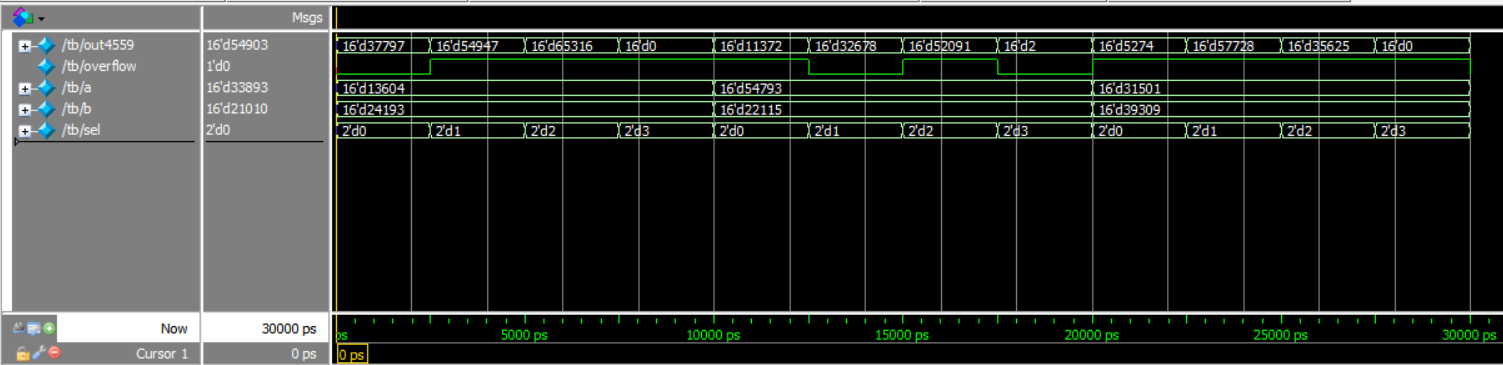
**Project 1 : ALU Design**

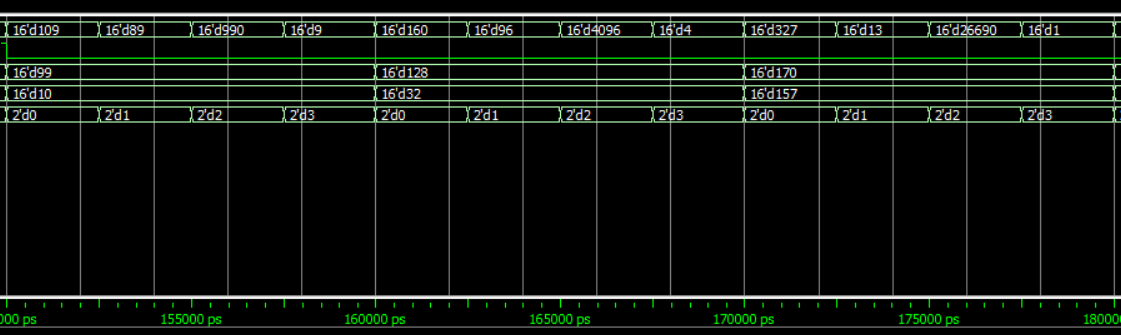
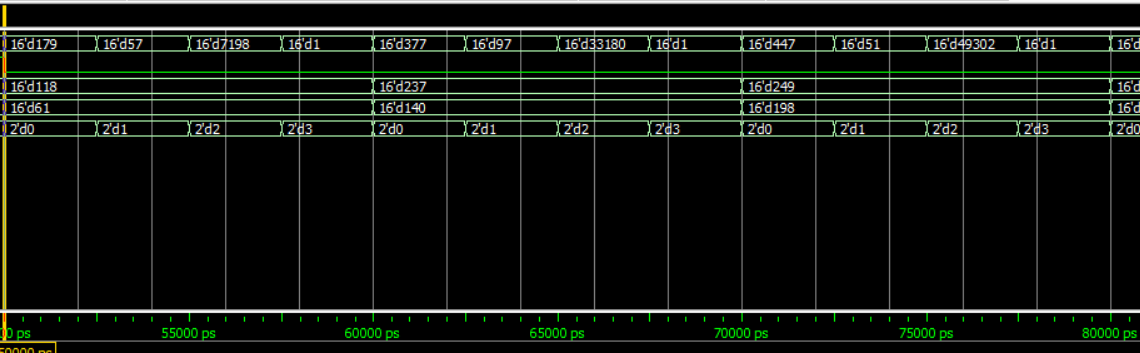
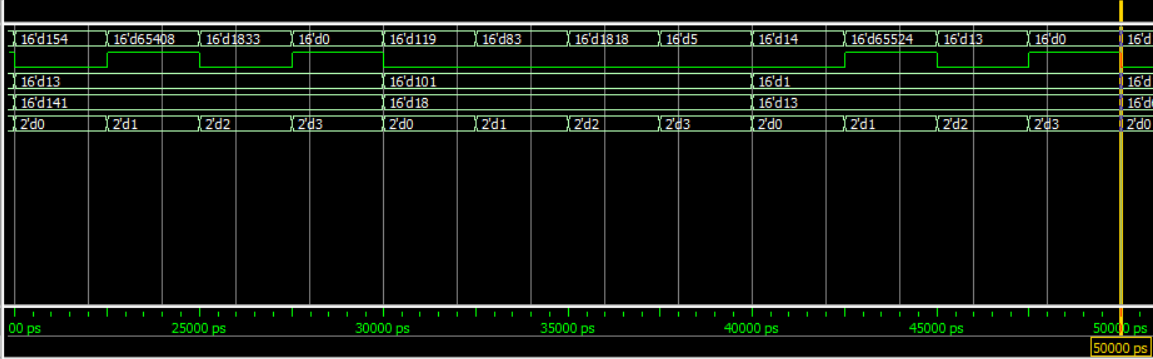
2017114559 박선우

