LOGIC ; e : OUT STD_LOGIC);
END my_ckt;
```

4.5.1 Behavioral Modeling Architecture of my_ckt

```
ARCHITECTURE behavior OF my_ckt IS
BEGIN
        e <= NOT((a AND b) OR (c AND d));
END behavior;
```

4.5.2 Data Flow Modeling Architecture of my_ckt

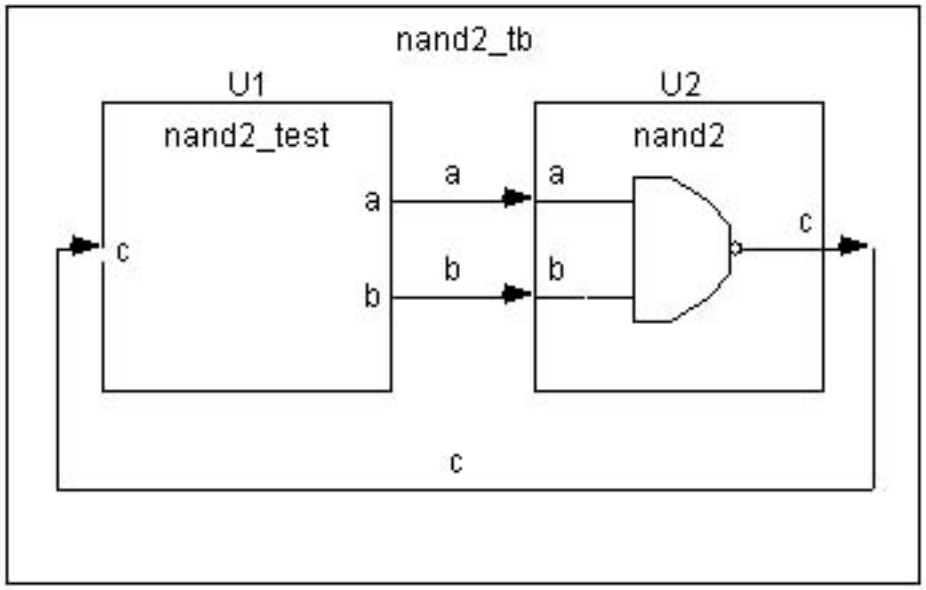
```
ARCHITECTURE dataflow OF my_ckt IS
        SIGNAL o1, o2, o3 : STD_LOGIC;          --specify internal signals or wires
```

```
BEGIN
        e <= NOT o3;
        o1 <= a AND b;
        o2 <= c AND d;
        o3 <= o1 OR o2;
END dataflow;
```

4.5.3 Structural Modeling Architecture of my_ckt

```
ARCHITECTURE structural OF my_ckt IS
        COMPONENT and2
                PORT(a, b : IN STD_LOGIC; c : OUT STD_LOGIC);
        END COMPONENT and2;

        COMPONENT or2
                PORT(a, b : IN STD_LOGIC; c : OUT STD_LOGIC);
        END COMPONENT or2;

        COMPONENT inv
                PORT(a : IN STD_LOGIC; b : OUT STD_LOGIC);
        END COMPONENT inv;

        SIGNAL o1, o2, o3: STD_LOGIC;
        FOR U1, U2 :    and2    USE ENTITY work.and_gate(behav);
        FOR U3:         or2     USE ENTITY work.or_gate(behav);
        FOR U4:         inv     USE ENTITY work.inv(behav);

BEGIN
        U1:     and2    PORT MAP(a, b, o1);
        U2:     and2    PORT MAP(c, d, o2);
        U3:     or2     PORT MAP(o1, o2, o3);
        U4:     inv     PORT MAP(o3, e);
END structural;
```

4.6 Test Bench Example




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY nand2 IS
        PORT(a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
END ENTITY nand2;

ARCHITECTURE arch_nand2 OF nand2 IS
BEGIN
        c <= a NAND b;
END ARCHITECTURE arch_nand2;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--------------------------------------------------
entity nand2_test is
        port (a,b:out std_logic;
                c:in std_logic);
end entity nand2_test_bench;
--------------------------------------------------
architecture behav of nand2_test is
begin
        test_process: process is
        begin
                a<='1';
                b<='1';
                wait for 1 ns;
                assert (c='0')
                        report "Not Working" severity error;
                end if;
```

```
                wait for 10 ns;
                a<='0';
                b<='1';
                wait for 1 ns;
                assert (c='1')
                        report "Not Working" severity error;
                end if;
                wait for 10 ns;
                a<='1';
                b<='0';
                wait for 1 ns;
                assert (c='1')
                        report "Not Working" severity error;
                end if;
                wait for 10 ns;
                a<='0';
                b<='0';
                wait for 1 ns;
                assert (c='1')
                        report "Not Working" severity error;
                end if;
                wait for 100 ns;
        end process test_process;
end architecture behav;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity nand2_tb is
end nand2_tb;
----------------------------
architecture behav of nand2_tb is
        component nand2
                port (a,b:in std_logic;
                c:out std_logic);
        end component;
        component nand2_test
                port (a,b:out std_logic;
                c:in std_logic);
        end component;
        for U1:nand2_test use entity work.nand2_test_bench(behav);
        for U2:nand2 use entity work.nand2(arch_nand2);
        signal a, b, c:std_logic;
begin
        U1: nand2_test port map(a, b, c);
        U2: nand2 port map(a, b, c);
end behav;
```