Gowin I3C SDR IP

参考设计

RD509-1.1,2018-09-26

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2018/01/30	1.0	初始版本。
2018/09/26	1.1	修改 IP 名称。

目录

1 I3C SDR IP 参考设计概述

本文档所介绍的参考设计主要包括以下三种类型：

- I3C SDR 主机和 I3C SDR 从机之间的通信；
- I3C SDR 主机和 GW-I2C 从机之间的通信；
- I3C SDR 主机和 ASIC-I2C 从机之间的通信。

2 I3C SDR 主机和 I3C SDR 从机间的通信

2.1 目的

本参考设计主要用于验证 I3C SDR 主机和 I3C SDR 从机间通信是否正常。

2.2 功能

本参考设计案例名字为：w_ibi，其主要功能包括：

1. 主机进行写操作（发送从机地址和写操作位 8'h30，写数据给从机）；
2. 进入 IBI 状态（从机发送地址和读操作位，主机强读一个字节数据，从机发送读数据）。

2.3 MCU 及子模块

MCU 即 Master 工程中的 top.v。

简介

1. top.v 定义了很多端口，其中有些端口虽定义但暂时未用，这样做的目的是，若想观察某些信号，可将其拉到这些预定端口上；
2. top.v 主要用于完成时钟分频、消抖、写读数据。
   - counter 子模块：时钟分频。首先 50MHz 的时钟频率经过 PLL 分频成 10MHz，然后经过 counter 子模块分频成 10KHz；
   - deUstb 子模块：按键消抖；
   - 写读数据：I3C SDR 主机输出端口 STATE，会输出相应 I3C SDR 的状态。
     根据其状态，完成相应功能。如，STATE 为 S_CM_WAIT_AD，则 I3C SDR 主机处于等待地址状态，MCU 可发送地址；STATE 为 S_CM_WAIT_W_DA，则 I3C SDR 主机处于等待写入数据状态，MCU 可写数据。

代码

1. 时钟分频。
   经过 PLL 和 counter 模块处理后，时钟频率由 50MHz 分频为 10KHz。

```
GW_PLL myPll(
    .clkout(midClk), //output clkout 10M
    .clkin(clk_ext) //input clkin 50M
 );
counter cnt1(lowOut,lowClk,lowOver,midClk,1'b1);
defparam cnt1.OVER = 1_000;
```

2. 实例化 I3C SDR Master。

```
I3C_REG_top master(
    LGYS, CMS, ACS, AAS, STOS, STAS,
    LGYO, CMO, ACO, AAO, SIO, STOO, STAO,
     LGYC, CMC, ACC, AAC, SIC, STOC, STAC,
    STA_HD_S,
    SEND_AD_HIGH_S,
```

```
        SEND_AD_LOW_S,
        ACK_HIGH_S,
        ACK_LOW_S,
        STO_HIGH_S,
        STO_LOW_S,
        SEND_DA_HIGH_S,
        SEND_DA_LOW_S,
        RCVE_DA_HIGH_S,
        RCVE_DA_LOW_S,
        ADDRESS_S,
        PARITYERROR,
        DI, DOBUF, DO, STATE,
        SCLH, SCLL,
        j10_1, j10_5, j10_2, j10_6,
        CE, RST, CLK
       );
```

