**Course Code: EE488**

**Assignment -6**

**PREPARED BY**

**Khandoker Samiul Hoque**

**Student Id: 19837**

**Github url:**

**Question No 1 Answer:**

**Module:**

module CarryLookAhead(a, b, Ci, sum, cout);

input [7:0] a, b;input Ci;

output [7:0] sum;output cout;

wire [8:0] p, g; wire [7:0] C;

assign p = {1'b0, a} ^ {1'b0, b} ^ {1'b0, Ci};

assign g = {1'b0, a} & {1'b0, b} | ({1'b0, a} ^ {1'b0, b}) & {1'b0, Ci};

assign C[0] = Ci;

genvar i;

generate

for (i = 1; i < 8; i = i + 1)

begin : EightBits\_LkAhead\_Gen

assign C[i] = p[i] | g[i] & C[i - 1];

end

endgenerate

assign

sum = a + b + Ci;

assign

cout = g[7] | p[7] & C[6];

endmodule

**TestBench:**

module CarryLookAhead\_tb();

reg [7:0] a, b;

reg Ci;

wire [7:0] sum;

wire cout;

CarryLookAhead Assign6\_prob1(.a(a), .b(b), .Ci(Ci),.sum(sum),.cout(cout));

initial begin

$dumpfile("CarryLookAhead\_tb.vcd");

$dumpvars;

a = 8'b00000000;b = 8'b00000000;Ci = 0;

#1

$display("Test case 1: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00000001; b = 8'b00000001; Ci = 0;

#1

$display("Test case 2: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00000011; b = 8'b00000011; Ci = 0;

#1

$display("Test case 3: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00000111; b = 8'b00000111; Ci = 0;

#1

$display("Test case 4: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00001111; b = 8'b00001111; Ci = 0;

#1

$display("Test case 5: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00011111; b = 8'b00011111; Ci = 0;

#1

$display("Test case 6: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b00111111; b = 8'b00111111; Ci = 0;

#1

$display("Test case 7: a=%b, b=%b, Ci=%b, sum=%b , cout=%b", a, b, Ci, sum, cout);

#1

a = 8'b01111111; b = 8'b01111111; Ci = 0;

#1

$display("Test case 8: a=%b, b=%b, Ci=%b, sum=%b , Ci=%b", a, b, Ci, sum, cout);

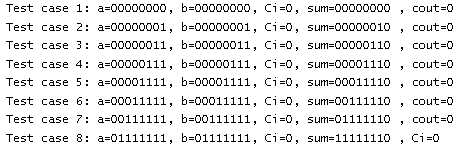
#1

$finish();

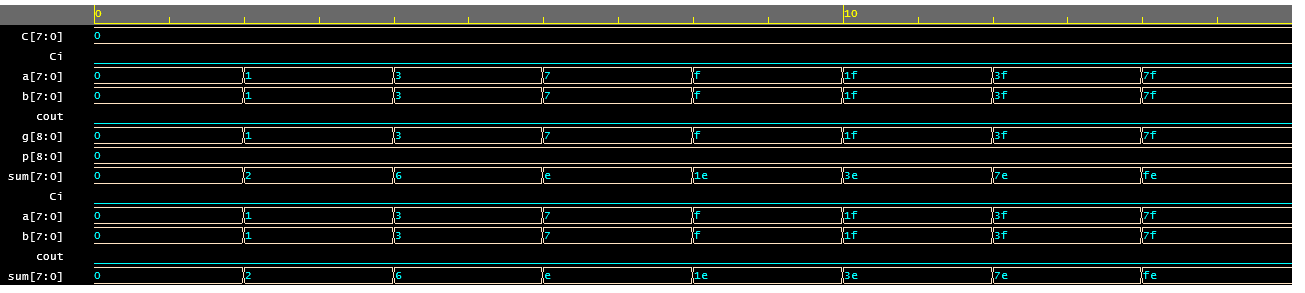
end

endmodule

**Output :**



**Signal Wave:**

****

**Question No 2 Answer:**

**Module:**

module Alu8bitsopcode(input [7:0] A,input [7:0] B,input [3:0] opcode,

output reg [7:0] out);

wire [7:0] propagate\_carry;

genvar i;

generate

wire [7:0]

generate\_carry;

assign

generate\_carry = 1'b0;

for (i = 0; i < 8; i = i + 1) begin : lookahead

assign

generate\_carry[i] = A[i] & B[i];

end

for (i = 1; i < 8; i = i + 1) begin : propagate

assign

propagate\_carry[i] = A[i-1] | B[i-1];

end

assign propagate\_carry[0] = 1'b0;

endgenerate

always @(\*) begin

case(opcode)

// Addition

4'b0000: out = A + B;

// Subtraction

4'b0001: out = A - B;

4'b0010: out = A \* B;

4'b0011: out = A / B;

4'b0100: out = A << 1;

4'b0101: out = A >> 1;

4'b0110: out = {A[6:0], A[7]};

4'b0111: out = {A[0], A[7:1]};

4'b1000: out = A & B;

4'b1001: out = A | B;

4'b1010: out = A ^ B;

4'b1011: out = ~(A | B);

4'b1100: out = ~(A & B);

4'b1101: out = ~(A ^ B);

// Greater than

4'b1110: out = (A > B) ? 1'b1 : 1'b0;

// Equal to

4'b1111: out = (A == B) ? 1'b1 : 1'b0;

default: out = 15'b0;

endcase

end

endmodule

**TestBench:**

module Alu8bitsopcode\_TB;

reg [7:0] A;reg [7:0] B;reg [3:0] opcode;

wire [7:0] out;

Alu8bitsopcode assign6\_prob2(.A(A), .B(B), .opcode(opcode), .out(out));

initial begin

$dumpfile("tb.vcd");

$dumpvars;

// Addition

A = 8'b01010101;

B = 8'b10101010;

opcode = 4'b0000;

#10;

$display("Output is:",);

$display("A + B = %b", out);

// Subtraction

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b0001;

#5;

$display("A - B = %b", out);

A = 8'b01010101;

B = 8'b10101010;

opcode = 4'b0010;

#5;

$display("A \* B = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b0011;

#5;

$display("A / B = %b", out);

A = 8'b10101010;

opcode = 4'b0100;

#5;

$display("A << 1 = %b", out);

A = 8'b10101010;

opcode = 4'b0101;

#5;

$display("A >> 1 = %b", out);

A = 8'b10101010;

opcode = 4'b0110;

#5;

$display("{A[6:0], A[7]} = %b", out);

A = 8'b10101010;

opcode = 4'b0111;

#5;

$display("{A[0], A[7:1]} = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1000;

#5;

$display("A & B = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1001;

#5;

$display("A | B = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1010;

#5;

$display("A ^ B = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1011;

#5;

$display("~(A | B) = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1100;

#5;

$display("~(A & B) = %b", out);

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1101;

#5;

$display("~(A ^ B) = %b", out);

// Greater than

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1110;

#5;

$display("A > B = %b",out);

// Equal to

A = 8'b10101010;

B = 8'b01010101;

opcode = 4'b1111;

#5;

$display("A == B = %b", out);

end

endmodule

**Output:**

