```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity gate_4 is
  port(w, x, y, z: in std_logic; f: out std_logic);
end gate_4;

-- nor4
  architecture nor4_basic of gate_4 is
    component nor_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component nor_2;

    signal wire0, wire1, wire2, wire3: std_logic;

    begin
    nor2_0: nor_2 port map(w, x, wire0);
    nor2_1: nor_2 port map(wire0, wire0, wire1);

    nor2_3: nor_2 port map(y, z, wire2);
    nor2_4: nor_2 port map(wire2, wire2, wire3);

    nor2_5: nor_2 port map(wire1, wire3, f);
  end nor4_basic;

  architecture nor4_primitive of gate_4 is
    component or_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component or_2;

    component not_1 is
      port(x: in std_logic; f: out std_logic);
    end component not_1;

    signal wire0, wire1, wire2: std_logic;

    begin
    or2_0: or_2 port map(w, x, wire0);
    or2_1: or_2 port map(y, z, wire1);
```

```
    or2_3: or_2 port map(wire0, wire1, wire2);

    not1_0: not_1 port map(wire2, f);
  end nor4_primitive;

  architecture nor4_behavior of gate_4 is
    begin
    f <= '0' when (w = '1' or x = '1' or y = '1' or z = '1')
      else '1';
  end nor4_behavior;

  architecture nor4_equation of gate_4 is
    begin
    f <= NOT (w OR x OR y OR z);
  end nor4_equation;

-- or4
  architecture or4_basic of gate_4 is
    component or_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component or_2;

    signal wire0, wire1: std_logic;

    begin
    or2_0: or_2 port map(w, x, wire0);
    or2_1: or_2 port map(y, z, wire1);

    or2_3: or_2 port map(wire0, wire1, f);
  end or4_basic;

  architecture or4_primitive of gate_4 is
    component or_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component or_2;

    signal wire0, wire1: std_logic;

    begin
```

```
    or2_0: or_2 port map(w, x, wire0);
    or2_1: or_2 port map(y, z, wire1);

    or2_3: or_2 port map(wire0, wire1, f);
  end or4_primitive;

  architecture or4_behavior of gate_4 is
    begin
    f <= '1' when (w = '1' OR x = '1' OR y = '1' OR z = '1')
      else '0';
  end or4_behavior;

  architecture or4_equation of gate_4 is
    begin
    f <= (w OR x OR y OR z);
  end or4_equation;

-- xor
  architecture xor4_basic of gate_4 is
    component xor_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component xor_2;

    signal wire0, wire1: std_logic;

    begin
    xor_0: xor_2 port map(w, x, wire0);
    xor_1: xor_2 port map(y, z, wire1);

    xor_3: xor_2 port map(wire0, wire1, f);
  end xor4_basic;

  architecture xor4_primitive of gate_4 is
    component and_2 is
      port(x, y: in std_logic; f: out std_logic);
    end component and_2;

    component not_1 is
      port(x: in std_logic; f: out std_logic);
```

```
    end component not_1;

    signal wire00, wire01, wire02, wire03, wire04, wire05, wire06, wire07:
std_logic;
    signal wire10, wire11, wire12, wire13, wire14, wire15, wire16, wire17:
std_logic;
    signal wire20, wire21, wire22, wire23, wire24, wire25, wire26:
std_logic;

    begin
      -- W XOR X
      and02_0: and_2 port map(w, x, wire00);
      not01_0: not_1 port map(wire00, wire01); -- W NAND X

      and02_1: and_2 port map(w, wire01, wire02); -- W AND (W NAND X)
      and02_2: and_2 port map(x, wire01, wire03); -- X AND (W NAND X)

      not01_1: not_1 port map(wire02, wire04); -- W NAND (W NAND X)
      not01_2: not_1 port map(wire03, wire05); -- X NAND (W NAND X)

      and02_3: and_2 port map(wire04, wire05, wire06); -- (W NAND (W NAND
X)) AND (X NAND (W NAND X))
      not01_3: not_1 port map(wire06, wire07); -- wire07 = W XOR X

      -- W XOR X XOR Y (wire07 XOR Y)
      and12_0: and_2 port map(wire07, y, wire10);
      not11_0: not_1 port map(wire10, wire11);

      and12_1: and_2 port map(wire07, wire11, wire12);
      and12_2: and_2 port map(y, wire11, wire13);

      not11_1: not_1 port map(wire12, wire14);
      not11_2: not_1 port map(wire13, wire15);

      and12_3: and_2 port map(wire14, wire15, wire16);
      not11_3: not_1 port map(wire16, wire17); -- W XOR X XOR Y

      -- W XOR X XOR Y XOR Z (wire17 XOR Z)
      and22_0: and_2 port map(wire17, z, wire20);
```

```
    not21_0: not_1 port map(wire20, wire21);

    and22_1: and_2 port map(wire17, wire21, wire22);
    and22_2: and_2 port map(z, wire21, wire23);

    not21_1: not_1 port map(wire22, wire24);
    not21_2: not_1 port map(wire23, wire25);

    and22_3: and_2 port map(wire24, wire25, wire26);
    not21_3: not_1 port map(wire26, f); -- W XOR X XOR Y XOR Z

end xor4_primitive;

architecture xor4_behavior of gate_4 is
  begin
  f <= '1' when (w = '1' XOR x = '1' XOR y = '1' XOR z = '1')
    else '0';
end xor4_behavior;

architecture xor4_equation of gate_4 is
  begin
  f <= (w XOR x XOR y XOR z);
end xor4_equation;
```

