

Main Language Concepts

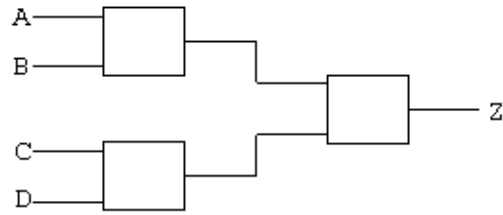


Figure 1. Concurrent

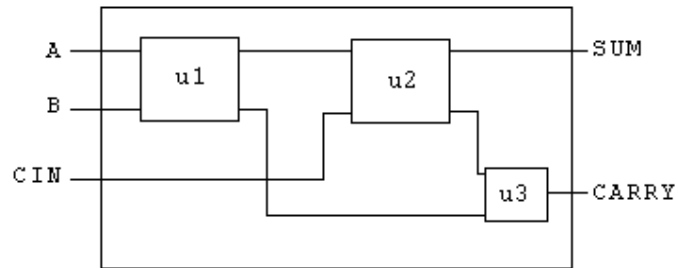


Figure 2. Structure

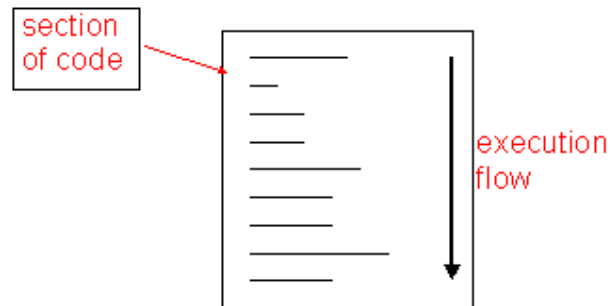


Figure 3. Procedural

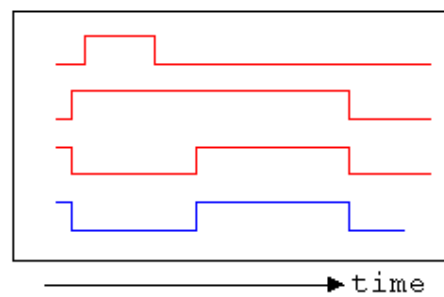


Figure 4. Time

Entity

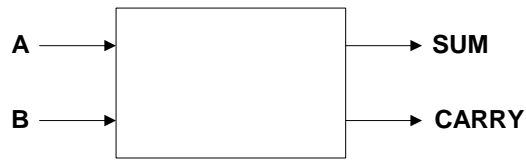


Figure 5. Half adder

```
entity HALFADD is
  port(A,B : in bit;
        SUM, CARRY : out bit);
end HALFADD;

architecture DATAFLOW of HALFADD is
begin
  SUM <= A xor B;
  CARRY <= A and B;
end DATAFLOW;
```

Figure 6. Dataflow architecture for half adder

Several Architectures for One Entity

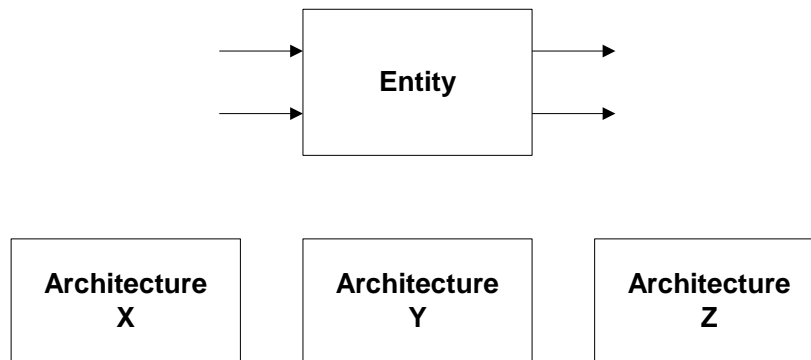


Figure 7. Entity with three architectures

```
library IEEE;
use IEEE.std_logic_1164.all;

entity FULLADD is
  port (A, B, CIN : in bit;
        SUM, CARRY : out bit);
end FULLADD;
```

