```
`timescale 1ps/1ps

module ZOctalRAMOperator(
        input iClk,
        input iRst_N,
        //iOp_Code=0: Idle.
        //iOp_Code=1: Reset IC.
        //iOp_Code=2: Write Mode Register.
        //iOp_Code=3: Read Mode Register and write to EBR 4K.
        //iOp_Code=4: Sync Write, iAddress[31:0], iData[15:0].
        //iOp_Code=5: Sync Read, iAddress[31:0], oData[15:0].
        input [2:0] iOp_Code,
        output reg oOp_Done,

        //iOp_Code=4: Burst Write, Address[31:0], Data[15:0].
        input [31:0] iAddress,
        input [15:0] iData,
        output reg [15:0] oData,

        //Octal RAM/EEPROM Interface.
    output reg oPSRAM_RST, //RESET# : Input Reset signal, active low.
    output reg oPSRAM_CE, //CE#: Input, Chip select, active low. When CE#=1, chip is in standby state.
        output oPSRAM_CLK,
        //DQS, IO.
        //DQ Strobe clock for DQ[7:0] during all reads.
        //Data mask for DQ[7:0] during memory writes.
        //DM is active HIGH, DM=1 means "do not write".
    inout ioPSRAM_DQS_DM,
    inout [7:0] ioPSRAM_DATA, //Address/Data bus [7:0].

        //Embedded Block RAM Interface.
        output reg oEBR_Wr_En,
        output reg [15:0] oEBR_Wr_Data,
        output reg [8:0] oEBR_Wr_Addr
);

//OctalRAM Clock.
//assign oPSRAM_CLK=0; //iClk;
//ERROR <62001109> -
//par: The connection from ic_pll.lscc_pll_inst.u_PLL_B/OUTGLOBAL (R13C32_JOUTGLOBAL_PLL) to oPSRAM_CLK/PADDO (R8C0_JPADDOA_PIO) in signal
clk_System is not routable.
//assign oPSRAM_CLK=iClk;
OB OB_PSRAM_CLK(
  .I (iClk),  // I
  .O (oPSRAM_CLK)   // O
);
```

```
//DQS Tri-State control.
//Data Mask for Wring(1: Not Write), Data Strobe for Reading(1: Data Valid).
reg oe_DQS_DM;
reg DQS_DM_Out;
wire DQS_DM_In;
// assign ioPSRAM_DQS_DM=(oe_DQS_DM)?(DQS_DM_Out):(1'bz);
// assign DQS_DM_In=ioPSRAM_DQS_DM;
BB_B bb_b_DQS (
  .T_N     (oe_DQS_DM),  // I,  from oe/tristate output to pad
  .I       (DQS_DM_Out),  // I,  from output register to pad
  .O       (DQS_DM_In),  // O,  from pad to input register input
  .B       (ioPSRAM_DQS_DM)   // IO, bidirectional pad
);

//Command List.
parameter CMD_SYNC_RD=8'h00;
parameter CMD_SYNC_WR=8'h80;
parameter CMD_LINEAR_BURST_RD=8'h20;
parameter CMD_LINEAR_BURST_WR=8'hA0;
parameter CMD_MODE_REG_RD=8'h40;
parameter CMD_MODE_REG_WR=8'hC0;
parameter CMD_GBL_RST=8'hFF;

//bidirectional IO.
wire [7:0] iPSRAM_DATA;
wire [7:0] oPSRAM_DATA;

//tristate Data Bus IO.
reg fab2oe; //Tri-state Control, Active Low. from Fabric to OE/Tri-State Control.
wire [7:0] oe2pad; //tri-state output enable to pad, driven by IOL_B primitive.

//Why can't I use oe2pad signal?
//Once I use this signal, I got following error.
//Error 67201108        Place & Route   ERROR <67201108>
//- Error in finding PIO comp for comp "ic_OctalRAM.DDR_DataBus_IOL_B[7]"!
//Please verify the correctness of your PIO to IOL connectivity by checking the device architecture constraints.
