基于 VerilogHDL 的 HDB3 编解码的实现

叶小敏
广州航新航空科技股份有限公司，广州 510663

摘　要： HDB3 码即三阶高密度双极性码，具有无直流成份，且只有很少的低频成份，连零个数不超过 3 个以及便于提取定时信息的特点。本文介绍了 HDB3 的基本编解码原理，并给出了如何使用 VerilogHDL 语言来实现 HDB3 的编解码，并给出了 ModelSim 功能仿真的波形。代码可以移植到 FPGA 中，对实际工程应用有一定的参考价值。

关键词： HDB3; 编解码; VerilogHDL; ModelSim 仿真

Implementation of HDB3 Codec Based on VerilogHDL

YE Xiaomin
(Guangzhou Hangxin Aviation Technology Co., Ltd., Guangzhou 510663, China)

【**Abstract**】HDB3 code that is third-order high-density bipolar code, with no DC component, and only a few low-frequency components, and the number of zero is not more than three, and clock information facilitate to extract. This paper introduces the basic principles of HDB3 codec, and shows how to use VerilogHDL code HDB3 codec implementation, and gives the ModelSim functional simulation waveforms. This code can be migrated to the FPGA, for practical engineering applications have a certain reference value.

【Key words】HDB3; Codec; VerilogHDL; ModelSim Simulation

0 引言

HDB3 作为基带通信系统中重要的码型之一，具有无直流成份，且只有很少的低频成份，连零个数不超过 3 个以及便于提取定时信息的特点，在实际中有着重要的应用。本文将首先给出 HDB3 的编解码原理，然后给出具体的 VerilogHDL 实现过程，同时还给出 ModelSim 的功能仿真波形。[1]

1 HDB3 编解码规则

本小节将简单介绍 HDB3 的编解码原理，为下文编解码的实现作铺垫。

1.1 HDB3 的编码原理

HDB3 的编码过程如下：

1) 首先将输入的原始数据码流进行 4 个连 0 的检测，对第 4 个 0 替换成 V 码。
2) 对 2 个 V 码间 1 的个数进行检测，如果个数为偶数，则将 4 个连 0 个的第一个 0 替换为 B 码。
3) 双极性转换，即原数据码流中的 1 与增加的 B 码一起进行正负极性交替变换；第一个 V 码与第一个数据 1 的极性一致，从第二个 V 码开始则正负极性交替变换。

例如，输入的原始码流为“1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1 1”，[2]分别经过加 V 码、加 B 码以及极性变换后得出的 HDB3 码，具体如下表所示。

表 1 HDB3 编码举例

原始码流	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*
加V 码	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*V*	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*0*	*V*	*1*	*1*	*1*
加B 码	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*V*	*1*	*0*	*1*	*1*	*0*	*1*	*B*	*0*	*0*	*V*	*1*	*1*	*1*
HDB3 码	*+1*	*0*	*0*	*-1*	*+1*	*0*	*0*	*0*	*+1*	*-1*	*0*	*+1*	*-1*	*0*	*+1*	*-1*	*0*	*0*	*-1*	*+1*	*-1*	*+1*

1.2 HDB3 的解码原理

HDB3 的解码过程如下：

1) 检测 HDB3 码流中出现连续 2 个极性相同的码，后一个则是 V 码，而 V 码前 2 位必然为 0，而 V 码前的第三位若不为 0，则必然是 B 码，找出 V 码和 B 码后，将其用 0 替换。

2) 去极性，即将码流中的+1 与-1 码全部转换为 1，0 码保持不变。

在此紧接着上面列举的 HDB3 编码的例子，进行解码举例，经过解码后可以还原出与原始码流一致的数据。

表 2 HDB3 解码举例

HDB3	+1	0	0	-1	+1	0	0	0	+1	-1	0	+1	-1	0	+1	-1	0	0	-1	+1	-1	+1
去 BV 码	+1	0	0	-1	+1	0	0	0	0	-1	0	+1	-1	0	+1	0	0	0	0	+1	-1	+1
去极性	1	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	1	1	1

2 HDB3 编解码实现的原理框图

根据上一小节的 HDB3 编解码原理，设计了如图 1 所示的编解码实现方案，总的来说由 HDB3 编码模块与 HDB3 解码模块构成，模块内部的具体实现与第 1 小节介绍的编解码过程一一对应，在此就不重复叙述。[2]