**Waveform result**

( a = $random , b = $random )



( a = {$random}%256 , b = {$random}%256 )



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input1** |  | **Input2** | **Correct output** | **My output** | **Overflow** |
| 13604 | + | 24193 | 37797 | 37797 | 0 |
|  | - |  | -10589 | 54947 | 1 |
|  | \* |  | 329,121,572 | 65316 | 1 |
|  | / |  | 0 | 0 | 1 |
| 54793 | + | 22115 | 76908 | 11372 | 1 |
|  | - |  | 32678 | 32678 | 0 |
|  | \* |  | 1,211,747,195 | 52091 | 1 |
|  | / |  | 2 | 2 | 0 |
| 31501 | + | 39309 | 70810 | 5274 | 1 |
|  | - |  | -7808 | 57728 | 1 |
|  | \* |  | 1,238,272,809 | 34525 | 1 |
|  | / |  | 0 | 0 | 1 |
| 13 | + | 141 | 154 | 154 | 0 |
|  | - |  | -128 | 65408 | 1 |
|  | \* |  | 1833 | 1833 | 0 |
|  | / |  | 0 | 0 | 1 |
| 101 | + | 18 | 119 | 119 | 0 |
|  | - |  | 83 | 83 | 0 |
|  | \* |  | 1818 | 1818 | 0 |
|  | / |  | 5 | 5 | 0 |
| 1 | + | 13 | 14 | 14 | 0 |
|  | - |  | -12 | 65524 | 1 |
|  | \* |  | 13 | 13 | 0 |
|  | / |  | 0 | 0 | 1 |
| 118 | + | 61 | 179 | 179 | 0 |
|  | - |  | 57 | 57 | 0 |
|  | \* |  | 7198 | 7198 | 0 |
|  | / |  | 1 | 1 | 0 |
| 237 | + | 140 | 377 | 377 | 0 |
|  | - |  | 97 | 97 | 0 |
|  | \* |  | 33180 | 33180 | 0 |
|  | / |  | 1 | 1 | 0 |
| 249 | + | 198 | 447 | 447 | 0 |
|  | - |  | 51 | 51 | 0 |
|  | \* |  | 49302 | 49302 | 0 |
|  | / |  | 1 | 1 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 99 | + | 10 | 109 | 109 | 0 |
|  | - |  | 89 | 89 | 0 |
|  | \* |  | 990 | 990 | 0 |
|  | / |  | 9 | 9 | 0 |
| 128 | + | 32 | 160 | 160 | 0 |
|  | - |  | 96 | 96 | 0 |
|  | \* |  | 4096 | 4096 | 0 |
|  | / |  | 4 | 4 | 0 |
| 170 | + | 157 | 327 | 327 | 0 |
|  | - |  | 13 | 13 | 0 |
|  | \* |  | 26690 | 26690 | 0 |
|  | / |  | 1 | 1 | 0 |