Figure 8. Entity for full adder

Table 1. Truth table for full adder

A	B	CIN	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

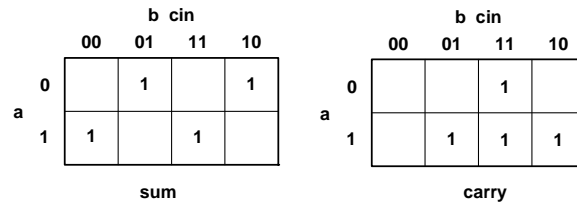


Figure 9. Karnaugh maps for SUM and CARRY

```

architecture DATAFLOW of FULLADD is
begin
    SUM <= CIN xor A xor B;
    CARRY <= (A and B) or (CIN and A) or (CIN and B);
end DATAFLOW;

configuration CFG_DAT of FULLADD is
    for DATAFLOW
    end for;
end CFG_DAT;

```

Figure 10. Dataflow architecture for full adder

```

architecture BEHAV of FULLADD is
begin
    process (A, B, CIN)
    begin
        if CIN = '0' and A = '0' and B = '0' then
            SUM <= '0';
            CARRY <= '0';
        elsif (CIN = '0' and A = '0' and B = '1') or
              (CIN = '0' and A = '1' and B = '0') or
              (CIN = '1' and A = '0' and B = '0') then
            SUM <= '1';
            CARRY <= '0';
        elsif (CIN = '0' and A = '1' and B = '1') or
              (CIN = '1' and A = '0' and B = '1') or
              (CIN = '1' and A = '1' and B = '0') then
            SUM <= '0';
            CARRY <= '1';
        elsif CIN = '1' and A = '1' and B = '1' then
            SUM <= '1';
            CARRY <= '1';
        end if;
    end process;
end BEHAV;

configuration CFG_BEH of FULLADD is
    for BEHAV
    end for;
end CFG_BEH;

```

Figure 11. Behavioral architecture for full adder

Structural architecture

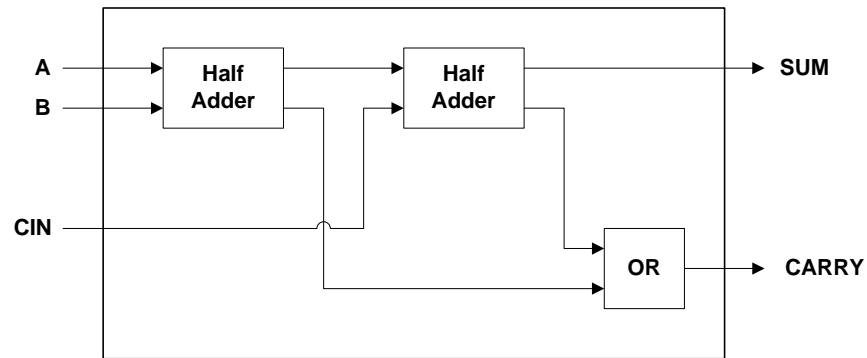


Figure 12. Full adder

```
entity HALFADD is
  port(A,B : in bit;
        SUM, CARRY : out bit);
end HALFADD;

architecture DATAFLOW of HALFADD is
begin
  SUM <= A xor B;
  CARRY <= A and B;
end DATAFLOW;
```

Figure 13. Design entity for half adder

```
entity ORGATE is
  port(A,B : in bit;
        Z : out bit);
end ORGATE;

architecture DATAFLOW of ORGATE is
begin
  Z <= A or B;
end DATAFLOW;
```