// assign ioPSRAM_DATA=(oe2pad)?(oPSRAM_DATA):(8'hzz);
// assign iPSRAM_DATA=ioPSRAM_DATA;
//What the fuck! I use BB_B primitives replace RTL implementation of tri-state, it works!
BB_B bb_b_Pad[7:0] (
  .T_N     (oe2pad),  // I,  from oe/tristate output to pad
  .I       (oPSRAM_DATA),  // I,  from output register to pad
  .O       (iPSRAM_DATA),  // O,  from pad to input register input
  .B       (ioPSRAM_DATA)   // IO, bidirectional pad
);

//PADDI(Pad In) => Split into => DI0(Rising Edge) & DI1(Falling Edge)
```

```
wire [7:0] DDR_Data_In_Rising;
wire [7:0] DDR_Data_In_Falling;
//DO0(Rising Edge) & DO1(Falling Edge) => Packed into => PADDO(Pad Out)
reg [7:0] DDR_Data_Out_Rising;
reg [7:0] DDR_Data_Out_Falling;

//8-bits bidirectional DDR input&output.
IOL_B #(.LATCHIN ("NONE_DDR"), .DDROUT ("YES"))
DDR_DataBus_IOL_B[7:0] (
  .PADDI  (iPSRAM_DATA),  // I, DDR data input from pad.
  .DO1    (DDR_Data_Out_Falling),  // I, output data to pad at falling edge.
  .DO0    (DDR_Data_Out_Rising),  // I, output data to pad at rising edge.
  //I enable clock always, so it encodes continously, its output changes until input changes.
  .CE     (1'b1),  // I, clock enabled.
  .IOLTO  (fab2oe),  // I, from Fabric to OE/Tri-State Control, Active Low.
  .HOLD   (1'b0),  // I
  .INCLK  (iClk),  // I, clock for input DDR.
  .OUTCLK (iClk),  // I, clock for output DDR.
  .PADDO  (oPSRAM_DATA),  // O, DDR data output to pad.
  .PADDT  (oe2pad),  // O, tri-state control to pad.
  .DI1    (DDR_Data_In_Falling),  // O, input data from pad at falling edge.
  .DI0    (DDR_Data_In_Rising)   // O, input data from pad at rising edge.
);

//Octal RAM Configuration before using.
reg [7:0] cfg_No;
wire [7:0] cfg_RegAddr;
wire [7:0] cfg_RegData;
ZOctalRAMCfg ic_cfg(
    .iNo(cfg_No),
    .oRegAddr(cfg_RegAddr),
    .oRegData(cfg_RegData)
);

//driven by counter.
reg [15:0] CNT1;
reg [15:0] CNT2;
reg [15:0] Temp_DR; //Temporary Data Register.
always @(posedge iClk or negedge iRst_N)
if(!iRst_N) begin
                                CNT1<=0; CNT2<=0;
                                oOp_Done<=0;
                                oPSRAM_CE<=1; oPSRAM_RST<=1;
                                DDR_Data_Out_Rising<=0; DDR_Data_Out_Falling<=0;
                                cfg_No<=0;
                                //From fabric to oe/tri-state control, active low.
                                fab2oe<=0;
```

```
                                //DQS Tri-State control.
                                //Data Mask for Wring(1: Not Write), Data Strobe for Reading(1: Data Valid).
                                oe_DQS_DM<=1; DQS_DM_Out<=0;
                                //EBR 4K Interface.
                                oEBR_Wr_En<=0;
                                oEBR_Wr_Data<=0;
                                oEBR_Wr_Addr<=0;
                        end
else begin
                case (iOp_Code)
                        1: //iOp_Code=1: Reset IC, RESET# Timing.
                                case(CNT1)
                                        0: //At default, CE=1, RST=1.
                                                begin oPSRAM_CE<=1; oPSRAM_RST<=1; CNT1<=CNT1+1; end
                                        1: //Device Initialization, tPU>150uS.
                                          //Wait for OctalRAM to be stable after power on.
