PPI Report

Submitted By : G7

Anss Khaled Farouk

Akram Ahmed Mohamed

Abdelrhman Tarek Wahba

Submitted To:

Eng. Ahmed Hesham El Moselhy

cairo,
2018.

Control Word Register Module:

```
module control_word_reg (in , out, en, RESET);
    input [7:0]in;
    input RESET;
    output [3:0]out;
    input wire en;
    assign out[0] = (RESET==0)? 1'b0 : (in[7]==1 && en==1) ?
      ~in[4] :(in[7]==0 && en==1) ? 1'b0  : 1'b0;
    assign out[1] = (RESET==0)? 1'b0 : (in[7]==1 && en==1) ?
      ~in[1] :(in[7]==0 && en==1) ? 1'b0  : 1'b0;
    assign out[2] = (RESET==0)? 1'b0 : (in[7]==1 && en==1) ?
      ~in[0] :(in[7]==0 && en==1) ? 1'b1  : 1'b0;
    assign out[3] = in[7];
endmodule
```

Module Description

This module is implementation for Controlling Port A,B,C Input/Output modes.

`[7:0] in` is implementation for control word from Port D when `a0 = 1` or `a1 = 1`

`[2:0] out` is implementation for Input/Output modes for Port A,B,C and `[3] out` is implementation for `IC_mode` BSR or `Mode_0`

`en` is implementation for Enable Module when `a0 = 1` or `a1 = 1`

`RESET` is implementation for Change Ports to input mode

Control Unit Module:

```
module control_unit (CS,RD,WR,A0,A1,control, ic_mode);

    input CS,RD,WR,A0,A1,ic_mode;    output [3:0]control;

    reg [3:0]control_reg;
    assign control = control_reg;

    always @(*) begin

        if(~CS)begin
            if(~A0 && ~A1) begin // port A
                if(~RD || ~WR) begin
                    control_reg[0] <= 1'b1;
                    control_reg[3] <= 1'b1;

                    control_reg[1] <= 1'b0;
                    control_reg[2] <= 1'b0;
                end // if(RD || WR)end
            end
            if(A0 && ~A1) begin // port B
                if(~RD || ~WR) begin
                    control_reg[1] <= 1'b1;
                    control_reg[3] <= 1'b1;

                    control_reg[0] <= 1'b0;
                    control_reg[2] <= 1'b0;
                end // if(RD || WR)end
            end
            if(~A0 && A1) begin // port C
                if(~RD || ~WR) begin
                    control_reg[2] <= 1'b1;
                    control_reg[3] <= 1'b1;

                    control_reg[0] <= 1'b0;
                    control_reg[1] <= 1'b0;
                end // if(RD || WR)end
```

```
            end
            if(A0 && A1) begin // port D to ctrl-word-reg
                if(~RD || ~WR) begin
                    if(ic_mode == 0) begin
                        control_reg[2] <= 1'b1;
                        control_reg[3] <= 1'b1;


                    end else begin
                        control_reg[3] <= 1'b1;

                        control_reg[0] <= 1'b0;
                        control_reg[1] <= 1'b0;
                        control_reg[2] <= 1'b0;
                    end
                end // if(RD || WR)end
            end
        end
        else begin
            control_reg = 4'b0000;
        end
    end // always @(CS,RD,WR,A0,A1,RESET)
endmodule
```

Module Description:

Controlling Port A,B,C,D Enables in `Mode_0` and BSR

`output [3:0] control` is implementation for Enables wires for ports

Tri Module:

```
module Tri_State_Buffer (in, out, enable);
    input [7:0]in ;
    input enable;
    output [7:0]out;
    assign out = enable? in : 8'bzzzzzzzz;
endmodule
```

8 Bit Port Module:

```
module port8 (port , databus, mode , enable);
    inout [7:0]port;
    inout [7:0]databus;
     input mode , enable;
    wire en1 , en2;
    assign en1 = enable & (~mode);
    assign en2 = enable & (mode);
    Tri_State_Buffer t2(port , databus , en1);
    Tri_State_Buffer t1(databus , port , en2);
endmodule
```

Module Description:

