**하드웨어 시스템 설계 5주차 실습 보고서**

2017-12751 컴퓨터공학부 이동학

Goal: Implement BRAM model, Processing Element & test bench according to scenario.

Code:

My\_bram.v

module my\_bram #(

parameter integer BRAM\_ADDR\_WIDTH = 15,

parameter INIT\_FILE = "input.txt",

parameter OUT\_FILE = "output.txt"

)(

input wire [BRAM\_ADDR\_WIDTH-1:0] BRAM\_ADDR,

input wire BRAM\_CLK,

input wire [31:0] BRAM\_WRDATA,

output reg [31:0] BRAM\_RDDATA,

input wire BRAM\_EN,

input wire BRAM\_RST,

input wire [3:0] BRAM\_WE,

input wire done

);

reg [31:0] mem[0:8191];

wire [BRAM\_ADDR\_WIDTH-3:0] addr = BRAM\_ADDR[BRAM\_ADDR\_WIDTH-1:2];

reg [31:0] dout;

reg [31:0] readREG[1:0];

reg readREG\_V[1:0];

reg [BRAM\_ADDR\_WIDTH-3:0] addrREG;

reg [31:0] writeREG;

reg [3:0] writeREG\_V;

wire [31:0] mask;

initial begin

if (INIT\_FILE != "") begin

$readmemh(INIT\_FILE, mem);

end

wait (done) begin

$writememh(OUT\_FILE, mem);

end

end

assign mask = {{8{writeREG\_V[0]}}, {8{writeREG\_V[1]}},{8{writeREG\_V[2]}}, {8{writeREG\_V[3]}}};

always @(posedge BRAM\_CLK) begin

if(BRAM\_RST == 1) begin

BRAM\_RDDATA <= 0;

end

else begin

if(readREG\_V[1] == 1) begin

BRAM\_RDDATA = readREG[1];

end

mem[addrREG] = (mem[addrREG] & (~mask)) | (writeREG & mask);

readREG\_V[1] = readREG\_V[0];

readREG[1] = readREG[0];

readREG\_V[0] = BRAM\_EN & (!BRAM\_WE);

readREG[0] = mem[addr];

writeREG\_V = {4{BRAM\_EN}} & BRAM\_WE;

addrREG = addr;

writeREG = BRAM\_WRDATA;

end

end

endmodule

tb\_bram.v

module tb\_bram(

);

reg BRAM\_CLK;

parameter BRAM\_ADDR\_WIDTH = 15;

reg [BRAM\_ADDR\_WIDTH-1:0] BRAM\_ADDR;

wire [31:0] BRAM\_RDDATA1;

wire [31:0] BRAM\_RDDATA2;

reg BRAM\_EN1;

reg BRAM\_EN2;

reg BRAM\_RST;

reg [3:0] BRAM\_WE;

reg done;

integer i;

initial begin

BRAM\_CLK <= 0;

BRAM\_EN1 <= 0;

BRAM\_EN2 <= 0;

BRAM\_RST <= 0;

done <= 0;

BRAM\_WE <= 0;

for(i=0;i<8192;i=i+1) begin

#10;

BRAM\_ADDR <= {i, 2'b00};

BRAM\_EN1 <= 1;

BRAM\_EN2 <= 0;

#10;

BRAM\_EN1 <= 0;

#20;

BRAM\_EN2 <= 1;

BRAM\_WE <= 4'b1111;

#10;

BRAM\_EN2 <= 0;

BRAM\_WE <= 0;

#10

BRAM\_EN2 <= 1;

end

#10

#100;

BRAM\_EN2 <= 0;

done <= 1;

$finish;

end

always #5 BRAM\_CLK = ~BRAM\_CLK;

my\_bram read\_BRAM(

.BRAM\_ADDR(BRAM\_ADDR),

.BRAM\_CLK(BRAM\_CLK),

.BRAM\_WRDATA(32'b0),

.BRAM\_RDDATA(BRAM\_RDDATA1),

.BRAM\_EN(BRAM\_EN1),

.BRAM\_RST(BRAM\_RST),

.BRAM\_WE(4'b0),

.done(0)