                                          //f=100MHz, t=1/100MHz(s)=1000/100MHz(ms)=1000_000/100MHz(us)=10uS
                                          //Here we wait 2 times of tPU, 300uS/10uS=30.
                                                if(CNT2==100) begin CNT2<=0; CNT1<=CNT1+1; end
                                                else begin CNT2<=CNT2+1; end
                                        2:  //pull down RST while CE=1, tRP>1uS,
                                                begin oPSRAM_RST<=0; CNT1<=CNT1+1; end
                                        3: //pull up RST, tRST>=2uS, Reset to CMD valid.
                                                begin oPSRAM_RST<=1; CNT1<=CNT1+1; end
                                        4: //generate done signal, single pulse.
                                                begin oOp_Done<=1; CNT1<=CNT1+1; end
                                        5: //generate done signal, single pulse.
                                                begin oOp_Done<=0; CNT1<=0; end
                                endcase
                        2: //iOp_Code=2: Write Mode Register.
                                case(CNT1)
                                        0: //Prepare rising edge data before 1 clock.
                                                begin
                                                        fab2oe<=1; //fabric to oe/tri-state control, tri-state enabled, output.
                                                        //DDR_Data_Out_Rising<=8'h01; //1st clock rising edge data.
                                                        DDR_Data_Out_Rising<=CMD_MODE_REG_WR; //1st clock rising edge data.
                                                        oPSRAM_CE<=0; //Pull down CE to start.
                                                        CNT1<=CNT1+1;
                                                end
                                        1: //1st Clock to issue INST. 8'h01(rising)+8'h02(falling)
                                                begin
                                                        //DDR_Data_Out_Falling<=8'h02; //1st clock falling edge data.
                                                        DDR_Data_Out_Falling<=CMD_MODE_REG_WR; //1st clock falling edge data.

                                                        //DDR_Data_Out_Rising<=8'h19;  //2nd clock rising edge data.
                                                        DDR_Data_Out_Rising<=8'h00;  //2nd clock rising edge data.
```

```
                                CNT1<=CNT1+1;
                        end
                2: //2nd Clock, ignore.
                        begin
                                //DDR_Data_Out_Falling<=8'h87; //2nd clock falling edge data.
                                DDR_Data_Out_Falling<=8'h00; //2nd clock falling edge data.

                                //DDR_Data_Out_Rising<=8'h09; //3rd clock rising edge data.
                                DDR_Data_Out_Rising<=0; //3rd clock rising edge data.
                                CNT1<=CNT1+1;
                        end
                3: //3rd Clock to issue MA#(ModeRegisterAddress) at falling edge.
                        begin
                                //DDR_Data_Out_Falling<=8'h01; //3rd clock falling edge data.
                                DDR_Data_Out_Falling<=cfg_RegAddr; //3rd clock falling edge data.

                                //DDR_Data_Out_Rising<=8'h55; //4th clock rising edge data.
                                DDR_Data_Out_Rising<=cfg_RegData; //4th clock rising edge data.
                                CNT1<=CNT1+1;
                        end
                4: //4th Clock to issue MR#(ModeRegisterData) at rising edge, Latency=1.
                        begin
                                //DDR_Data_Out_Falling<=8'haa; //4th clock falling edge data.
                                DDR_Data_Out_Falling<=0; //4th clock falling edge data.
                                fab2oe<=0; //fabric to oe/tri-state control, tri-state disabled, input.
                                CNT1<=CNT1+1;
                        end
                5: //pull up CE to end.
                        begin oPSRAM_CE<=1; CNT1<=CNT1+1; end

                6: //Loop to write all mode registers.
                        begin
                                if(cfg_No==3) begin CNT1<=CNT1+1; end
                                else begin cfg_No<=cfg_No+1; CNT1<=0; end
                        end
                7: //generate one single pulse done signal.
                        begin oOp_Done<=1; CNT1<=CNT1+1; end
                8: //generate one single pulse done signal.
                        begin
                                oOp_Done<=0; CNT1<=0;
                                cfg_No<=4; //pre-setting read mode register address for iOp_Code=3.
                                oEBR_Wr_Addr<=0; //pre-setting EMB-4K Write Address.
