module ASC (

clk,

reset,

slave\_signals,

i\_done,

o\_start,

o\_color,

o\_x0,

o\_y0,

o\_x1,

o\_y1

)

(

input clk,

input reset,

input [38:0] slave\_signals,

input i\_done,

output o\_start,

output [2:0] o\_color,

output [8:0] o\_x0,

output [7:0] o\_y0,

output [8:0] o\_x1,

output [7:0] o\_y1,

);

wire base\_address = 0x00700000; // how do i write into memory address

assign i\_done = base\_address;

assign o\_start = base\_address + 0’x8;

assign o\_x0 = base\_address + 0’xC;

assign o\_y0 = base\_address + 0’xC + 0’b9;

assign o\_x1 = base\_address + 0’x10;

assign o\_y1 = base\_address + 0’x10 + 0’b9;

assign o\_start = base\_address + o’x8;

enum int unsigned

{

S\_stall,

S\_poll

} state, nextstate;

Always\_ff @ (posedge clk or posedge reset) begin  
 if (reset) begin

case (state)  
S\_stall:  
 if (o\_start) begin  
 i\_done <= 1’b0;  
 state <= S\_poll;  
 end

S\_poll:  
 o\_color <= slave\_signals [4:2];

o\_x0 <= slave\_signals [13:5];

o\_y0 <= slave\_signals [21:14];

o\_x1 <= slave\_signals [30:22];

o\_y1 <= slave\_signals [38:31];

o\_start <= 1‘b1;

state <= S\_stall;

end

endcase  
end

endmodule