Figure 14. Design entity for OR gate

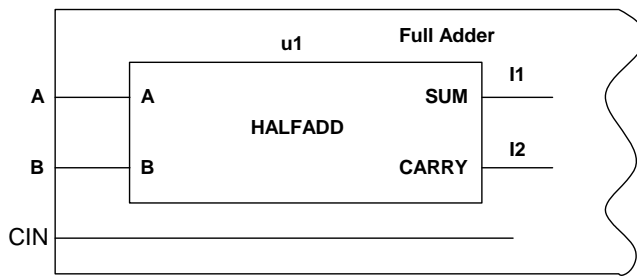


Figure 15. Instantiation of half adder into full adder

```
architecture STRUCT of FULLADD is
  signal I1, I2, I3 : bit;
  component HALFADD
    port(A,B : in bit;
         SUM, CARRY : out bit);
  end component;

  component ORGATE
    port(A,B : in bit;
         Z : out bit);
  end component;

begin
  u1:HALFADD port map(A,B,I1,I2);
  u2:HALFADD port map(I1,CIN,SUM,I3);
  u3:ORGATE port map(I3,I2,CARRY);
end STRUCT;

configuration CFG_STR of FULLADD is
  for STRUCT
  end for;
end CFG_STR;
```

Figure 16. Structural architecture for full adder

Configuration

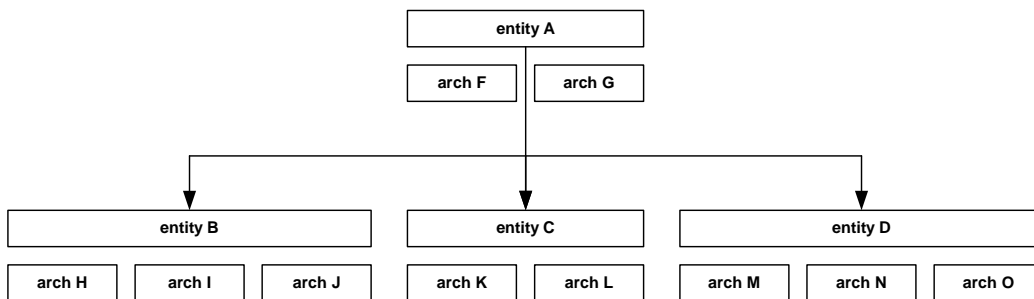


Figure 17. Entities with several architectures

Table 2. Correspondence between entity and architecture

Entity	Architecture
A	G
B	J
C	K
D	M

Project1 for discussion

Project1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity proj1 is
    port (x1, x2, x3, x4 : in bit;
          y1, y2 : out bit);
end proj1;

architecture arc_proj1 of proj1 is
begin
    y1 <= (x1 and not x2) or (not x1 and x2);
    y2 <= (x1 and x3 and not x4) or (x1 and not x2) or (not x1 and
        x2);
end arc_proj1;

configuration cfg_proj1 of proj1 is
    for arc_proj1
    end for;
end cfg_proj1;
```

Test bench2 for project1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity proj1_tb is
end proj1_tb;

architecture arc_proj1_tb of proj1_tb is

    component proj1
        port (x1, x2, x3, x4 : in bit;
              y1, y2 : out bit);
    end component;

    signal x1, x2, x3, x4, y1, y2 : bit;

begin
    design: proj1 port map (x1, x2, x3, x4, y1, y2);

    process
    begin
        x1 <= '0'; x2 <= '0'; x3 <= '0'; x4 <= '0';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
            severity error;

        x1 <= '0'; x2 <= '0'; x3 <= '0'; x4 <= '1';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
```

```

        severity error;

x1 <= '0'; x2 <= '0'; x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x1 <= '0'; x2 <= '0'; x3 <= '1'; x4 <= '1';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x1 <= '0'; x2 <= '1'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '0'; x2 <= '1'; x3 <= '0'; x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '0'; x2 <= '1'; x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '0'; x2 <= '1'; x3 <= '1'; x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '0'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

```

```

x1 <= '1'; x2 <= '0'; x3 <= '0'; x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '0'; x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '0'; x3 <= '1'; x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '1'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '1'; x3 <= '0'; x4 <= '1';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '1'; x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '1'; x3 <= '1'; x4 <= '1';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;
wait;

```



```

        end process;
    end arc_proj1_tb;
configuration cfg_proj1_tb of proj1_tb is
    for arc_proj1_tb
        end for;
end cfg_proj1_tb;

