// AHB to APG Bridge | Maven Silicon

//

//

//

// AHB Slave Interface

// Date:04-06-2022

//

//

// Modifications: The Combinational part sensitivity list did not inclued Hresetn and hence they gave x output on reset

module AHB\_slave\_interface(Hclk,Hresetn,Hwrite,Hreadyin,Htrans,Haddr,Hwdata,Prdata,valid,Haddr1,Haddr2,Hwdata1,Hwdata2,Hwritereg,tempselx);

input Hclk,Hresetn;

input Hwrite,Hreadyin;

input [1:0] Htrans;

input [31:0] Haddr,Hwdata,Prdata;

output reg valid;

output reg [31:0] Haddr1,Haddr2,Hwdata1,Hwdata2;

output reg Hwritereg;

output reg [2:0] tempselx;

/// Implementing Pipeline Logic for Address,Data and Control Signal

/// Implementing Pipeline Logic for Address, Data, and Control Signal

always @(posedge Hclk)

begin

if (~Hresetn)

begin

Haddr1 <= 0;

Haddr2 <= 0;

end

else

begin

Haddr1 <= Haddr;

Haddr2 <= Haddr1;

end

end

always @(posedge Hclk)

begin

if (~Hresetn)

begin

Hwdata1 <= 0;

Hwdata2 <= 0;

end

else

begin

Hwdata1 <= Hwdata;

Hwdata2 <= Hwdata1;

end

end

always @(posedge Hclk)

begin

if (~Hresetn)

Hwritereg <= 0;

else

Hwritereg <= Hwrite;

end

/// Implementing Valid Logic Generation

always @(Hreadyin, Haddr, Htrans, Hresetn) begin

valid = 0;

if (Hresetn && Hreadyin && (Haddr >= 32'h8000\_0000 && Haddr < 32'h8C00\_0000) && (Htrans == 2'b10 || Htrans == 2'b11))

valid = 1;

end

/// Implementing Tempselx Logic

always @(Haddr, Hresetn)

begin

tempselx = 3'b000;

if (Hresetn && Haddr >= 32'h8000\_0000 && Haddr < 32'h8400\_0000)

tempselx = 3'b001;

else if (Hresetn && Haddr >= 32'h8400\_0000 && Haddr < 32'h8800\_0000)

tempselx = 3'b010;

else if (Hresetn && Haddr >= 32'h8800\_0000 && Haddr < 32'h8C00\_0000)

tempselx = 3'b100;

end

endmodule

// AHB to APG Bridge | Maven Silicon

//

//

//

// APB FSM Controller

// Date:08-06-2022

//

// By-Prajwal Kumar Sahu

module APB\_FSM\_Controller( Hclk,Hresetn,valid,Haddr1,Haddr2,Hwdata1,Hwdata2,Prdata,Hwrite,Haddr,Hwdata,Hwritereg,tempselx,

Pwrite,Penable,Pselx,Paddr,Pwdata,Hreadyout);

input Hclk,Hresetn,valid,Hwrite,Hwritereg;

input [31:0] Hwdata,Haddr,Haddr1,Haddr2,Hwdata1,Hwdata2,Prdata;

input [2:0] tempselx;

output reg Pwrite,Penable;

output reg Hreadyout;

output reg [2:0] Pselx;

output reg [31:0] Paddr,Pwdata;

//////////////////////////////////////////////////////PARAMETERS

parameter ST\_IDLE=3'b000;

parameter ST\_WWAIT=3'b001;

parameter ST\_READ= 3'b010;

parameter ST\_WRITE=3'b011;

parameter ST\_WRITEP=3'b100;

parameter ST\_RENABLE=3'b101;

parameter ST\_WENABLE=3'b110;

parameter ST\_WENABLEP=3'b111;

//////////////////////////////////////////////////// PRESENT STATE LOGIC

reg [2:0] PRESENT\_STATE,NEXT\_STATE;

always @(posedge Hclk)

begin:PRESENT\_STATE\_LOGIC

if (~Hresetn)