                        end
                default:
                        begin CNT1<=0; end
        endcase
3: //iOp_Code=3: Read Mode Register and write to EBR 4K.
```

```
case(CNT1)
        0: //Prepare rising edge data before 1 clock.
                begin
                        oPSRAM_CE<=0; //Pull down CE to start.
                        fab2oe<=1; //fabric to oe/tri-state control, tri-state enabled, output.
                        DDR_Data_Out_Rising<=CMD_MODE_REG_RD; //1st clock rising edge data.
                        CNT1<=CNT1+1;
                end
        1: //1st Clock to issue INST.
                begin
                        DDR_Data_Out_Falling<=CMD_MODE_REG_RD; //1st clock falling edge data.

                        DDR_Data_Out_Rising<=8'h00;  //2nd clock rising edge data.
                        CNT1<=CNT1+1;
                end
        2: //2nd Clock, ignore.
                begin
                        DDR_Data_Out_Falling<=8'h00; //2nd clock falling edge data.

                        DDR_Data_Out_Rising<=0; //3rd clock rising edge data.
                        CNT1<=CNT1+1;
                end
        3: //3rd Clock to issue MA#(ModeRegisterAddress) at falling edge, Latency=1.
                begin
                        DDR_Data_Out_Falling<=cfg_RegAddr; //3rd clock falling edge data.
                        fab2oe<=0; //fabric to oe/tri-state control, tri-state disabled, input.
                        CNT1<=CNT1+1;
                end
        4: //4th Clock, Latency=2.
                begin CNT1<=CNT1+1; end
        5: //5th Clock, Latency=3.
                begin CNT1<=CNT1+1; end
        6: //6th Clock, Latency=4.
                begin CNT1<=CNT1+1; end
        7: //7th Clock, Latency=5.
                begin CNT1<=CNT1+1; end
        8: //8th Clock, Latency=6.
                begin CNT1<=CNT1+1; end
        9: //8th Clock, Latency=7.
                begin CNT1<=CNT1+1; end
        10: //8th Clock, Latency=8.
                begin CNT1<=CNT1+1; end
        11: //8th Clock, Latency=9.
                begin CNT1<=CNT1+1; end
        12: //8th Clock, Latency=10.
                begin CNT1<=CNT1+1; end
        13:     //Only Falling Edge Data is valid, it's D0.
```

Type	Operation	VL (default)		FL
		No Refresh	Refresh	
Memory	Read	LC	Up to LCx2	LCx2
	Write	WLC		WLC
Register	Read	LC		LC
	Write	1		1

*Note: see Table 15 for WLC settings.

RLC=10

Latency Type:Fixed. Read Latency=10.

```
                        begin
                                oEBR_Wr_En<=1;
                                //oEBR_Wr_Addr<=0;
                                oEBR_Wr_Data<={8'h55, DDR_Data_In_Falling}; //Leading with 0x55 to recognize easily.
                                CNT1<=CNT1+1;
                        end
                14: //pull up CE to end.
                        begin oPSRAM_CE<=1; oEBR_Wr_En<=0; CNT1<=CNT1+1; end

                15: //Loop to read all mode registers.
                        if(cfg_No==9) begin cfg_No<=0; CNT1<=CNT1+1; end
                        else begin cfg_No<=cfg_No+1; oEBR_Wr_Addr<=oEBR_Wr_Addr+1; CNT1<=0; end

                16: //generate one single pulse done signal.
                        begin oOp_Done<=1; CNT1<=CNT1+1; end
                17: //generate one single pulse done signal.
                        begin oOp_Done<=0; CNT1<=0; end
                default:
                        begin CNT1<=0; end
        endcase
4: //iOp_Code=4: Sync Write, Address[31:0], Data[15:0].
        //Octal DDR PSRAM device is byte-addressable.
        //Memory accesses must start on even addresses (A[0]=’0).
        //Mode Register accesses can start on even or odd address.
        case(CNT1)
                0: //Prepare rising edge data before 1 clock.
