Code

1-A

```
module assign_1_A(A,B,C,F0,F1,F2,F3);
  input A,B,C; // input
  wire W1,W2,W3,W4,W5,W6; // wire 정의 (중간 node), wire는 W1, W2...로 naming 하기!
  output F0,F1,F2,F3;

  and G1 (W1, ~A, ~B); // (out wire, in, in), gate는 G1, G2... 로 naming 하기!
  and G2 (W2, A, ~C);
  and G3 (W4, B, ~C);
  and G4 (W5, A, C);

  or G5 (F0, W1, W2);
  or G6 (F1, W2, B);
  or G7 (F2, W1, W4);
  or G8 (F3, W4, W5);

  // assign F3 = W4 | W5; 이런 식으로도 정의 가능하데!

endmodule
```

```
module assign_1_A_tb;

  // Inputs
  reg A, B, C; // input으로 바꾸면 안돼!

  // Outputs
  wire F0, F1, F2, F3; // output으로 바꾸면 안돼!

  assign_1_A U0(
    .A(A),
    .B(B),
    .C(C),
    .F0(F0),
    .F1(F1),
    .F2(F2),
    .F3(F3)
  );

  initial begin
    A = 0; B = 0; C = 0;
    #10;
    $display("Test Case 1: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 0; B = 0; C = 1;
    #10;
    $display("Test Case 2: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 0; B = 1; C = 0;
    #10;
    $display("Test Case 3: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 0; B = 1; C = 1;
    #10;
    $display("Test Case 4: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 1; B = 0; C = 0;
    #10;
```

```
    $display("Test Case 5: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 1; B = 0; C = 1;
    #10;
    $display("Test Case 6: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 1; B = 1; C = 0;
    #10;
    $display("Test Case 7: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);

    A = 1; B = 1; C = 1;
    #10;
    $display("Test Case 8: A=%b, B=%b, C=%b, F0=%b, F1=%b, F2=%b, F3=%b", A, B, C, F0, F1, F2, F3);


    $finish;
  end
endmodule
```

1-B

```
// ======== full adder ========
module assign_1_B_FullAdder (
  input A, B, C_, // A, B, 아래서 올라온 carry
  output S, C // sum, carry
);
  wire w1, w2, w3;

  xor G1 (w1, A, B);
  xor G2 (S, w1, C_);

  and G3 (w2, A, B);
  and G4 (w3, w1, C_);
  or G5 (C, w2, w3);
endmodule

// ======== 4b parallel adder ========

module assign_1_B_4bParallelAdder (
  input [3:0] A, B,
  output [3:0] Sum, Cout
);
  wire [3:0] w_sum;
  wire w_carry1, w_carry2, w_carry3;

  // 모듈 사용은 주로 U 사용하는 듯!!
  assign_1_B_FullAdder U0 (.A(A[0]), .B(B[0]), .C_(1'b0), .S(w_sum[0]), .C(w_carry1));
  assign_1_B_FullAdder U1 (.A(A[1]), .B(B[1]), .C_(w_carry1), .S(w_sum[1]), .C(w_carry2));
  assign_1_B_FullAdder U2 (.A(A[2]), .B(B[2]), .C_(w_carry2), .S(w_sum[2]), .C(w_carry3));
  assign_1_B_FullAdder U3 (.A(A[3]), .B(B[3]), .C_(w_carry3), .S(w_sum[3]), .C(Cout));

  // 결과 출력
  assign Sum = w_sum;

endmodule
```

```
module assign_1_B_4bParallelAdder_tb;

  reg [3:0] A, B;

  wire [3:0] Sum;
  wire [3:0] Cout;
```

```
  assign_1_B_4bParallelAdder U0 (
    .A(A),
    .B(B),
    .Sum(Sum),
    .Cout(Cout)
  );

  initial begin
    for (int i = 0; i < 8; i=i+1) begin
      A = $random;
      B = $random;
      #10;
      $display("Test Case %0d: A=%b, B=%b, Sum=%b, Cout=%b", i+1, A, B, Sum, Cout);
    end

    $finish;
  end

endmodule
```

1-C

```
module assign_1_C (
  output reg Q,
  input D, Clk, PreN, ClrN);

  always @ (posedge Clk, negedge PreN, negedge ClrN) // posedge: positive edge, negedge: negative edge
    if (!ClrN) Q <= 1'b0;
  else if (!PreN) Q <= 1'b1;
    else Q <= D;
endmodule
```

```
module assign_1_C_tb;
  reg D, Clk, PreN, ClrN;

  wire Q;

  assign_1_C U0 (
    .Q(Q),
    .D(D),
    .Clk(Clk),
    .PreN(PreN),
    .ClrN(ClrN)
  );

  initial begin
    // Test case 1: No preset or clear, normal operation
    D = 1'b0; Clk = 0; PreN = 1; ClrN = 1;
    #5 Clk = 1;
    #5;
    $display("Test Case 1: D=%b, Q=%b", D, Q);

    // Test case 2: Clear activated
    D = 1'b1; Clk = 0; PreN = 1; ClrN = 0;
    #5 Clk = 1;
    #5;
    $display("Test Case 2: D=%b, Q=%b", D, Q);

    // Test case 3: Preset activated
    D = 1'b0; Clk = 0; PreN = 0; ClrN = 1;
    #5 Clk = 1;
```

```
    #5;
    $display("Test Case 3: D=%b, Q=%b", D, Q);

    // Test case 4: Data changes on rising edge
    D = 1'b1; Clk = 0; PreN = 1; ClrN = 1;
    #5 Clk = 1; // Rising edge
    #5;
    $display("Test Case 4: D=%b, Q=%b", D, Q);

    // Test case 5: Data changes on falling edge
    D = 1'b0; Clk = 1; PreN = 1; ClrN = 1;
    #5 Clk = 0; // Falling edge
    #5;
    $display("Test Case 5: D=%b, Q=%b", D, Q);

    $finish;
  end
endmodule
```