Figure 1. VHDL for various 4 input gates

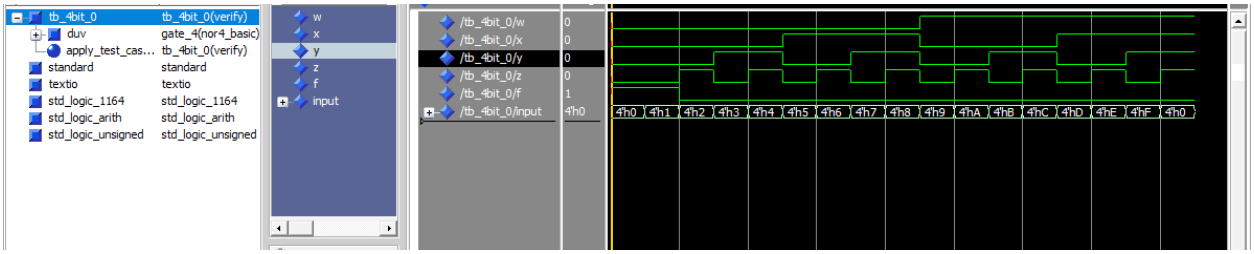

Figure 2. nor4 basic

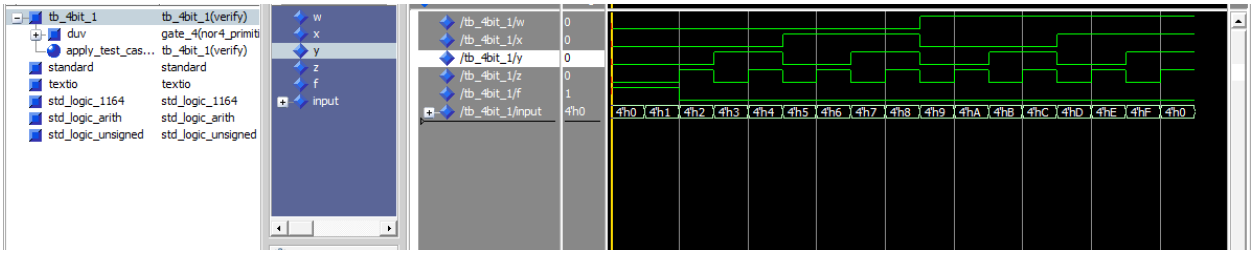

Figure 3. nor4 primitive

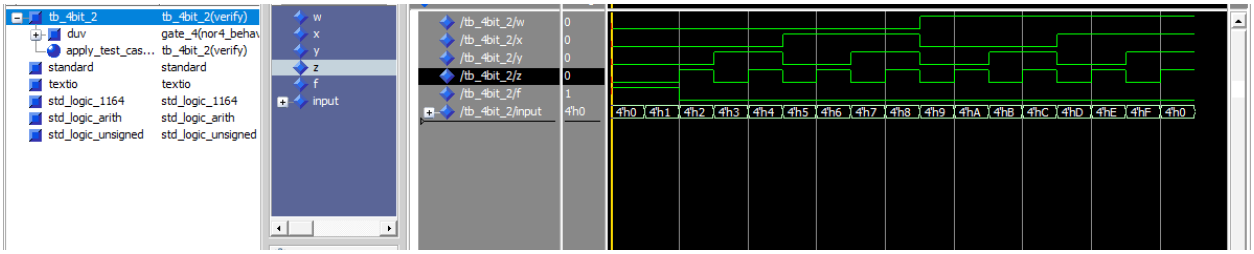

Figure 4. nor4 behavior

Figure 5. nor4 equation

Figure 6. or4 basic

Figure 7. or4 primitive

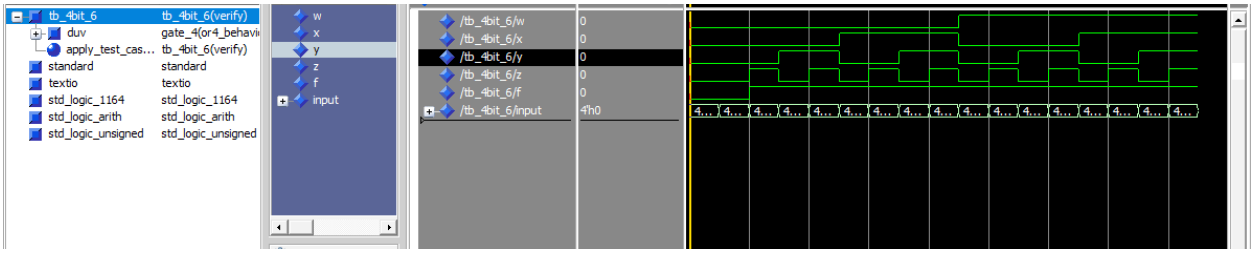

Figure 8. or4 behavior

