module apb\_mock\_uart

#(

parameter APB\_ADDR\_WIDTH = 12 //APB slaves are 4KB by default

)

(

input logic CLK,

input logic RSTN,

input logic [APB\_ADDR\_WIDTH-1:0] PADDR,

input logic [31:0] PWDATA,

input logic PWRITE,

input logic PSEL,

input logic PENABLE,

output logic [31:0] PRDATA,

output logic PREADY,

output logic PSLVERR,

output logic INT, //Interrupt output

output logic OUT1N, // Output 1

output logic OUT2N, // Output 2

output logic RTSN, // RTS output

output logic DTRN, // DTR output

input logic CTSN, // CTS input

input logic DSRN, // DSR input

input logic DCDN, // DCD input

input logic RIN, // RI input

input logic SIN, // Receiver input

output logic SOUT // Transmitter output

);

// enum { THR = 0, RBR = 0, DLL = 0, IER = 1, DLM = 1, IIR = 2, FCR = 2, LCR = 3, MCR, LSR, MSR, SCR} uart\_regs;

logic [3:0] register\_adr;

logic [7:0][7:0] regs\_q, regs\_n;

assign register\_adr = PADDR[2:0];

// UART Registers

// register write logic

always\_comb

begin

regs\_n = regs\_q;

if (PSEL && PENABLE && PWRITE)

begin

regs\_n[$unsigned(register\_adr)] = PWDATA[7:0];

end

regs\_n[8'h5] = 32'h60;

end

// register read logic

always\_comb

begin

PRDATA = 'b0;

if (PSEL && PENABLE && !PWRITE)

begin

PRDATA = {24'b0, regs\_q[$unsigned(register\_adr)] };

end

end

// synchronouse part

always\_ff @(posedge CLK, negedge RSTN)

begin

if(~RSTN)

begin

regs\_q <= '{8{8'b0}};

end

else

begin

regs\_q <= regs\_n;

if (PSEL && PENABLE && PWRITE)

begin

if ($unsigned(register\_adr) == 0)

begin

$write("%C", PWDATA[7:0]);

end

end

end

end

// APB logic: we are always ready to capture the data into our regs

// not supporting transfare failure

assign PREADY = 1'b1;

assign PSLVERR = 1'b0;

endmodule