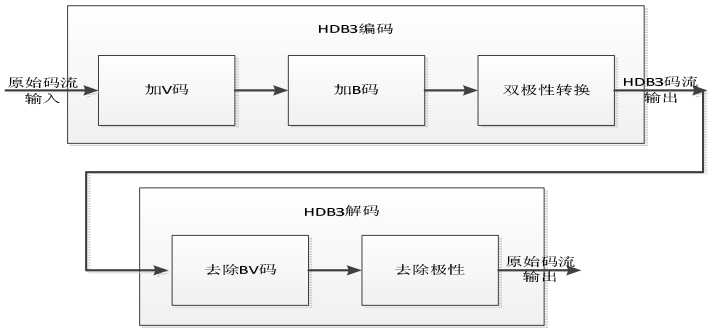

图 1 HDB3 的编解码原理框图

3 基于 VerilogHDL 的 HDB3 编码实现

本小节将介绍如何使用 VerilogHDL 来实现 HDB3 的编码。编码的框图如图 2 所示，三个模块都共用一个时钟与复位信号，对输入的原始数据流首先进行加 V 码操作，接着加 B 码，最后对其进行极性转换，最终输出的就是 HDB3 编码。

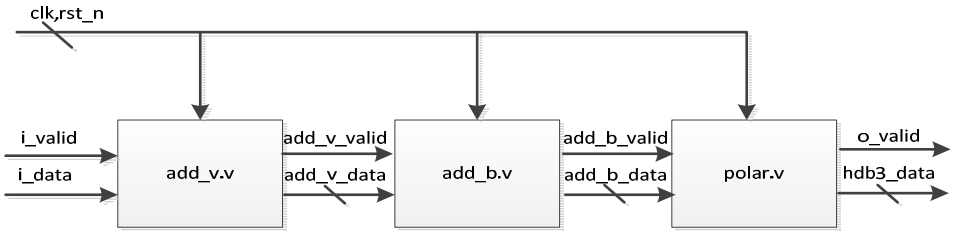

图 2 HDB3 编码原理框图

由于仿真测试的需要，与下面实现相对应的测试，都是使用 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1 1 作为原始输入码流作为测试。为了在 Modelsim 下观察方便，定义了如下参数：

```
parameter HDB3_0 = 2'B00; //0
parameter HDB3_1 = 2'B01; //+1
parameter HDB3_B = 2'B11; //-1
parameter HDB3_V = 2'B10; //V code,-2

parameter HDB3_P = 2'B01; //+1
parameter HDB3_N = 2'B11; //-1
parameter HDB3_Z = 2'B00; //0
```

3.1 add_v 模块的实现

此模块的功能是检测输入的码流中，如果出现了连续 4 个 0 的情况，则将第 4 个 0 替换为 V 码，下面

是实现此功能的关键代码。

```
o_valid <= i_valid;
if(i_valid)
    begin
    data_buf <= {data_buf[1:0],i_data};
    if(i_data==1)
        begin
        o_add_v <= HDB3_1;
        end
    else  //(i_data==0)
        begin
        o_add_v <= (data_buf==3'b000) ? HDB3_V : HDB3_0;
        end
    end
```