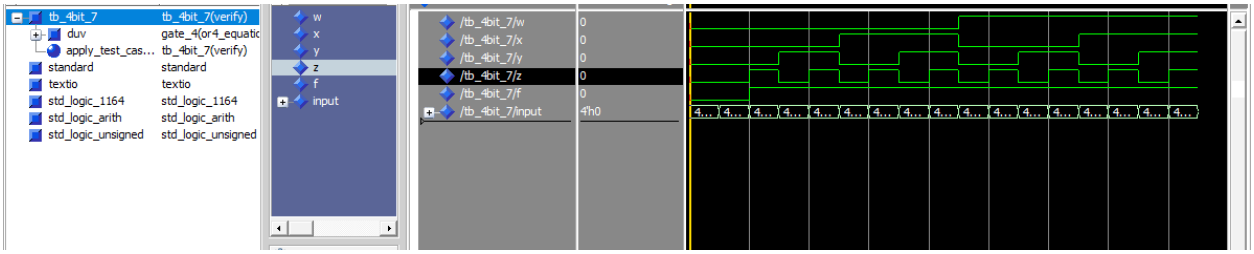

Figure 9. or4 equation

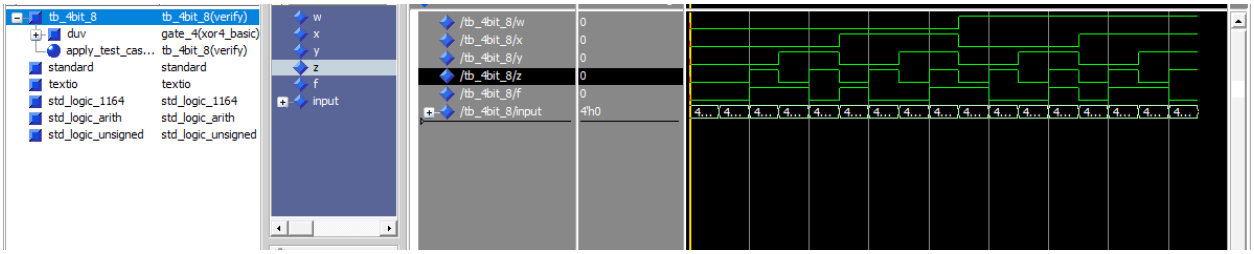

Figure 10. xor basic

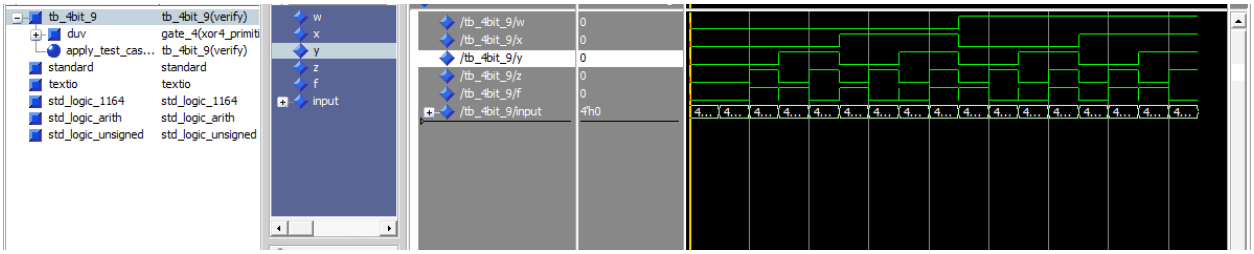

Figure 11. xor primitive

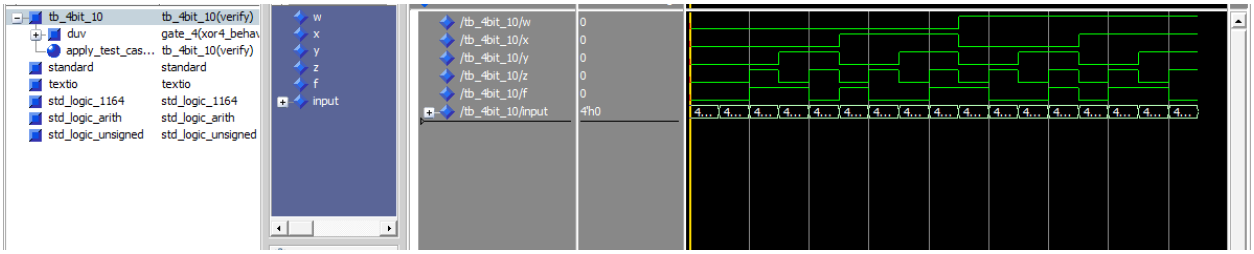

Figure 12. xor behavior

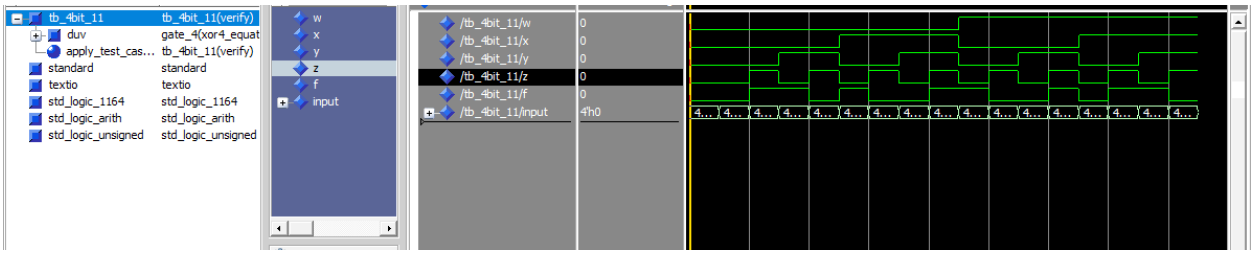

Figure 13. xor equation

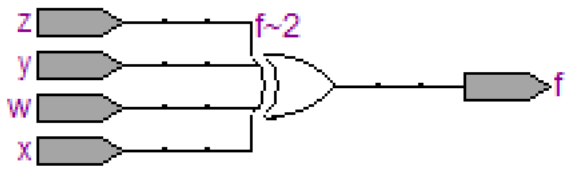

Figure 14. xor equation RTL