);

my\_bram #(BRAM\_ADDR\_WIDTH, "", "output.txt") write\_BRAM(

.BRAM\_ADDR(BRAM\_ADDR),

.BRAM\_CLK(BRAM\_CLK),

.BRAM\_WRDATA(BRAM\_RDDATA1),

.BRAM\_RDDATA(BRAM\_RDDATA2),

.BRAM\_EN(BRAM\_EN2),

.BRAM\_RST(BRAM\_RST),

.BRAM\_WE(BRAM\_WE),

.done(done)

);

Endmodule

My\_pe.v

module my\_pe #(

parameter L\_RAM\_SIZE = 6

)

(

input aclk,

input aresetn,

input [31:0] ain,

input [31:0] din,

input [L\_RAM\_SIZE-1:0] addr,

input we,

input valid,

output dvalid,

output [31:0] dout

);

(\* ram\_style = "block" \*) reg [31:0] peram [0:2\*\*L\_RAM\_SIZE - 1];

reg [31:0] psum;

wire [31:0] buffer;

always @(posedge aclk) begin

if(we == 1) peram[addr] <= din;

if(~aresetn) psum <= 0;

if(dvalid == 1) psum = buffer;

end

assign dout = (dvalid == 1) ? psum : 32'b0;

floating\_point\_0 mac(

.aclk(aclk),

.aresetn(aresetn),

.s\_axis\_a\_tdata(ain),

.s\_axis\_a\_tvalid(valid),

.s\_axis\_b\_tdata(peram[addr]),

.s\_axis\_b\_tvalid(valid),

.s\_axis\_c\_tdata(psum),

.s\_axis\_c\_tvalid(valid),

.m\_axis\_result\_tdata(buffer),

.m\_axis\_result\_tvalid(dvalid)

);

Endmodule

Tb\_pe.v

module tb\_pe(

);

parameter L\_RAM\_SIZE = 6;

reg aclk;

reg aresetn;

reg [31:0] ain;

reg [31:0] din;

reg [31:0] mem [15:0];

reg [L\_RAM\_SIZE - 1:0] addr;

reg we;

reg valid;

wire dvalid;

wire [31:0] dout;

integer i;

initial begin

aclk <= 0;

aresetn <= 0;

we <= 1;

valid <= 0;

$readmemh("din.txt", mem);

#10;

for(i = 0; i < 16; i = i + 1) begin

addr <= i;

din <= mem[i];

#10;

end

$readmemh("ain.txt", mem);

we <= 0;

aresetn <= 1;

#10;

for(i = 0; i < 16; i = i + 1) begin

addr <= i;

ain <= mem[i];

valid <= 1;

#10;

valid <= 0;

#160;

end

$finish;

end

always #5 aclk = ~aclk;