3. 按键消抖。

```
deUstb deKey1(key1,~key_1,CLK);
```

4. 写读数据等。

```
always @(posedge CLK)begin
    if(RST)begin
        dataTobeSend <=8'h10;
        DI <=0;
        SIC <= 0;
        .......
    end
    else begin
        if(SIO)begin
            case(STATE)
```

```
S_CM_ADRS_ABTR_OK:begin
    SIC <=1;
    ACS <=1;

    DI<=8'h00;
    STOS<=0;
    STAS<=0;
end
S_CM_WAIT_AD:begin
    DI<=8'h30;
    SIC<=1;

    STOS<=0;
    STAS<=0;
    ACS<=0;
end
S_CM_WAIT_NACK:begin
    STOS <=1;
    SIC<=1;

    DI<=8'h00;
    STAS<=0;
    ACS<=0;
end
S_CM_WAIT_W_DA:begin
    if(dataCnt_w==3)begin
        STOS<=1;
        ACS<=0;
        SIC<=1;
        dataCnt_w <= 0;
```

```
                end
                else begin
                    if(w_cnt_flag)begin
                        DI <= dataTobeSend;
                        dataTobeSend <= dataTobeSend+1;
                        dataCnt_w <= dataCnt_w+1;
                        w_cnt_flag<=0;

                        STOS<=0;
                        ACS<=0;
                    end
                end
                SIC<=1;


                STAS<=0;

            end
            S_CM_READ_OK_SI:begin
                if(!readOver) begin
                    if(read_cnt_flag)begin
                        dataCnt_r <= dataCnt_r+1;
                        read_cnt_flag<=0;
                        end
                    end
                    SIC<=1;

                    DI<=8'h00;
                    STOS<=0;
                    STAS<=0;
                    ACS<=0;
```

```
                end


            S_IDLE:begin
                pass <=1;
                SIC<=1;


                DI<=8'h00;
                STOS<=0;
                STAS<=0;
                ACS<=0;
            end

            default: begin
                DI<=8'h00;
                SIC<=0;
                STAS<=0;
                STOS<=0;
                ACS<=0;
            end
        endcase
    end
    else begin
        case(STATE)
            S_CM_W_STO:begin
                AAS<=1;
                ACS<=1;

                DI<=8'h00;
                SIC<=0;
```

```
            STAS<=0;
            STOS<=0;

        end
        S_CM_READ_SL_OK:begin
            if(dataCnt_r == 3 )begin
                if(!readOver)begin
                    STOS<=1;
                end
                readOver <=1;
                ACS <= 0;
            end
            else begin
                read_cnt_flag<=1;
                ACS<=1;
            end

            DI<=8'h00;
            SIC<=0;
            STAS<=0;

        end
        S_CM_ADRS_ABTR_OK:begin
            SIC <=1;
            ACS <=1;

            DI<=8'h00;

            STAS<=0;
            STOS<=0;
```

```
                    end
                    default:begin
                        DI<=8'h00;
                        SIC<=0;
                        STOS<=0;
                        STAS<=0;
                        ACS<=0;
                        w_cnt_flag<=1;
                    end
                endcase
            end
        end
        if(key3)begin
            STAS<=1;
        end
    end
```

5. 拉出观察的信号。

```
assign j9_4 = sda_line;//a1:0
assign j9_5 = scl_line;
assign j9_6 = STAS;
.....
```

2.4 板级连接

本参考设计采用双板方式完成 I3C SDR 主从机之间的通信，即一块 GW1N-9K 板上放 I3C SDR 主机，另外一块 GW1N-9K 板上放 I3C SDR 从机。因此板间连接需三根线：一根作为 SCL；一根作为 SDA；一根作为 GND。另外，其他相关观察信号物理约束可参考 dk_start_ge1n4.cst 文件。

2.5 结果观察

本参考设计采用示波器观察 SCL，SDA 是否正确。另外，若想观察 STATE，DI，DOBUF 等 Bus 型信号，建议采用逻辑分析仪。

3 I3C SDR 主机和 GW-I2C 从机间的通信

3.1 目的

本参考设计验证 I3C SDR 主机和 GW-I2C 从机是否正常通信。

3.2 功能

本参考设计案例名字为：write_read，其主要功能包括：

1. 主机进行写操作（发送从机地址和写操作位 8'hA0，发送寄存器地址和数据给从机）；
2. 主机进行读操作（发送从机地址和读操作位 8'hA0，发送寄存器地址，发送从机地址和读操作位 8'hA1，然后读取从机的数据）。