                        begin
                                oPSRAM_CE<=0; //Pull down CE to start.
                                fab2oe<=1; //fabric to oe/tri-state control, tri-state enabled, output.
                                DDR_Data_Out_Rising<=CMD_LINEAR_BURST_WR; //1st clock rising edge data.

                                //DQS Tri-State control, DQS=0, doesn't mask any data.
                                //Data Mask for Wring(1: Not Write), Data Strobe for Reading(1: Data Valid).
                                oe_DQS_DM<=1; DQS_DM_Out<=0;
                                CNT1<=CNT1+1;
                        end
                1: //1st Clock to issue INST.
                        begin
                                DDR_Data_Out_Falling<=CMD_LINEAR_BURST_WR; //1st clock falling edge data.

                                DDR_Data_Out_Rising<=iAddress[31:24];  //2nd clock rising edge data.
                                CNT1<=CNT1+1;
                        end
                2: //2nd Clock, A[3] (Rising Edge) + A[2] (Falling Edge).
                        begin
                                DDR_Data_Out_Falling<=iAddress[23:16]; //2nd clock falling edge data.
```

```
                                DDR_Data_Out_Rising<=iAddress[15:8]; //3rd clock rising edge data.
                                CNT1<=CNT1+1;
                        end
                3: //3rd Clock, A[1] (Rising Edge) + A[0] (Falling Edge), Latency=1.
                        begin
                                DDR_Data_Out_Falling<=iAddress[7:0]; //3rd clock falling edge data.

                                DDR_Data_Out_Rising<=0; //4th clock rising edge data.
                                CNT1<=CNT1+1;
                        end
                4: //4th Clock, Latency=2.
                        begin DDR_Data_Out_Rising<=8'h19;DDR_Data_Out_Falling<=8'h87; CNT1<=CNT1+1; end
                5: //5th Clock, Latency=3.
                        begin DDR_Data_Out_Rising<=8'h19;DDR_Data_Out_Falling<=8'h87;CNT1<=CNT1+1; end
                6: //6th Clock, Latency=4.
                        begin
                                DDR_Data_Out_Rising<=8'h19/*iData[15:8]*/;
                                DDR_Data_Out_Falling<=8'h87/*iData[7:0]*/;CNT1<=CNT1+1; end
                7: //7th Clock, Latency=5.
                        begin
                                DDR_Data_Out_Rising<=8'h5A/*iData[15:8]*/;
                                DDR_Data_Out_Falling<=8'h49/*iData[7:0]*/;CNT1<=CNT1+1; end
                8: //8th Clock, Output Data.
                        begin
                                DDR_Data_Out_Rising<=8'h33/*iData[15:8]*/;
                                DDR_Data_Out_Falling<=8'hFF/*iData[7:0]*/;CNT1<=CNT1+1; end
                9: //pull up CE to end.
                        begin oPSRAM_CE<=1; CNT1<=CNT1+1; end
                10: //generate one single pulse done signal.
                        begin oOp_Done<=1; CNT1<=CNT1+1; end
                11: //generate one single pulse done signal.
                        begin oOp_Done<=0; CNT1<=0; end
                default:
                        begin CNT1<=0; end
        endcase
5: //iOp_Code=5: Sync Read, iAddress[31:0], oData[15:0].
//Octal DDR PSRAM device is byte-addressable.
//Memory accesses must start on even addresses (A[0]=’0).
//Mode Register accesses can start on even or odd address.
        case(CNT1)
                0: //Prepare rising edge data before 1 clock.
                        begin
                                oPSRAM_CE<=0; //Pull down CE to start.
                                fab2oe<=1; //fabric to oe/tri-state control, tri-state enabled, output.
                                DDR_Data_Out_Rising<=CMD_SYNC_RD; //1st clock rising edge data.

                                //DQS Tri-State control, High-Z.
```

```
                //Data Mask for Wring(1: Not Write), Data Strobe for Reading(1: Data Valid).
                oe_DQS_DM<=0;
                CNT1<=CNT1+1;
        end
1: //1st Clock to issue INST.
        begin
                DDR_Data_Out_Falling<=CMD_SYNC_RD; //1st clock falling edge data.