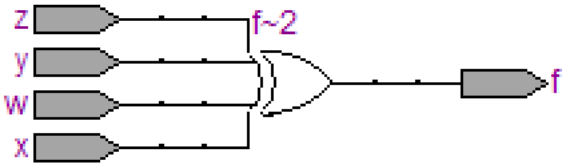

Figure 15. xor behavior RTL

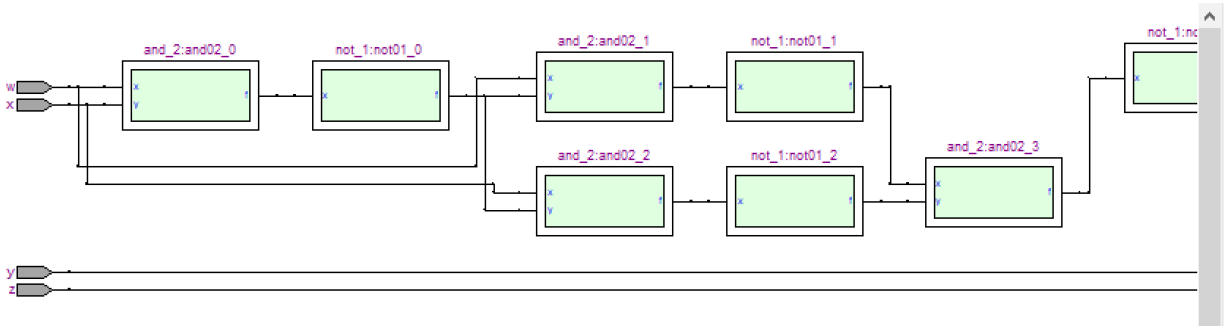

Figure 16. xor primitive RTL

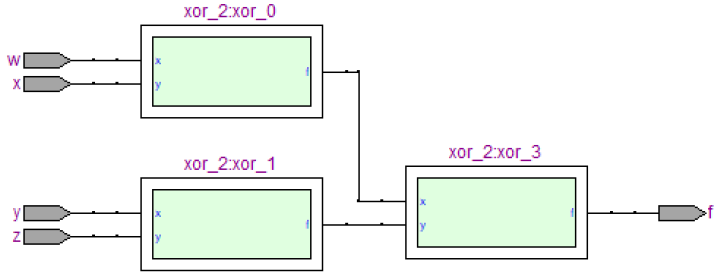

Figure 17. xor basic RTL

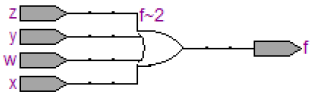

Figure 18. or equation RTL

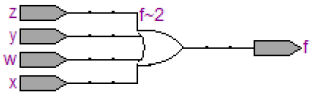

Figure 19. or behavior RTL

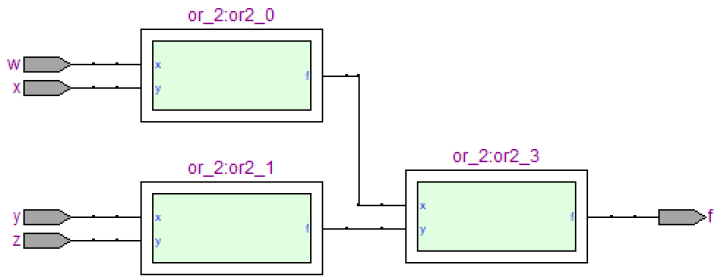