PRESENT\_STATE<=ST\_IDLE;

else

PRESENT\_STATE<=NEXT\_STATE;

end

/////////////////////////////////////////////////////// NEXT STATE LOGIC

always @(PRESENT\_STATE,valid,Hwrite,Hwritereg)

begin:NEXT\_STATE\_LOGIC

case (PRESENT\_STATE)

ST\_IDLE:begin

if (~valid)

NEXT\_STATE=ST\_IDLE;

else if (valid && Hwrite)

NEXT\_STATE=ST\_WWAIT;

else

NEXT\_STATE=ST\_READ;

end

ST\_WWAIT:begin

if (~valid)

NEXT\_STATE=ST\_WRITE;

else

NEXT\_STATE=ST\_WRITEP;

end

ST\_READ: begin

NEXT\_STATE=ST\_RENABLE;

end

ST\_WRITE:begin

if (~valid)

NEXT\_STATE=ST\_WENABLE;

else

NEXT\_STATE=ST\_WENABLEP;

end

ST\_WRITEP:begin

NEXT\_STATE=ST\_WENABLEP;

end

ST\_RENABLE:begin

if (~valid)

NEXT\_STATE=ST\_IDLE;

else if (valid && Hwrite)

NEXT\_STATE=ST\_WWAIT;

else

NEXT\_STATE=ST\_READ;

end

ST\_WENABLE:begin

if (~valid)

NEXT\_STATE=ST\_IDLE;

else if (valid && Hwrite)

NEXT\_STATE=ST\_WWAIT;

else

NEXT\_STATE=ST\_READ;

end

ST\_WENABLEP:begin

if (~valid && Hwritereg)

NEXT\_STATE=ST\_WRITE;

else if (valid && Hwritereg)

NEXT\_STATE=ST\_WRITEP;

else

NEXT\_STATE=ST\_READ;

end

default: begin

NEXT\_STATE=ST\_IDLE;

end

endcase

end

////////////////////////////////////////////////////////OUTPUT LOGIC:COMBINATIONAL

reg Penable\_temp,Hreadyout\_temp,Pwrite\_temp;

reg [2:0] Pselx\_temp;

reg [31:0] Paddr\_temp, Pwdata\_temp;

always @(\*)

begin:OUTPUT\_COMBINATIONAL\_LOGIC

case(PRESENT\_STATE)

ST\_IDLE: begin

if (valid && ~Hwrite)

begin:IDLE\_TO\_READ

Paddr\_temp=Haddr;

Pwrite\_temp=Hwrite;

Pselx\_temp=tempselx;

Penable\_temp=0;

Hreadyout\_temp=0;

end

else if (valid && Hwrite)

begin:IDLE\_TO\_WWAIT

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=1;

end

else

begin:IDLE\_TO\_IDLE

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=1;

end

end

ST\_WWAIT:begin

if (~valid)

begin:WAIT\_TO\_WRITE

Paddr\_temp=Haddr1;

Pwrite\_temp=1;

Pselx\_temp=tempselx;

Penable\_temp=0;

Pwdata\_temp=Hwdata;

Hreadyout\_temp=0;

end

else

begin:WAIT\_TO\_WRITEP

Paddr\_temp=Haddr1;

Pwrite\_temp=1;

Pselx\_temp=tempselx;

Pwdata\_temp=Hwdata;

Penable\_temp=0;

Hreadyout\_temp=0;

end

end

ST\_READ: begin:READ\_TO\_RENABLE

Penable\_temp=1;

Hreadyout\_temp=1;

end

ST\_WRITE:begin

if (~valid)

begin:WRITE\_TO\_WENABLE

Penable\_temp=1;

Hreadyout\_temp=1;

end

else

begin:WRITE\_TO\_WENABLEP ///DOUBT

Penable\_temp=1;