3.3 MCU 及子模块

简介

MCU 即 Master 工程中的 top.v。另外，本 top.v 及 I3C SDR 主机和 I2C 从机间的通信 Master 工程中的 top.v 都是在 I3C SDR 主机和 I3C SDR 从机间的通信 Master 工程中的 top.v 的基础上进行的修改。因此，参考设计间的 top.v 内容相似。

top.v 主要用于完成时钟分频、消抖、写读数据。

1. counter 子模块：时钟分频。首先 50MHz 的时钟频率经过 PLL 分频成 10MHz，然后经过 counter 子模块分频成 10KHz；
2. deUstb 子模块：按键消抖；
3. 写读数据：I3C SDR 主机输出端口 STATE，会输出相应 I3C SDR 的状态。根据其状态，完成相应功能。如，STATE 为 S_CM_WAIT_AD，则 I3C SDR 主机处于等待地址状态，MCU 可发送地址；STATE 为 S_CM_WAIT_W_DA，则 I3C SDR 主机处于等待写入数据状态，MCU 可写数据。

代码

1. 时钟分频。经过 PLL 和 counter 模块处理后，时钟频率由 50MHz 分频为 10KHz。

```
GW_PLL myPll(
    .clkout(midClk), //output clkout 10M
    .clkin(clk_ext) //input clkin 50M
 );
counter cnt1(lowOut,lowClk,lowOver,midClk,1'b1);
defparam cnt1.OVER = 1_000;
```

2. 实例化 I3C SDR Master。

```
I3C_REG_top master(
    LGYS, CMS, ACS, AAS, STOS, STAS,
    LGYO, CMO, ACO, AAO, SIO, STOO, STAO,
     LGYC, CMC, ACC, AAC, SIC, STOC, STAC,
    STA_HD_S,
    SEND_AD_HIGH_S,
    SEND_AD_LOW_S,
    ACK_HIGH_S,
    ACK_LOW_S,
    STO_HIGH_S,
    STO_LOW_S,
    SEND_DA_HIGH_S,
    SEND_DA_LOW_S,
    RCVE_DA_HIGH_S,
```

```
                RCVE_DA_LOW_S,
                ADDRESS_S,
                PARITYERROR,
                DI, DOBUF, DO, STATE,
                SCLH, SCLL,
                SDA,SCL,SDA_PULL,SCL_PULL,
                CE, RST, CLK
                `ifdef DEBUG_REG
                ,dbg_ABTR
                `endif
                );
```