Figure 20. or primitive RTL

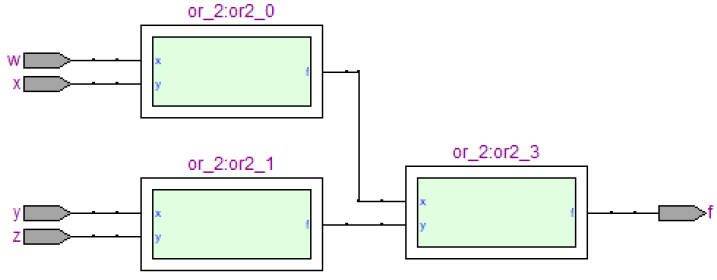

Figure 21. or basic RTL

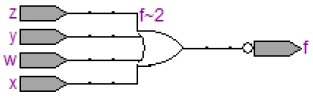

Figure 22. nor equation RTL

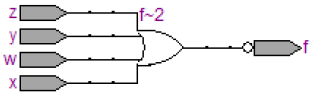

Figure 23. nor behavior RTL

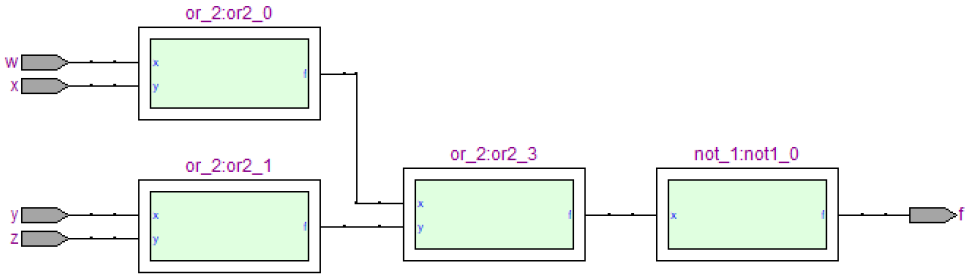

Figure 24. nor primitive RTL

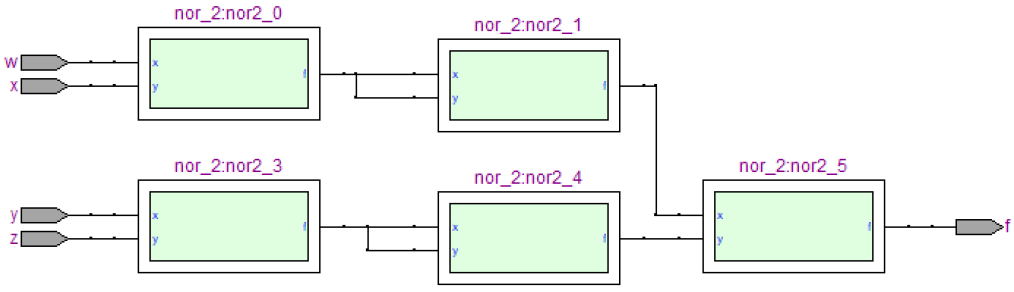

Figure 25. nor basic RTL