下图是功能仿真的时序图，图 3 中的红色框处就是出现 4 个连 0 替换成 V 码的位置。

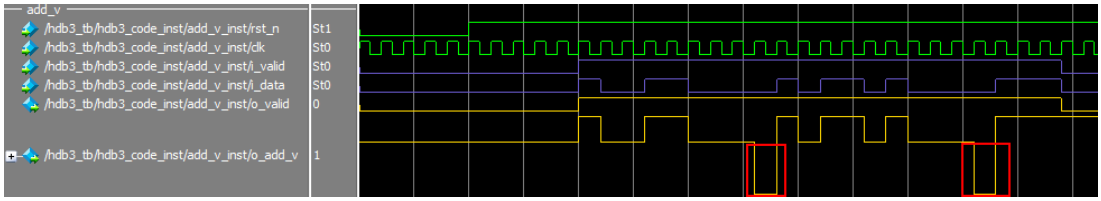

图 3 add_v 模块的功能仿真图

3.2 add_b 模块的实现

add_b 模块对 2 个 V 码之间 1 的个数进行检测，如果个数为偶数，则将 4 个连 0 个的第一个 0 替换为 B 码。下面是具体的 VerilogHDL 实现代码：

```
{o_valid,valid_buf[2:0]} <= {valid_buf[2:0],i_valid};//有效标志位的延时

if(i_data==HDB3_V) //如果输入的数据是 V 码
    begin
        v_occur<=1;//置 V 码出现了的标志位
        odd <= 0;   //重新设置奇偶标志位
    end
else if(i_data == HDB3_1)//如果输入的数据是 1
    begin
        odd <= ~odd;     //奇偶标志位取反
    end

//如果当前输入的数据是 V 码，而且不是第一次出现 V 码，而且 V 码间的 1 的个数为偶数
if(i_data==HDB3_V && v_occur && !odd)
    {o_add_b,data_buf[5:0]} <= {HDB3_B,data_buf[3:0],i_data};          //增加 B 码
else
    {o_add_b,data_buf[5:0]} <= {data_buf[5:0],i_data};        //数据无变换
```

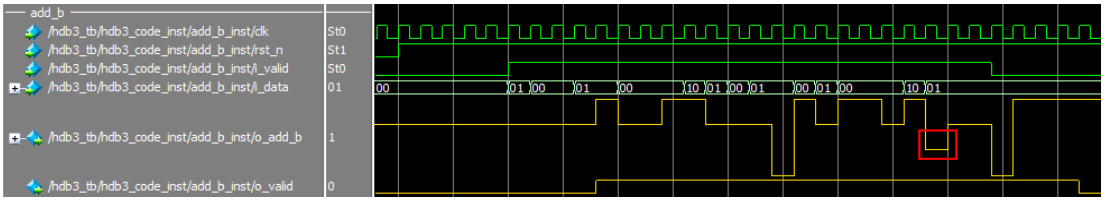

图 4 add_b 模块的功能仿真图

图 4 是具体的功能仿真图，红色框处就是增加的 B 码的位置。

3.3 polar 模块的实现

polar 模块的功能是实现双极性转换，即对原数据码流中的 1 与增加的 B 码一起进行正负极性交替变换；第一个 V 码与第一个数据 1 的极性一致，从第二个 V 码开始则正负极性交替变换。

```
if(i_data==HDB3_V)          //输入的是 V 码
        begin
        hdb3_data <= v_polar ? HDB3_P : HDB3_N;
        v_polar <= !v_polar;
        o_valid <= 1;
        end
else if(i_data==HDB3_1 || i_data==HDB3_B)  //  输入的是 HDB3_1 || HDB3_B
        begin
        hdb3_data <= b_polar ? HDB3_P : HDB3_N;
        b_polar <= !b_polar;
        o_valid <= 1;
        end
else if(i_data==HDB3_0)             //输入的是 0 码
        begin
        hdb3_data <= HDB3_Z;
        o_valid <= 1;
        end
```

