**APB Protocol**

module APB\_Master (

input wire clk,

input wire reset,

input wire[1:0] cmd\_i,

output wire psel\_o,

output wire penable\_o,

output wire[31:0] paddr\_o,

output wire pwrite\_o,

output wire[31:0] pwdata\_o,

input wire pready\_i,

input wire[31:0] prdata\_i

);

// Enum for the APB state

typedef enum logic[1:0] {ST\_IDLE = 2'b00, ST\_SETUP = 2'b01, ST\_ACCESS = 2'b10} apb\_state\_t;

apb\_state\_t nxt\_state;

apb\_state\_t state\_q;

logic[31:0] rdata\_q;

always\_ff @(posedge clk or posedge reset)

if (reset)

state\_q <= ST\_IDLE;

else

state\_q <= nxt\_state;

always\_comb begin

nxt\_state = state\_q;

case (state\_q)

ST\_IDLE : if (|cmd\_i) nxt\_state = ST\_SETUP; else nxt\_state = ST\_IDLE;

ST\_SETUP : nxt\_state = ST\_ACCESS;

ST\_ACCESS : begin

if (pready\_i) nxt\_state = ST\_IDLE;

end

default : nxt\_state = state\_q;

endcase

end

assign psel\_o = (state\_q == ST\_SETUP) | (state\_q == ST\_ACCESS);

assign penable\_o = (state\_q == ST\_ACCESS);

assign pwrite\_o = cmd\_i[1];

assign paddr\_o = 32'hDEAD\_CAFE;

assign pwdata\_o = rdata\_q + 32'h1;

// Capture the read data to store it for the next write

always\_ff @(posedge clk or posedge reset)

if (reset)

rdata\_q <= 32'h0;

else if (penable\_o && pready\_i)

rdata\_q <= prdata\_i;

endmodule

**Test Bench Code**

module tb\_APB\_master ();

logic clk;

logic reset;

logic[1:0] cmd\_i;

logic psel\_o;

logic penable\_o;

logic[31:0] paddr\_o;

logic pwrite\_o;

logic[31:0] pwdata\_o;

logic pready\_i;

logic[31:0] prdata\_i;

// Instantiate the RTL

APB\_Master DAY16 (.\*);

// Generate clock

always begin

clk = 1'b0;

#5;

clk = 1'b1;

#5;

end

int wait\_cycles;

// Generate pready

always begin

pready\_i = 1'b0;

wait\_cycles = $urandom\_range (1, 10);

while (wait\_cycles) begin

@(posedge clk);

wait\_cycles--;

end

pready\_i = 1'b1;

@(posedge clk);

end

// Generate the stimulus

initial begin

reset <= 1'b1;

cmd\_i <= 2'b00;

prdata\_i <= 32'h0;

@(posedge clk);

reset <= 1'b0;

@(posedge clk);

@(posedge clk);

for (int i=0; i<10; i++) begin

cmd\_i <= i%2 ? 2'b10 : 2'b01;

prdata\_i <= $urandom\_range(0, 4'hF);

// Wait for pready to be asserted

while (~pready\_i | ~psel\_o) @(posedge clk);

@(posedge clk);

end

$finish();

end

// Dump VCD

initial begin

$dumpfile("day16.vcd");

$dumpvars(0, tb\_APB\_master);

end

endmodule

**Simulation**