3. 按键消抖。

```
                deUstb deKey1(key1,~key_1,CLK);
```

4. 写读数据等。

```
always @(posedge CLK)begin
    if(RST)begin
        dataTobeSend <=8'h20;
        DI <=0;
        SIC <= 0;
        STOS<=0;
        STAS<=0;
        pass<=0;
        w_cnt_flag<=0;
        r_cnt_flag<=0;
        read_flag <=0;
        readOver <=0;
        send_ad_flag<=0;
        led_4_flag<=0;
        led_5_flag<=0;
```

```
        led_6_flag<=0;

    end
    else begin
        if(SIO)begin
            case(STATE)
                S_CM_WAIT_AD:begin
                    if(send_ad_flag && !reg_addr)begin
                        DI<=8'hA1;
                    end
                    else begin
                        DI<=8'hA0;
                        reg_addr<=1;
                        if(read_flag)begin
                            send_ad_flag<=1;
                        end
                    end
                    SIC<=1;
                end

                S_CM_WAIT_W_DA:begin//9
                    if(reg_addr)begin
                        DI<=8'h10;
                        //reg_addr<=0;
                        led_5_flag<=1;
                        if(read_flag)begin
                            STAS <=1;
                        end
                    end

                    else begin
```

```
                    if(!read_flag)begin
                        if(&dataCnt_w)begin
                            STAS <= 1;
                            dataCnt_w <= 0;
                            read_flag <= 1;
                            led_4_flag <= 1;
                        end
                        else begin
                            if(w_cnt_flag && !reg_addr)begin
                                DI <= dataTobeSend;
                                dataTobeSend <= dataTobeSend+1;
                                dataCnt_w <= dataCnt_w+1;
                                w_cnt_flag<=0;
                            end
                        end
                    end
                end
                SIC<=1;
            end

            S_CI2CM_SEND_ACK_OK:begin
                if(dataCnt_r == 6)begin
                    STOS <=1;
                    dataCnt_r <= 0;
                    led_6_flag <= 1;

                end
                else begin
                    if(r_cnt_flag)begin
                        dataCnt_r <= dataCnt_r +1;
                        r_cnt_flag <=0;
```

```
                    end

                end
                AAS <= 0;
                SIC <= 1;
            end

            S_CM_SL_END:begin
                SIC <=1;
            end

            S_IDLE:begin
                pass <=1;
                SIC<=1;
            end

            default: begin
                DI<=8'h00;
                SIC<=0;
            end
        endcase
    end
    else begin
        case(STATE)
            S_CI2CM_SEND_ACK:begin//17-----state current I2C master read slave
                if(dataCnt_r == 5)begin
                    AAS<=0;
                    readOver <= 1;
                end
                else begin
                    if(readOver)begin
```

```
                    AAS <= 0;
                end
                else begin
                    AAS <= 1;
                end
            end
        end

        S_CI2CM_WAIT_ACK_HF:begin
            reg_addr <= 0;
        end

        default:begin
            DI<=8'h00;
            SIC<=0;
            STOS<=0;
            STAS<=0;
            AAS <=0;
            w_cnt_flag <=1;
            r_cnt_flag <=1;

        end
    endcase
    if(key4) begin
        STAS<=1;
        pass <=0;
        send_ad_flag <=0;
        led_4_flag <=0;
        led_5_flag <=0;
        led_6_flag <=0;
    end
```

```
        end
    end
end
```

5. 拉出观察的信号

```
assign j9_4 = sda_line;//a1:0
assign j9_5 = scl_line;
assign j9_6 = STAS;
.....
```

3.4 板级连接

本参考设计采用两板方式完成 I3C SDR 主机与 GW-I2C 从机间的通信，即第一块 GW1N-9K 板上放 I3C SDR 主机，第二块 GW1N-9K 板上放 GW-I2C 从机。因此第一块和第二块板间连接需三根线：一根作为 SCL；一根作为 SDA；一根作为 GND。

3.5 结果观察

本参考设计采用示波器观察 SCL，SDA 是否正确。另外，若想观察 STATE，DI，DOBUF 等 Bus 型信号，建议采用逻辑分析仪。

4 I3C SDR 主机和 ASIC-I2C 从机间的通信

4.1 目的

本参考设计验证 I3C SDR 主机和 ASIC-I2C 从机是否正常通信。

4.2 功能

本参考设计案例名字为：write，其主要功能为主机进行写操作（发送从机地址和写操作位 8'hA0，发送寄存器地址和数据给从机）。

4.3 MCU 及子模块

简介

MCU 即 Master 工程中的 top.v。top.v 主要用于完成时钟分频、消抖、写读数据。

1. counter 子模块：时钟分频。首先 50MHz 的时钟频率经过 PLL 分频成 10MHz，然后经过 counter 子模块分频成 10KHz；
2. deUstb 子模块：按键消抖；
3. 写读数据：I3C SDR 主机输出端口 STATE，会输出相应 I3C SDR 的状态。根据其状态，完成相应功能。如，STATE 为 S_CM_WAIT_AD，则 I3C SDR 主机处于等待地址状态， MCU 可发送地址；STATE 为 S_CM_WAIT_W_DA，则 I3C SDR 主机处于等待写入数据状态，MCU 可写数据。