In + overflow there are same result in my output + 65536 and correct output

In – overflow there are same result in my output – 65536 and correct output

In \* overflow the result is other value because the alu can show only 16bits

In division if the result is smaller than 1 the output is 0 and overflow is 1

The meaning of 65536 is the the max value in 16bit number = 2^16

**Code**

tb.v

`timescale 1ns/100ps

module tb();

wire[15:0] out4559;

wire overflow;

reg[15:0] a,b;

reg[1:0] sel;

alu a1(a,b,sel,out4559,overflow);

initial begin

a=$random;

b=$random;

sel=0;

end

always begin

#2.5 sel = sel+1;

end

always begin

#10 a=$random;

b=$random;

end

endmodule

to change sel value in 2.5ns

unsigned 16bit input a,b , 2bit input sel

unsigned 16bit output out4559 ,

1bit output overflow

Initialize random a,b and sel = 0

In every 2.5ns add 1 in sel value

In every 10ns set random value a,b

alu.v

module alu(input[15:0] a,b,input[1:0] sel, output[15:0] out4559, output overflow);

wire[15:0] addout,subout,mulout,divout;

wire addof,subof,mulof,divof;

add a1(a,b,addout,addof);

sub s1(a,b,subout,subof);

mul m1(a,b,mulout,mulof);

div d1(a,b,divout,divof);

mux41 mux1(addout,subout,mulout,divout,sel,out4559);

bit1mux41 mux2(addof,~subof,mulof,divof,sel,overflow);

endmodule

I set add,sub,mul,div module in alu and select output with 41mux

multiandgate.v

for calculate 16bits AND

module multiandgate(input[15:0] a,input b, output[15:0] y);

and a0(y[0],a[0],b);

and a1(y[1],a[1],b);

and a2(y[2],a[2],b);

and a3(y[3],a[3],b);

and a4(y[4],a[4],b);

and a5(y[5],a[5],b);

and a6(y[6],a[6],b);

and a7(y[7],a[7],b);

and a8(y[8],a[8],b);

and a9(y[9],a[9],b);

and a10(y[10],a[10],b);

and a11(y[11],a[11],b);

and a12(y[12],a[12],b);

and a13(y[13],a[13],b);

and a14(y[14],a[14],b);

and a15(y[15],a[15],b);

endmodule

use in mul.v

add.v , bit16adder.v , bit4adder.v , bit1fulladder.v

module add(input[15:0] a,b, output[15:0] out, output overflow);

bit16adder a1(a, b, 0, out, overflow);

endmodule

module bit16adder(input[15:0] a, b,input c\_in, output[15:0] sum, output c\_out);

wire c1,c2,c3;

bit4adder a0(a[3:0],b[3:0],c\_in,sum[3:0],c1);

bit4adder a1(a[7:4],b[7:4],c1,sum[7:4],c2);

bit4adder a2(a[11:8],b[11:8],c2,sum[11:8],c3);

bit4adder a3(a[15:12],b[15:12],c3,sum[15:12],c\_out);

endmodule

module bit4adder(input[3:0] a, b,input c\_in, output[3:0] sum, output c\_out);

wire c1,c2,c3;

bit1fulladder a0(a[0],b[0],c\_in,sum[0],c1);

bit1fulladder a1(a[1],b[1],c1,sum[1],c2);

bit1fulladder a2(a[2],b[2],c2,sum[2],c3);

bit1fulladder a3(a[3],b[3],c3,sum[3],c\_out);

endmodule

module bit1fulladder(input a,b,c\_in, output sum, c\_out);

wire s1,s2,c1;

and a1(c1,a,b), a2(s2,s1,c\_in);

xor x1(s1,a,b), x2(sum,s1,c\_in), x3(c\_out,s2,c1);

endmodule

I make bit1adder first and make bit4adder with 4 bit1fulladder, and make bit16adder with

4 bit4adder, and I set input, output in bit16adder.

In adder c\_in is 0

sub.v

module sub(input[15:0] a,b, output[15:0] out, output overflow);

bit16adder a1(a, ~b, 1, out, overflow);

endmodule

I recycle bit16adder in sub, set c\_in 1 and input inverse b

mul.v

module mul(input[15:0] a,b, output[15:0] out, output overflow);

wire[15:0] a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15;

wire[15:0] o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14;

wire c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15;

wire r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r16,r17,r18,r19,r20,r21,r22,r23,r24,r25,r26,r27,r28;

multiandgate u0(a,b[0],a0);

multiandgate u1(a,b[1],a1);

multiandgate u2(a,b[2],a2);

multiandgate u3(a,b[3],a3);

multiandgate u4(a,b[4],a4);

multiandgate u5(a,b[5],a5);