图 5 是具体的功能仿真波形图，黄色波形就是最终的 HDB3 码流输出，图中波形与本文第 1 小节列举的例子一致，证明代码的正确性。

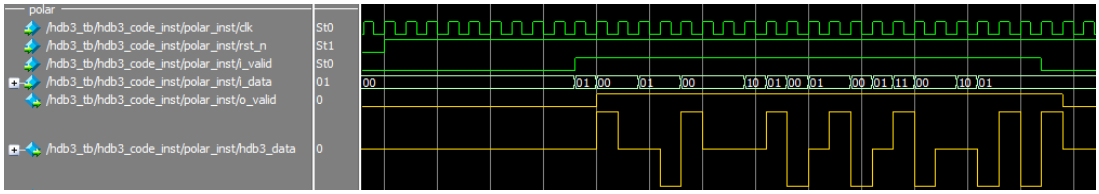

图 5 polar 模块的功能仿真图

4 基于 VerilogHDL 的 HDB3 解码实现

上一小节已经介绍了 HDB3 的编码实现，本小节将介绍 HDB3 的解码过程，图 6 给了解码的原理框图，较编码而已，解码更加容易。

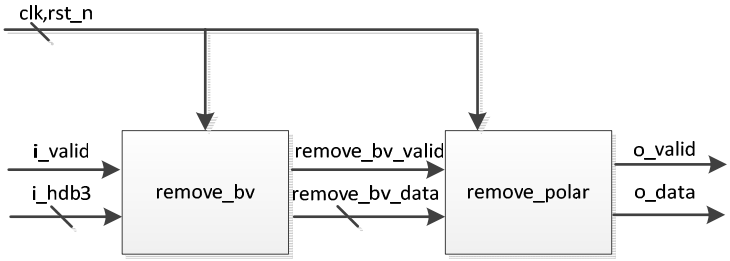

图 6 HDB3 解码原理框图

4.1 remove_bv 模块的实现

此模块的功能是对输入码流中检测出现 X00X 或 X000X 的情况，其中的两个 X 是同极性的情况，如果是 X00X，则说明是 B00V 的情况，需要将其替换为 0000；如果检测到的是 X000X 的情况，说明最后的 X 是 V 码，将其替换为 X0000。下面是具体实现的关键代码。

```
{o_valid,valid_buf} <= {valid_buf,i_valid};
if(i_hdb3==data_buf[5:4] && data_buf[3:0]==4'h0 && (i_hdb3==HDB3_P || i_hdb3==HDB3_N))
//检测到X00X
    {o_data,data_buf[7:0]} <= {data_buf[7:6],8'h00};
else if(i_hdb3==data_buf[7:6] && data_buf[5:0]==6'h00 && (i_hdb3==HDB3_P || i_hdb3==HDB3_N)) //检测到X000X
    {o_data,data_buf[7:0]} <= {data_buf[7:6],8'h00};
else
    {o_data,data_buf[7:0]} <= {data_buf[7:0],i_hdb3};
end
```

图 7 是具体的功能仿真时序图，将 B 码与 V 码都去除。

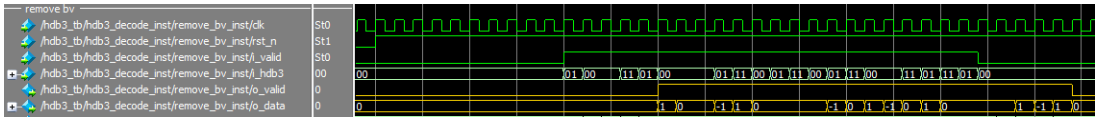

图 7 remove_bv 模块的功能仿真图

4.2 remove_polar 模块的实现

此模块的功能很简单，即将码流的中+1 与-1 都变换为 1。下面是具体实现的 VerilogHDL 代码。

```
if((i_rm_bv == HDB3_P) || (i_rm_bv == HDB3_N))
    begin
        o_data <= 1;
        o_valid <= 1;
    end
else if((i_rm_bv == HDB3_Z))
    begin
        o_data <= 0;
        o_valid <= 1;
    end
end
```

图 8 是经过去除极性后的功能仿真图，图中最下一行得出的就是解码得到的原始数据码流，与第 1 小节列举的例子一致。

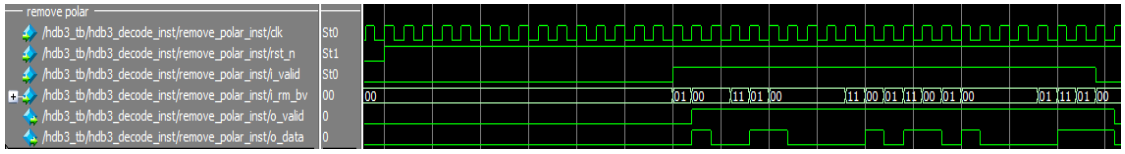

图 8 remove_polar 模块的功能仿真图

5 结论

本文给了 HDB3 编解码的总体设计方案，并给出了具体的 VerilogHDL 代码实现，最终通过 ModelSIm 软件进行仿真并给出仿真波形，验证了代码的正确性。

参考文献

[1] 朱勤为,唐宁,赵明剑. 利用 FPGA 实现 HDB3 编解码功能[J].电子设计工程,2009,(12):76-79.doi:10.3969/j.issn.1674-6236.2009.12.029.
[2] Hdb3 之 FPGA 实现[EB/OL]. http://bbs.ednchina.com/BLOG_ARTICLE_2060189.HTM.