```

Test bench3 for project1

```

library IEEE;
use IEEE.std_logic_1164.all;

entity proj1_tb is
end proj1_tb;

architecture arc_proj1_tb of proj1_tb is

    component proj1
        port (x1, x2, x3, x4 : in bit;
              y1, y2 : out bit);
    end component;

    signal x1, x2, x3, x4, y1, y2 : bit;

begin
    design: proj1 port map (x1, x2, x3, x4, y1, y2);

    process
    begin
        x1 <= '0'; x2 <= '0'; x3 <= '0'; x4 <= '0';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
            severity error;

        x4 <= '1';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
            severity error;

        x3 <= '1'; x4 <= '0';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
            severity error;

        x4 <= '1';
        wait for 10 ns;
        assert y1 = '0'
            report "output y1 is wrong!"
            severity error;
        assert y2 = '0'
            report "output y2 is wrong!"
            severity error;
    end process;
end arc_proj1_tb;

```

```

        severity error;

x2 <= '1'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x1 <= '1'; x2 <= '0'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

```

```

x4 <= '1';
wait for 10 ns;
    assert y1 = '1'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x2 <= '1'; x3 <= '0'; x4 <= '0';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x4 <= '1';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;

x3 <= '1'; x4 <= '0';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '1'
        report "output y2 is wrong!"
        severity error;

x4 <= '1';
wait for 10 ns;
    assert y1 = '0'
        report "output y1 is wrong!"
        severity error;
    assert y2 = '0'
        report "output y2 is wrong!"
        severity error;
wait;
end process;
end arc_proj1_tb;

configuration cfg_proj1_tb of proj1_tb is
    for arc_proj1_tb
        end for;
end cfg_proj1_tb;

```

Test bench1 for project1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity proj1_tb is
end proj1_tb;

architecture arc_proj1_tb of proj1_tb is

    type sample is record
        x1 : bit;
        x2 : bit;
        x3 : bit;
        x4 : bit;
        y1 : bit;
        y2 : bit;
    end record;

    type sample_array is array (natural range <>) of sample;
    constant test_data : sample_array :=
    (
--      x1   x2   x3   x4   y1   y2
        ('0', '0', '0', '0', '0', '0'),
        ('0', '0', '0', '1', '0', '0'),
        ('0', '0', '1', '0', '0', '0'),
        ('0', '0', '1', '1', '0', '0'),
        ('0', '1', '0', '0', '1', '1'),
        ('0', '1', '0', '1', '1', '1'),
        ('0', '1', '1', '0', '1', '1'),
        ('0', '1', '1', '1', '1', '1'),
        ('1', '0', '0', '0', '1', '1'),
        ('1', '0', '0', '1', '1', '1'),
        ('1', '0', '1', '0', '1', '1'),
        ('1', '0', '1', '1', '1', '1'),
        ('1', '1', '0', '0', '0', '0'),
        ('1', '1', '0', '1', '0', '0'),
        ('1', '1', '1', '0', '0', '1'),
        ('1', '1', '1', '1', '0', '0')
    );

    component proj1
        port (x1, x2, x3, x4 : in bit;
              y1, y2 : out bit);
    end component;

    signal x1, x2, x3, x4, y1, y2 : bit;

begin
    design: proj1 port map (x1, x2, x3, x4, y1, y2);
    process
    begin
        for i in test_data'range loop
            x1 <= test_data(i).x1;
            x2 <= test_data(i).x2;
            x3 <= test_data(i).x3;
            x4 <= test_data(i).x4;
            wait for 10 ns;
            assert y1 = test_data(i).y1
                report "output y1 is wrong!"
                severity error;
            assert y2 = test_data(i).y2
                report "output y2 is wrong!"
                severity error;
        end loop
    end process
end
```

```
        end loop;
        wait;
    end process;
    end arc_proj1_tb;
configuration cfg_proj1_tb of proj1_tb is
    for arc_proj1_tb
        end for;
end cfg_proj1_tb;
```