Port A,B,D Declarations

8 Bit PortC Module:

```
module portC8 (port , databus, mode , enable, ic_mode ,A);
        inout [7:0]port;
        inout [7:0]databus;
        input mode , enable, ic_mode ,A;
         wire en1 , en2;

    assign en1 = enable & (~mode) & ic_mode;
    assign en2 = enable & (mode) & ic_mode;

    Tri_State_Buffer t2(port , databus , en1);
    Tri_State_Buffer t1(databus , port , en2);

    /* ** ** ** BSR MODE ** ** ** */
    assign port[0] = (databus[3:1] == 3'b000 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[0]  : 1'bz;
    assign port[1] = (databus[3:1] == 3'b001 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[1]  : 1'bz;
    assign port[2] = (databus[3:1] == 3'b010 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[2]  : 1'bz;
    assign port[3] = (databus[3:1] == 3'b011 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[3]  : 1'bz;
    assign port[4] = (databus[3:1] == 3'b100 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[4]  : 1'bz;
    assign port[5] = (databus[3:1] == 3'b101 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[5]  : 1'bz;
    assign port[6] = (databus[3:1] == 3'b110 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[6]  : 1'bz;
    assign port[7] = (databus[3:1] == 3'b111 && ic_mode == 0 &&
A==1)? databus[0] : (ic_mode == 0) ? port[7]  : 1'bz;
endmodule
```

Module Description:

Port C Declarations and BSR mode Implementation

The Top Module:

```
module Top_module (PA,PB,PC,PD,CS,A,RST,RD,WR);

    inout [7:0] PA; inout [7:0] PB; inout [7:0] PC; inout [7:0] PD; input wire CS,RST,RD,WR;    input wire [1:0] A; /* IC interface */


    wire [7:0] databus; /* internal bus between Port A,B,C,D*/

    assign PD_mode = (~RD) ? 1'b1 : (~WR) ? 1'b0 : 1'b0; /* Port D enable as read or write*/
    wire [3:0] port_enable; /* each bit connected to enable pin of each port A,B,C */

    control_unit ctrl(CS,RD,WR,A[0],A[1],port_enable, PD[7]);

    wire ctrl_word_reg_enable; /* Enable control word reg when A0 = 1 | A1 = 1 */
    assign ctrl_word_reg_enable = (A[0] & A[1]);

    wire [3:0] ctrl_word_reg_out; /* carry control word for port mode select */
    reg [7:0] control_word; /* saving control word when A0 = 1 | A1 = 1 */
    control_word_reg ctrl_word_reg (control_word  , ctrl_word_reg_out, 1'b1, RST);


    port8  PortA(PA,databus          ,ctrl_word_reg_out[0] ,port_enable[0]);
    port8  PortB(PB,databus          ,ctrl_word_reg_out[1] ,port_enable[1]);
    portC8 portC(PC,databus          ,ctrl_word_reg_out[2] ,port_enable[2] ,ctrl_word_reg_out[3],  ctrl_word_reg_enable);
    port8  portD(PD,databus          ,PD_mode ,port_enable[3]);
```

```
    always @(*) begin
        if(A[0] & A[1]) begin
            control_word <= PD;
        end else begin
            control_word <= control_word;
        end
    end

endmodule
```

Module Description:

implementation for the whole 2285 ic

The Test Bench Of Top Module:

```
module top_module_tb();

    reg [7:0] DeviceA;  reg [7:0] DeviceB;  reg [7:0] DeviceC;
reg [7:0] DeviceD;
    wire [7:0] PA;  wire [7:0] PB;  wire [7:0] PC;  wire [7:0]
PD;
    reg cs,rd,wr,rst;   reg [1:0]a;

    Top_module tpmd (PA,PB,PC,PD,cs,a,rst,rd,wr);

    assign PA = (~rd && wr) ? DeviceA : 8'bzzzzzzzz;
    assign PB = (~rd && wr) ? DeviceB : 8'bzzzzzzzz;
    assign PC = (~rd && wr) ? DeviceC : 8'bzzzzzzzz;
    assign PD = (~wr && rd) ? DeviceD : 8'bzzzzzzzz;

    initial begin

        cs = 1;     rd = 1;     wr = 1;     rst = 1;    a[0] =
1'b0;    a[1] = 1'b0;

        $monitor("rd %b wr %b ||| a0 %b a1 %b |||PD %b PA %b PB
%b PC %b",rd,wr,a[0],a[1],PD,PA,PB,PC);


$display("********************WRITE*************************");
        #5
        cs <= 0;
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b1000_0000;
```

```
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 0;
        a[1] <= 0;
        DeviceD <= 8'b0101_0101;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 1;
        a[1] <= 0;
        DeviceD <= 8'b1001_1001;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 0;
        a[1] <= 1;
        DeviceD <= 8'b1111_1001;
        wr <= 0;
        rd <= 1;

        #5
$display("********************READ*************************");

        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b1001_1011;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 0;
        a[1] <= 0;
        DeviceA <= 8'b1000_1000;
        DeviceB <= 8'b1001_1001;
```

```
        DeviceC <= 8'b1011_1101;
        rd <= 0;
        wr <= 1;

        #5
        a[0] <= 1;
        a[1] <= 0;
        DeviceA <= 8'b1000_1000;
        DeviceB <= 8'b1001_1001;
        DeviceC <= 8'b1011_1101;
        rd <= 0;
        wr <= 1;

        #5
        a[0] <= 0;
        a[1] <= 1;
        DeviceA <= 8'b1000_1000;
        DeviceB <= 8'b1001_1001;
        DeviceC <= 8'b1011_1101;
        rd <= 0;
        wr <= 1;


        #5
$display("**********************BSR************************");

        #5
        cs <= 0;
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0000_0001;
        wr <= 0;
        rd <= 1;
        #5
        cs <= 0;
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0000_1111;
        wr <= 0;
```

```
        rd <= 1;
        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0000_1011;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0xxx_1110;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0xxx_0011;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0xxx_0001;
        wr <= 0;
        rd <= 1;

        #5
        a[0] <= 1;
        a[1] <= 1;
        DeviceD <= 8'b0xxx_1110;
        wr <= 0;
        rd <= 1;


    end

endmodule
```

the output

```
abdotarek@abdotarek:~/Documents/git/PPI$ iverilog PPI-intel-8255.v
abdotarek@abdotarek:~/Documents/git/PPI$ vvp a.out
********************WRITE************************
rd 1 wr 1 ||| a0 0 a1 0 |||PD zzzzzzzz PA zzzzzzzz PB zzzzzzzz PC xxxxxxxx
rd 1 wr 0 ||| a0 1 a1 1 |||PD 10000000 PA zzzzzzzz PB zzzzzzzz PC zzzzzzzz
rd 1 wr 0 ||| a0 0 a1 0 |||PD 01010101 PA 01010101 PB zzzzzzzz PC zzzzzzzz
rd 1 wr 0 ||| a0 1 a1 0 |||PD 10011001 PA zzzzzzzz PB 10011001 PC zzzzzzzz
rd 1 wr 0 ||| a0 0 a1 1 |||PD 11111001 PA zzzzzzzz PB zzzzzzzz PC 11111001
********************READ*************************
rd 1 wr 0 ||| a0 1 a1 1 |||PD 10011011 PA zzzzzzzz PB zzzzzzzz PC zzzzzzzz
rd 0 wr 1 ||| a0 0 a1 0 |||PD 10001000 PA 10001000 PB 10011001 PC 10111101
rd 0 wr 1 ||| a0 1 a1 0 |||PD 10011001 PA 10001000 PB 10011001 PC 10111101
rd 0 wr 1 ||| a0 0 a1 1 |||PD 10111101 PA 10001000 PB 10011001 PC 10111101
**********************BSR************************
rd 1 wr 0 ||| a0 1 a1 1 |||PD 00000001 PA zzzzzzzz PB zzzzzzzz PC zzzzzzz1
rd 1 wr 0 ||| a0 1 a1 1 |||PD 00001111 PA zzzzzzzz PB zzzzzzzz PC 1zzzzzz1
rd 1 wr 0 ||| a0 1 a1 1 |||PD 00001011 PA zzzzzzzz PB zzzzzzzz PC 1z1zzzz1
rd 1 wr 0 ||| a0 1 a1 1 |||PD 0xxx1110 PA zzzzzzzz PB zzzzzzzz PC 0z0zzzz1
rd 1 wr 0 ||| a0 1 a1 1 |||PD 0xxx0011 PA zzzzzzzz PB zzzzzzzz PC 1z0zzz11
rd 1 wr 0 ||| a0 1 a1 1 |||PD 0xxx0001 PA zzzzzzzz PB zzzzzzzz PC 1z0zzz11
rd 1 wr 0 ||| a0 1 a1 1 |||PD 0xxx1110 PA zzzzzzzz PB zzzzzzzz PC 0z0zzz10
```

synthiziable code