代码

1. 时钟分频。经过 PLL 和 counter 模块处理后，时钟频率由 50MHz 分频为 10KHz。

```
GW_PLL myPll(
    .clkout(midClk), //output clkout 10M
```

```
        .clkin(clk_ext) //input clkin 50M
    );
    counter cnt1(lowOut,lowClk,lowOver,midClk,1'b1);
    defparam cnt1.OVER = 1_000;
```

2. 实例化 I3C SDR Master

```
    I3C_REG_top master(
        LGYS, CMS, ACS, AAS, STOS, STAS,
        LGYO, CMO, ACO, AAO, SIO, STOO, STAO,
        LGYC, CMC, ACC, AAC, SIC, STOC, STAC,
        STA_HD_S,
        SEND_AD_HIGH_S,
        SEND_AD_LOW_S,
        ACK_HIGH_S,
        ACK_LOW_S,
        STO_HIGH_S,
        STO_LOW_S,
        SEND_DA_HIGH_S,
        SEND_DA_LOW_S,
        RCVE_DA_HIGH_S,
        RCVE_DA_LOW_S,
        ADDRESS_S,
        PARITYERROR,
        DI, DOBUF, DO, STATE,
        SCLH, SCLL,
        SDA,SCL,SDA_PULL,SCL_PULL,
        CE, RST, CLK
);
```

3. 按键消抖

```
deUstb deKey1(key1,~key_1,CLK);
```

4. 写读数据等

```
always @(posedge CLK)begin
    if(RST)begin
        dataTobeSend <=8'h10;
        DI <=0;
        SIC <= 0;
        STOS<=0;
        STAS<=0;
        pass<=0;
        w_cnt_flag<=0;
        dataCnt_w <=0;
        led_flag4<=0;
        led_flag5<=0;
        led_flag6<=0;
    end
    else begin
        if(SIO)begin
            case(STATE)
                S_CM_WAIT_AD:begin
                    DI<=8'hA0;
                    SIC<=1;
                End

                S_CM_WAIT_W_DA:begin//9
                    if(&dataCnt_w)begin
                        led_flag4 <=1;
                        STOS<=1;
                        DI <= dataTobeSend;
                        STOS <=1;
                        dataCnt_w <= 0;
```

```
                    end
                    else begin
                        if(w_cnt_flag)begin
                            DI <= dataTobeSend;
                            dataTobeSend <= dataTobeSend+1;
                            dataCnt_w <= dataCnt_w+1;
                            w_cnt_flag<=0;
                        end
                    end
                    SIC<=1;
                end
                S_IDLE:begin
                    pass <=1;
                    SIC<=1;
                end
                default: begin
                    DI<=8'h00;
                    SIC<=0;
                    STAS<=0;
                end
            endcase
        end
        else begin
            case(STATE)
                S_CI2CM_WAIT_ACK_HF:begin
                    w_cnt_flag<=1;
                end
                default:begin
                    DI<=8'h00;
                    SIC<=0;
                    STOS<=0;
```

```
                    STAS<=0;
                end
             endcase
            if(key4)begin
                STAS<=1;
            end
        end
    end
end
```

5. 拉出观察的信号

```
    assign j9_4 = sda_line;//a1:0
assign j9_5 = scl_line;
assign j9_6 = STAS;
.....
```

4.4 板级连接

本参考设计采用两板方式完成 I3C SDR 主机与 ASIC-I2C 从机间的通信，即第一块 GW1N-9K 板上放 I3C 主机，第二块放 ASIC-I2C。因此第一块和第二块板间连接需三根线：一根作为 SCL；一根作为 SDA；一根作为 GND。

4.5 结果观察

本参考设计采用示波器观察 SCL，SDA 是否正确。另外，若想观察 STATE，DI，DOBUF 等 Bus 型信号，建议采用逻辑分析仪。


GOWIN高云
智慧逻辑 定制未来