multiandgate u6(a,b[6],a6);

multiandgate u7(a,b[7],a7);

multiandgate u8(a,b[8],a8);

multiandgate u9(a,b[9],a9);

multiandgate u10(a,b[10],a10);

multiandgate u11(a,b[11],a11);

multiandgate u12(a,b[12],a12);

multiandgate u13(a,b[13],a13);

multiandgate u14(a,b[14],a14);

multiandgate u15(a,b[15],a15);

bit16adder add0(a0, a1<<1, 0, o1, c1);

bit16adder add1(o1, a2<<2, 0, o2, c2);

bit16adder add2(o2, a3<<3, 0, o3, c3);

bit16adder add3(o3, a4<<4, 0, o4, c4);

bit16adder add4(o4, a5<<5, 0, o5, c5);

bit16adder add5(o5, a6<<6, 0, o6, c6);

bit16adder add6(o6, a7<<7, 0, o7, c7);

bit16adder add7(o7, a8<<8, 0, o8, c8);

bit16adder add8(o8, a9<<9, 0, o9, c9);

bit16adder add9(o9, a10<<10, 0, o10, c10);

bit16adder add10(o10, a11<<11, 0, o11, c11);

bit16adder add11(o11, a12<<12, 0, o12, c12);

bit16adder add12(o12, a13<<13, 0, o13, c13);

bit16adder add13(o13, a14<<14, 0, o14, c14);

bit16adder add14(o14, a15<<15, 0, out, c15);

I make multigate module for 16bit AND calculate

or or1(r1,|(a15>>1),|(a14>>2));

or or2(r2,r1,|(a13>>3));

or or3(r3,r2,|(a12>>4));

or or4(r4,r3,|(a11>>5));

or or5(r5,r4,|(a10>>6));

or or6(r6,r5,|(a9>>7));

or or7(r7,r6,|(a8>>8));

or or8(r8,r7,|(a7>>9));

or or9(r9,r8,|(a6>>10));

or or10(r10,r9,|(a5>>11));

or or11(r11,r10,|(a4>>12));

or or12(r12,r11,|(a3>>13));

or or13(r13,r12,|(a2>>14));

or or14(r14,r13,|(a1>>15));

or or15(r15,r14,c1);

or or16(r16,r15,c2);

or or17(r17,r16,c3);

or or18(r18,r17,c4);

or or19(r19,r18,c5);

or or20(r20,r19,c6);

or or21(r21,r20,c7);

or or22(r22,r21,c8);

or or23(r23,r22,c9);

or or24(r24,r23,c10);

or or25(r25,r24,c11);

or or26(r26,r25,c12);

or or27(r27,r26,c13);

or or28(r28,r27,c14);

or or29(overflow,r28,c15);

endmodule

AND calculate a,b each bits

Make a0 ~ a15 [0:15]

And add all a0 ~ a15 with left shift operator

The result is output

And I did OR calculate for overflow

I did all carry c1~c15 and a1~a15 with right shift

div.v

module div(input[15:0] a,b, output reg[15:0] out, output reg overflow);

reg[31:0] R;

always@(a,b)begin

R = 0;

R = a;

if(a<b)begin

out = 0;

overflow = 1;

end

else begin

repeat(16) begin

R = R<<1;

if(R[31:16]>=b) begin

R[0] = 1;

R[31:16] = R[31:16]-b;

end

end

out = R[15:0];

overflow = 0;

end

end

endmodule

I make 32bit reg R

And I use always@(\*)

to change my output.

First initialize R

And compare a,b

If a<b

output = 0 , overflow = 1

else

go repeat 16times loop

in loop first right shift R and

check R[31:16] >= b

if true set R[0] = 1 and do subtract R[31:16] – b

after loop set

output= R[15:0], overflow= 0

mux41.v , bit1mux41.v

module mux41(a,b,c,d,sel,y);

input [15:0] a,b,c,d;

input [1:0] sel;

output reg [15:0] y;

always@(\*) begin

case(sel)

2'b00: y=a;

2'b01: y=b;

2'b10: y=c;

2'b11: y=d;

endcase

end

endmodule

module bit1mux41(a,b,c,d,sel,y);

input a,b,c,d;

input [1:0] sel;

output reg y;

always@(\*) begin

case(sel)

2'b00: y=a;

2'b01: y=b;

2'b10: y=c;

2'b11: y=d;

endcase

end

endmodule

for select add,sub,mul,div by sel value

mux41 is 16bits

bit1mux41 is 1bit