```
module Add_half (input a, b,  output c_out, sum),
  xor G1(sum, a, b);                                    // Gate instance names are
optional
  and G2(c_out, a, b);
endmodule

module Add_full (input a, b, c_in, output c_out, sum);        // See Fig. 4.8
  wire w1, w2, w3;                                  // w1 is c_out; w2 is sum
  Add_half M1 (a, b, w1, w2);
  Add_half M0 (w2, c_in, w3, sum);
  or (c_out, w1, w3);
endmodule

module Add_rca_4 (input [3:0] a, b, input c_in output c_out, output [3:0] sum);
  wire c_in1, c_in3, c_in4;                         // Intermediate carries
  Add_full M0 (a[0], b[0], c_in,   c_in1, sum[0]);
  Add_full M1 (a[1], b[1], c_in1, c_in2, sum[1]);
  Add_full M2 (a[2], b[2], c_in2, c_in3, sum[2]);
  Add_full M3 (a[3], b[3], c_in3, c_out, sum[3]);
endmodule

module Add_rca_8 (input [7:0] a, b, input c_in, output c_out, output [7:0] sum,)
  wire c_in4;
  Add_rca_4 M0 (a[3:0], b[3:0], c_in, c_in4, sum[3:0]);
  Add_rca_4 M1 (a[7:4], b[7:4], c_in4, c_out, sum[7:4]);
endmodule
```

```
module Add_half (input a, b,  output c_out, sum),
  xor G1(sum, a, b);                                    // Gate instance names are
optional
  and G2(c_out, a, b);
endmodule

module Add_full (input a, b, c_in, output c_out, sum);        // See Fig. 4.8
  wire w1, w2, w3;                                  // w1 is c_out; w2 is sum
  Add_half M1 (a, b, w1, w2);
  Add_half M0 (w2, c_in, w3, sum);
  or (c_out, w1, w3);
endmodule

module Add_rca_4 (input [3:0] a, b, input c_in output c_out, output [3:0] sum);
  wire c_in1, c_in3, c_in4;                         // Intermediate carries
  Add_full M0 (a[0], b[0], c_in,   c_in1, sum[0]);
  Add_full M1 (a[1], b[1], c_in1, c_in2, sum[1]);
  Add_full M2 (a[2], b[2], c_in2, c_in3, sum[2]);
  Add_full M3 (a[3], b[3], c_in3, c_out, sum[3]);
endmodule

module Add_rca_8 (input [7:0] a, b, input c_in, output c_out, output [7:0] sum,)
  wire c_in4;
  Add_rca_4 M0 (a[3:0], b[3:0], c_in, c_in4, sum[3:0]);
  Add_rca_4 M1 (a[7:4], b[7:4], c_in4, c_out, sum[7:4]);
endmodule
```

```
-- Dataflow description of four-bit adder
entity binary_adder is
port (Sum: out Std_Logic_Vector (3 downto 0); C_out: out Std_Logic;
        A, B: in Std_Logic_Vector (3 downto 0); C_in: in Std_Logic);
end binary_adder;

architecture Dataflow of binary_adder is
   begin
   C_out & Sum <= A + B + ('000' & C_in);          -- Compatible word sizes
end Dataflow;
```

```
-- Dataflow description of four-bit adder
entity binary_adder is
port (Sum: out Std_Logic_Vector (3 downto 0); C_out: out Std_Logic;
            A, B: in Std_Logic_Vector (3 downto 0); C_in: in Std_Logic);
end binary_adder;

architecture Dataflow of binary_adder is
   begin
   C_out & Sum <= A + B + ('000' & C_in);          -- Compatible word sizes
end Dataflow;
```

```
entity bufif1_vhdl is
   port (buf_in, control,: in Std_Logic; buf_out: out Std_Logic);
end bufif1_vhdl;


architecture Dataflow of bufif1_vhdl is
begin
   buf_out <= buf_in when control = '1' else 'z';
end Dataflow;

entity notif1 is
   port (not_in, control: in Std_Logic; not_out: out Std_Logic);
end notif1;

architecture Dataflow of notif1 is
begin
   not_out <= not (not_in)  when control = '1' else z;
end architecture;
```