Hreadyout\_temp=1;

end

end

ST\_WRITEP:begin:WRITEP\_TO\_WENABLEP

Penable\_temp=1;

Hreadyout\_temp=1;

end

ST\_RENABLE:begin

if (valid && ~Hwrite)

begin:RENABLE\_TO\_READ

Paddr\_temp=Haddr;

Pwrite\_temp=Hwrite;

Pselx\_temp=tempselx;

Penable\_temp=0;

Hreadyout\_temp=0;

end

else if (valid && Hwrite)

begin:RENABLE\_TO\_WWAIT

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=1;

end

else

begin:RENABLE\_TO\_IDLE

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=1;

end

end

ST\_WENABLEP:begin

if (~valid && Hwritereg)

begin:WENABLEP\_TO\_WRITEP

Paddr\_temp=Haddr2;

Pwrite\_temp=Hwrite;

Pselx\_temp=tempselx;

Penable\_temp=0;

Pwdata\_temp=Hwdata;

Hreadyout\_temp=0;

end

else

begin:WENABLEP\_TO\_WRITE\_OR\_READ /////DOUBT

Paddr\_temp=Haddr2;

Pwrite\_temp=Hwrite;

Pselx\_temp=tempselx;

Pwdata\_temp=Hwdata;

Penable\_temp=0;

Hreadyout\_temp=0;

end

end

ST\_WENABLE :begin

if (~valid && Hwritereg)

begin:WENABLE\_TO\_IDLE

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=0;

end

else

begin:WENABLE\_TO\_WAIT\_OR\_READ /////DOUBT

Pselx\_temp=0;

Penable\_temp=0;

Hreadyout\_temp=0;

end

end

endcase

end

////////////////////////////////////////////////////////OUTPUT LOGIC:SEQUENTIAL

always @(posedge Hclk)

begin

if (~Hresetn)

begin

Paddr<=0;

Pwrite<=0;

Pselx<=0;

Pwdata<=0;

Penable<=0;

Hreadyout<=0;

end

else

begin

Paddr<=Paddr\_temp;

Pwrite<=Pwrite\_temp;

Pselx<=Pselx\_temp;

Pwdata<=Pwdata\_temp;

Penable<=Penable\_temp;

Hreadyout<=Hreadyout\_temp;

end

end

endmodule

// AHB to APG Bridge | Maven Silicon

//

//

//

// AHB Master

// Date:17-06-2022

//

// By-Prajwal Kumar Sahu

module APB\_Interface(Pwrite,Pselx,Penable,Paddr,Pwdata,Pwriteout,Pselxout,Penableout,Paddrout,Pwdataout,Prdata);

input Pwrite,Penable;

input [2:0] Pselx;

input [31:0] Pwdata,Paddr;

output Pwriteout,Penableout;

output [2:0] Pselxout;

output [31:0] Pwdataout,Paddrout;

output reg [31:0] Prdata;

assign Penableout=Penable;

assign Pselxout=Pselx;

assign Pwriteout=Pwrite;

assign Paddrout=Paddr;

assign Pwdataout=Pwdata;

always @(\*)

begin

if (~Pwrite && Penable)

Prdata=($random)%256;

else

Prdata=0;

end

endmodule

// AHB to APG Bridge | Maven Silicon

//

//

//

// Bridge Top

// Date:14-06-2022

//

// By-Prajwal Kumar Sahu

module Bridge\_Top(Hclk,Hresetn,Hwrite,Hreadyin,Hreadyout,Hwdata,Haddr,Htrans,Prdata,Penable,Pwrite,Pselx,Paddr,Pwdata,Hreadyout);

input Hclk,Hresetn,Hwrite,Hreadyin;

input [31:0] Hwdata,Haddr,Prdata;

input[1:0] Htrans;

output Penable,Pwrite,Hreadyout;

output [2:0] Pselx;

output [31:0] Paddr,Pwdata;