1-D

```
module assign_1_C (
  output reg Q,
  input D, Clk, PreN, ClrN
);

  always @(posedge Clk or negedge PreN or negedge ClrN) begin
    if (!ClrN) Q <= 1'b0;
    else if (!PreN) Q <= 1'b1;
    else Q <= D;
  end
endmodule

module assign_1_B_FullAdder (
  input A, B, C_, // A, B, 아래서 올라온 carry
  output S, C // sum, carry
);
  wire w1, w2, w3;

  xor G1 (w1, A, B);
  xor G2 (S, w1, C_);

  and G3 (w2, A, B);
  and G4 (w3, w1, C_);
  or G5 (C, w2, w3);
endmodule

module SerialAdder (
  output reg [3:0] sum_out,
  output wire carry_out,
  input wire clk,
  input wire rst,
  input wire [3:0] in_a,
  input wire [3:0] in_b
);

  reg [3:0] d_out;
  wire carry_in = 1'b0;
  reg carry0 = 1'b0;
  wire carry1 = 1'b0;
  wire carry2 = 1'b0;
  wire carry3 = 1'b0;
  wire carry4 = 1'b0;
  wire carry5 = 1'b0;
```

```
  // Full adder instances
  assign_1_B_FullAdder U1 (
    .A(in_a[0]),
    .B(in_b[0]),
    .C_(carry_in),
    .S(sum_out[0]),
    .C(carry0)
  );

  assign_1_C U11 (
    .Q(carry1),
    .D(carry0),
    .Clk(clk),
    .PreN(rst),
    .ClrN(~rst)
  );

  assign_1_B_FullAdder U2 (
    .A(in_a[1]),
    .B(in_b[1]),
    .C_(carry1),
    .S(sum_out[1]),
    .C(carry2)
  );

  assign_1_C U12 (
    .Q(carry3),
    .D(carry2),
    .Clk(clk),
    .PreN(rst),
    .ClrN(~rst)
  );

  assign_1_B_FullAdder U3 (
    .A(in_a[2]),
    .B(in_b[2]),
    .C_(carry3),
    .S(sum_out[2]),
    .C(carry4)
  );

  assign_1_C U13 (
    .Q(carry5),
    .D(carry4),
    .Clk(clk),
    .PreN(rst),
    .ClrN(~rst)
  );

  assign_1_B_FullAdder U4 (
    .A(in_a[3]),
    .B(in_b[3]),
    .C_(carry5),
    .S(sum_out[3]),
    .C(carry_out)
  );
endmodule
```

```
module SerialAdder_tb;

  reg clk;
  reg rst;
  reg [3:0] in_a;
  reg [3:0] in_b;
  wire [3:0] sum_out;
```

```
  SerialAdder U0 (
    .sum_out(sum_out),
    .carry_out(carry_out),
    .clk(clk),
    .rst(rst),
    .in_a(in_a),
    .in_b(in_b)
  );

  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end

  initial begin

    in_a = 4'b0000;
    in_b = 4'b0011;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 1: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b1100;
    in_b = 4'b0011;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 2: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b1000;
    in_b = 4'b0011;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 3: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b0011;
    in_b = 4'b0011;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 4: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b1000;
    in_b = 4'b1001;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 5: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b1111;
    in_b = 4'b1111;
```

```
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 6: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b0000;
    in_b = 4'b1111;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 7: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    in_a = 4'b0001;
    in_b = 4'b1011;

    @(posedge clk);
    @(posedge clk);
    @(posedge clk);
    @(posedge clk);

    $display("Test Case 8: Input A: %b, Input B: %b, Sum: %b, Carry_out: %b", in_a, in_b, sum_out, carry_out);

    $stop;
  end
endmodule
```

2

```
module flash_adc_decoder (
  input wire [3:0] COMP,
  output wire [1:0] B
);

  wire W1, W2, W3;

  nand G1 (W1, COMP[3], ~COMP[2]);
  nand G2 (W2, COMP[2], ~COMP[1]);
  nand G3 (W3, COMP[1], ~COMP[0]);

  nand G4 (B[1], W1, W2);
  nand G5 (B[0], W1, W3);

endmodule
```

```
module flash_adc_decoder_tb;
  reg [3:0] COMP;

  wire [1:0] B;

  flash_adc_decoder u1 (
    .COMP(COMP),
    .B(B)
  );

  // Testbench
  initial begin
    // Case 0
```

```
    COMP = 4'b1111;
    #10 $display("Test Case 0: COMP = %b, B = %b", COMP, B);

    // Case 1
    COMP = 4'b1110;
    #10 $display("Test Case 1: COMP = %b, B = %b", COMP, B);

    // Case 2:
    COMP = 4'b1100;
    #10 $display("Test Case 2: COMP = %b, B = %b", COMP, B);

    // Case 3:
    COMP = 4'b1000;
    #10 $display("Test Case 3: COMP = %b, B = %b", COMP, B);

    // Case 4:
    COMP = 4'b0000;
    #10 $display("Test Case 4: COMP = %b, B = %b", COMP, B);


    $stop;
  end
endmodule
```