                DDR_Data_Out_Rising<=iAddress[31:24];  //2nd clock rising edge data.
                CNT1<=CNT1+1;
        end
2: //2nd Clock, A[3] (Rising Edge) + A[2] (Falling Edge).
        begin
                DDR_Data_Out_Falling<=iAddress[23:16]; //2nd clock falling edge data.

                DDR_Data_Out_Rising<=iAddress[15:8]; //3rd clock rising edge data.
                CNT1<=CNT1+1;
        end
3: //3rd Clock, A[1] (Rising Edge) + A[0] (Falling Edge), Latency=1.
        begin
                DDR_Data_Out_Falling<=iAddress[7:0]; //3rd clock falling edge data.

                fab2oe<=0; //fabric to oe/tri-state control, tri-state disabled, input.
                DDR_Data_Out_Rising<=0; //4th clock rising edge data.
                CNT1<=CNT1+1;
        end
4: //4th Clock, Latency=2.
        begin
                DDR_Data_Out_Falling<=0; //4th clock falling edge data.
                CNT1<=CNT1+1;
        end
5: //5th Clock, Latency=3.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=0;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
6: //6th Clock, Latency=4.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=1;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
7: //7th Clock, Latency=5.
begin
```

```
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=2;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
8: //8th Clock, Latency=6.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=3;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
9:  //9th Clock, Latency=7.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=4;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
10: //10th Clock, Latency=8.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=5;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
11: //11th Clock, Latency=9.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=6;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
12: //12th Clock, Latency=10.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=7;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
13: //13th Clock, Latency=11.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=8;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
```

```
14: //14th Clock, Latency=12.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=9;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
15: //14th Clock, Latency=13.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=10;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
16: //14th Clock, Latency=14.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=11;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
17: //14th Clock, Latency=15.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=12;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
18: //14th Clock, Latency=16.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=13;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
19: //14th Clock, Latency=17.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=14;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
        CNT1<=CNT1+1;
end
20: //14th Clock, Latency=18.
begin
        oEBR_Wr_En<=1; oEBR_Wr_Addr<=15;
        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
        //oEBR_Wr_Data<=16'h2222;
```

```
                                        CNT1<=CNT1+1;
                                end
                                21: //14th Clock, Latency=19.
                                begin
                                        oEBR_Wr_En<=1; oEBR_Wr_Addr<=16;
                                        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
                                        //oEBR_Wr_Data<=16'h2222;
                                        CNT1<=CNT1+1;
                                end
                                22: //14th Clock, Latency=20.
                                begin
                                        oEBR_Wr_En<=1; oEBR_Wr_Addr<=17;
                                        oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
                                        //oEBR_Wr_Data<=16'h2222;
                                        CNT1<=CNT1+1;
                                end
                                23: //Read Data In, Falling Edge data is valid, it's D0.
                                        begin
                                                oEBR_Wr_En<=1; oEBR_Wr_Addr<=18;
                                                oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
                                                //oEBR_Wr_Data<=16'h2222;
                                                CNT1<=CNT1+1;
                                        end
                                24: //Read Data In, Rising Edge data is valid, it's D1.
                                        begin
                                                oEBR_Wr_En<=1; oEBR_Wr_Addr<=19;
                                                oEBR_Wr_Data<={DDR_Data_In_Rising,DDR_Data_In_Falling};
                                                //oEBR_Wr_Data<=16'h3333;
                                                CNT1<=CNT1+1;
                                        end
                                25: //Pull CE up to end.
                                        begin oPSRAM_CE<=1; oEBR_Wr_En<=0; CNT1<=CNT1+1; end
                                26: //generate one single pulse done signal.
                                        begin oOp_Done<=1; CNT1<=CNT1+1; end
                                27: //generate one single pulse done signal.
                                        begin oOp_Done<=0; CNT1<=0; end
                        endcase
                default:
                        begin oOp_Done<=0; end
        endcase

    end

endmodule
```