///////////////////////////////////////////////////////////////INTERMEDIATE SIGNALS

wire valid;

wire [31:0] Haddr1,Haddr2,Hwdata1,Hwdata2;

wire Hwritereg;

wire [2:0] tempselx;

/////////////////////////////////////////////////////////////// MODULE INSTANTIATIONS

AHB\_slave\_interface AHBSlave (Hclk,Hresetn,Hwrite,Hreadyin,Htrans,Haddr,Hwdata,Prdata,valid,Haddr1,Haddr2,Hwdata1,Hwdata2,Hwritereg,tempselx);

APB\_FSM\_Controller APBControl ( Hclk,Hresetn,valid,Haddr1,Haddr2,Hwdata1,Hwdata2,Prdata,Hwrite,Haddr,Hwdata,Hwritereg,tempselx,Pwrite,Penable,Pselx,Paddr,Pwdata,Hreadyout);

Endmodule

module tb();

reg Hclk,Hresetn;

wire [1:0]Hresp =0 ;

wire [31:0] Hwdata,Haddr,Prdata,Paddr,Pwdata,Paddrout,Pwdataout;

wire [31:0] Hrdata=Prdata;

wire Hwrite,Hreadyin;

wire [1:0] Htrans;

wire Penable,Pwrite,Hreadyout,Pwriteout,Penableout;

wire [2:0] Pselx,Pselxout;

AHB\_Master ahb\_master(Hclk,Hresetn,Hresp,Hrdata,Hwrite,Hreadyin,Hreadyout,Htrans,Hwdata,Haddr);

Bridge\_Top bridge\_top(Hclk,Hresetn,Hwrite,Hreadyin,Hreadyout,Hwdata,Haddr,Htrans,Prdata,Penable,Pwrite,Pselx,Paddr,Pwdata);

APB\_Interface apb(Pwrite,Pselx,Penable,Paddr,Pwdata,Pwriteout,Pselxout,Penableout,Paddrout,Pwdataout,Prdata);

initial

begin

Hclk=1'b0;

forever #10 Hclk=~Hclk;

end

task reset();

begin

@(negedge Hclk);

Hresetn = 0;

@(negedge Hclk);

Hresetn = 1;

end

endtask

initial

begin

reset;

ahb\_master.single\_write();

//ahb.single\_read();

//ahb.burst\_inc\_write();

//ahb.burst\_wrap\_write();

//ahb.burst\_inc\_read();

//ahb.burst\_wrap\_read();

#1000 $finish;

end

endmodule

// AHB to APG Bridge | Maven Silicon

//

//

//

// AHB Master

// Date:14-06-2022

//

// By-Prajwal Kumar Sahu

module AHB\_Master(Hclk,Hresetn,Hresp,Hrdata,Hwrite,Hreadyin,Hreadyout,Htrans,Hwdata,Haddr);

input Hclk,Hresetn,Hreadyout;

input [1:0]Hresp;

input [31:0] Hrdata;

output reg Hwrite,Hreadyin;

output reg [1:0] Htrans;

output reg [31:0] Hwdata,Haddr;

reg [2:0] Hburst;

reg [2:0] Hsize;

task single\_write();

begin

@(posedge Hclk)

#2;

begin

Hwrite=1;

Htrans=2'b10;

Hsize=3'b000;

Hburst=3'b000;

Hreadyin=1;

Haddr=32'h8000\_0001;

end

@(posedge Hclk)

#2;

begin

Htrans=2'b00;

Hwdata=8'hA3;

end

end

endtask

task single\_read();

begin

@(posedge Hclk)

#2;

begin

Hwrite=0;

Htrans=2'b10;

Hsize=3'b000;

Hburst=3'b000;

Hreadyin=1;

Haddr=32'h8000\_00A2;

end

@(posedge Hclk)

#2;

begin

Htrans=2'b00;

end

end

endtask

endmodule