my\_pe PE(

.aclk(aclk),

.aresetn(aresetn),

.ain(ain),

.din(din),

.addr(addr),

.we(we),

.valid(valid),

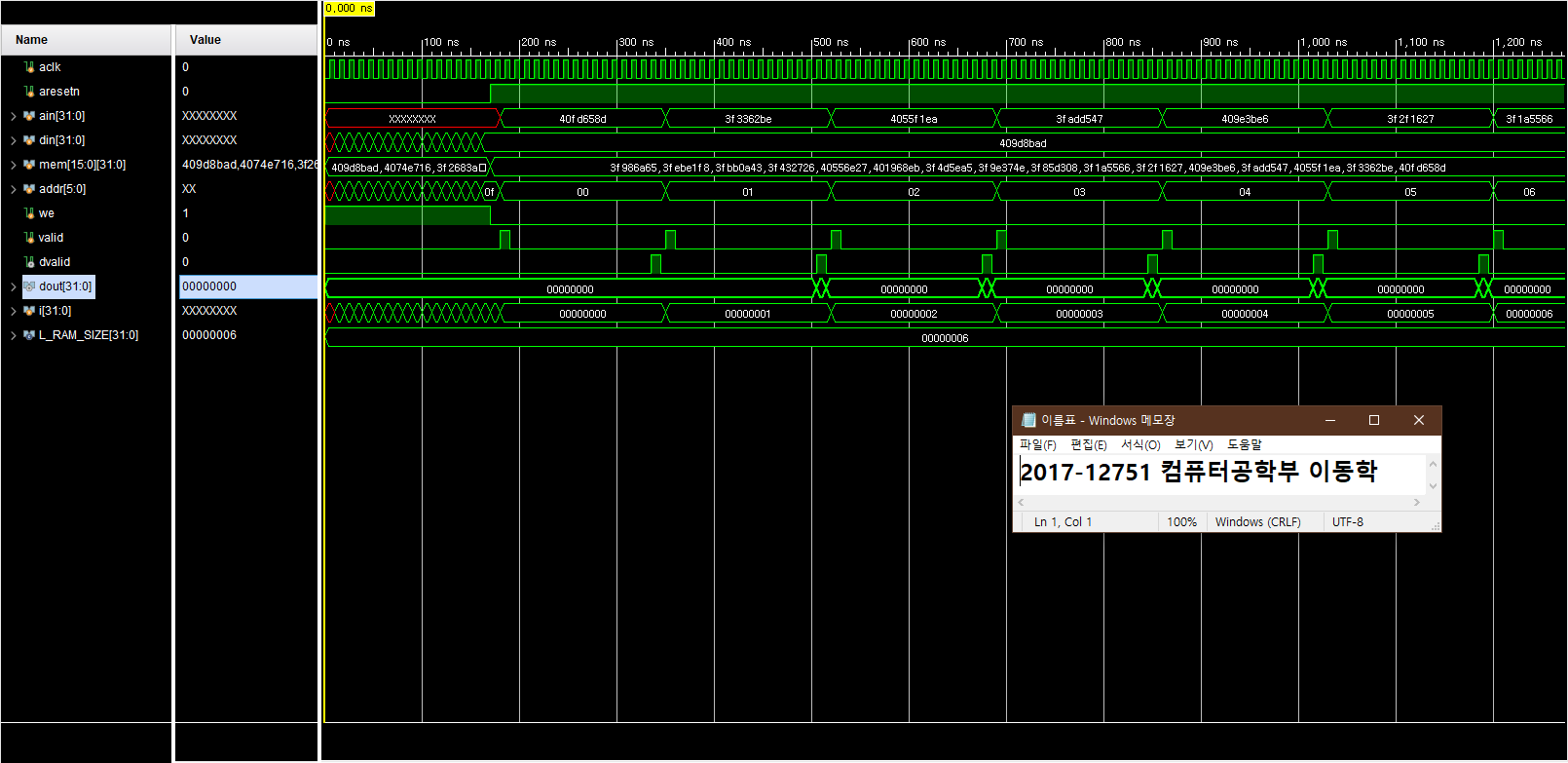
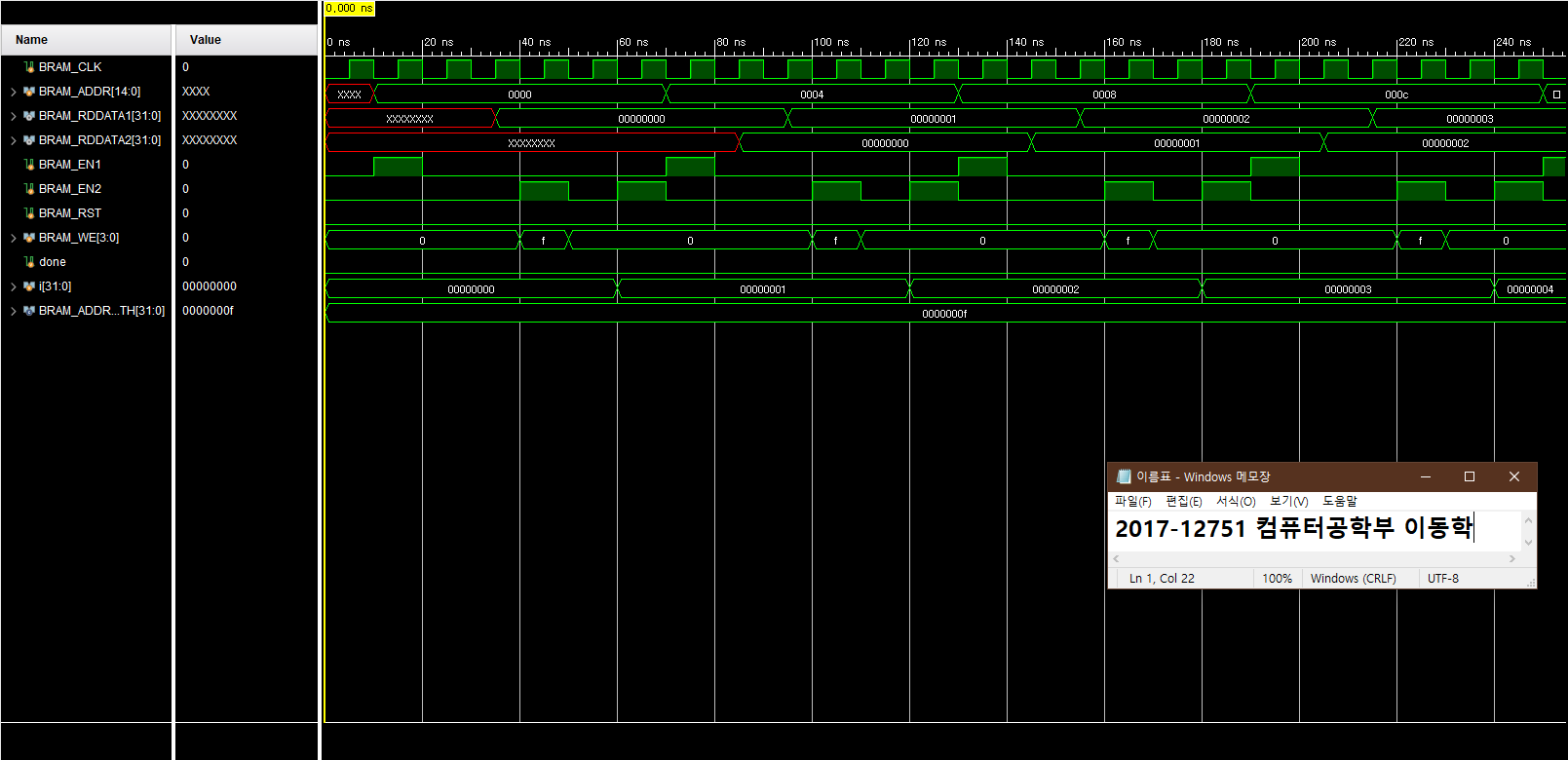
.dvalid(dvalid),

.dout(dout)

);

endmodule

Result: my\_bram / my\_pe



Discussion: my\_bram의 경우 read BRAM은 input.txt에서 값을 불러와 initialize 하였고, write BRAM은 read BRAM의 모든 값을 복사하도록 하였습니다. Write에 1cycle, read에 2cycle이 사용되도록 read register 2개와 write register 1개를 선언하고 posedge clk마다 한칸씩 밀리도록 구현하였습니다. 모든 과정이 끝난 후엔 output.txt에 write BRAM에 저장된 값을 출력하도록 하였습니다.

My\_pe의 경우 처음엔 din.txt에서 16개의 값을 불러와 local register의 알맞은 주소에 저장하고, 그 이후 ain.txt에서 하나씩 값을 읽어서 IP catalog를 이용해 MAC operation을 수행하고, 중간 결과를 저장하여 다음 연산에 사용하도록 하였습니다.