|  |
| --- |
| ADC与二进制 |
|  |
|  |
| 作者：HateHanzo |
| 联系方式：HateHanzo@163.com |

版权所有 侵权必究

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修改文件 | 描述 | 作者 | 时间 |
| v1.0 |  | 初稿 | HateHanzo | 20220531 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**缩略语与约定符号**

**缩略语**

|  |  |  |
| --- | --- | --- |
| **英文简写** | **英文全称** | **中文描述** |
| AHB | Advanced High-performance Bus | 先进高性能总线 |
| SRAM | Static Random-Access Memory | 静态随机存取存储器 |
| FSM | Finite State Machine | 有限状态机 |
| SR-Latch | Set-Reset-Latch | SR锁存器 |
| ICG | Integrated Clock Gating Cell | 门控时钟 |

**约定符号**

1、对于某信号名A，若无特殊声明，A\_d1表示该信号延时一拍，A\_d2表示延时两拍，以此类推。如ahb\_addr\_d1表示信号ahb\_addr延时一拍。

**目 录**

[**修改记录** I](#_Toc110501757)

[**缩略语与约定符号** II](#_Toc110501758)

[**目 录** 0](#_Toc110501759)

[**1 二进制原码与补码** 1](#_Toc110501760)

[**2 ADC数据表与平均** 1](#_Toc110501761)

[**参考文献** 3](#_Toc110501762)

[**附 录** 4](#_Toc110501763)

**1 二进制原码与补码**

计算机中对数据的运算采用补码进行，谈到二进制首先要谈到的一个就是符号问题，以4bit二进制数据为例子，如果其代表无符号数，那么其可表示的范围是0～15；如果其代表有符号数，那么可表示的范围是-8～+7，如图 1所示。无符号数的补码也就是其原码，有符号数的最高位是符号位（1负0正），补码计算一个简单方法是：对原码所有bit取反然后加1。例如+8的原码是1000，对所有bit取反为0111，0111+0001=1000。可以看到使用补码可以将符号位也一并带入计算，非常方便。

关于符号位扩展：-4用4bit数表示为1100，那么用8bit如何表示呢？答案是将4bit数的符号位扩展到剩余高位，也就是11111100。可以用上述中提到的方法进行验证，8bit 4表示为00000100，取反为11111011，11111011+00000001=11111100，与扩位产生的数字是一致的。

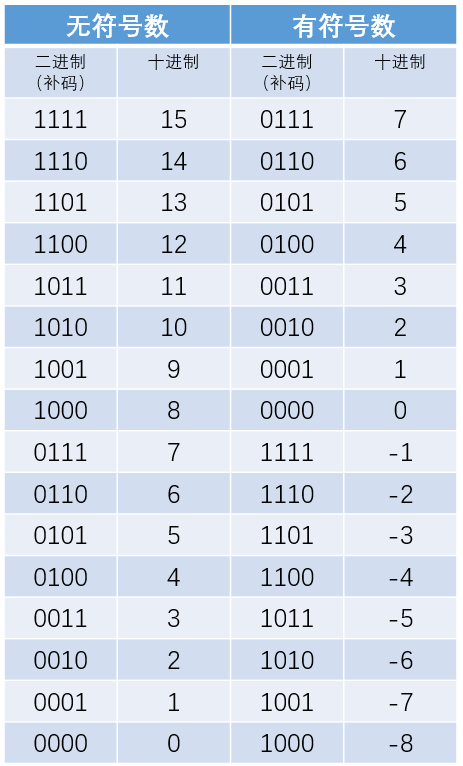


图 1 4bit有符号与无符号二进制数

**2 ADC数据表与平均**

图 2是某款ADC的数据输出量化表，11\_1111\_1111表示正方向电流的最大值，00\_0000\_0000表示负方向电流的最大值，可以看出来与图 1的编码方式是稍微有些许不一致的，所以在处理该电流通道的ADC数据时，首先要将最高位取反，这样得到的数据（补码）才和实际电流值对应，在作平均时需要对当前的adc数据进行扩展：

adc\_sign = en\_icha ? ～adc\_din[9] ： adc\_din[9]；//针对带方向的电流通道的符号位处理

pre\_stuff = en\_icha ? ～adc\_din[9] ： 1’b0 ；//扩展位处理

adc\_d\_cal = {adc\_sign,adc\_din[8:0]}；//与实际电流值对应的adc电流补码数据

//取平均数的RTL写法

reg [15:0] adc\_d\_add ;

reg [15:0] adc\_sum ;

case(ave\_sel)

adc\_d\_add = {{3{pre\_stuff}},adc\_d\_cal,3’h0}；//8 avage

adc\_d\_add = {{4{pre\_stuff}},adc\_d\_cal,2’h0}；//16 avage

adc\_d\_add = {{5{pre\_stuff}},adc\_d\_cal,1’h0}；//32 avage

adc\_d\_add = {{6{pre\_stuff}},adc\_d\_cal }；//64 avage

endcase

always@(posedge clk or negedge rst\_n)

begin

if(!rst\_n)

adc\_sum <= 16’d0 ;

else if(sum\_clr)

adc\_sum <= 16’d0 ;

else if(sum\_adden)

adc\_sum <= adc\_sum + adc\_d\_add

else ;

end

wire [11:0] ave\_x4 = adc\_sum[15:4] ;//截位，取平均

(思考：为什么ADC输出的位数是10bit，取平均后要扩展成12bit呢？)

答：主要是扩位后保留的ADC平均数精度更加准确，但模拟ADC的有效位数仍然是10bit。举个例子，假如10bit模拟ADC的LSB为4mv，如果平均后ADC输出的值为0000\_0000\_0101，由高往低截取10bit，那么计算出来的电压值为1\*4mv = 4mv

如果保留12bit，那么计算电压时LSB为1mv，但计算出来的电压值为5\*1mv = 5mv

5mv相比4mv是平均后更加准确的值，更能体现平均的结果，至于为什么是多取2bit，应该是经验使然。

|  |  |  |
| --- | --- | --- |
| 模拟输出[9:0] | ADC value | 电流值（带方向） |
| 11\_1111\_1111 | 1023 | 1717.072 |
| 11\_1111\_1110 | 1022 | 1713.715 |
| …… | …… | …… |
| 11\_0111\_0101 | 885 | 1253.815 |
| 11\_0111\_0100 | 884 | 1250.458 |
| …… | …… | …… |
| 10\_1110\_1001 | 745 | 783.844 |
| 10\_1110\_1000 | 744 | 780.4871 |
| …… | …… | …… |
| 10\_0111\_0101 | 629 | 394.4397 |
| 10\_0111\_0100 | 628 | 391.0828 |
| …… | …… | …… |
| 10\_0000\_0000 | 512 | 1.67845 |
| 01\_1111\_1111 | 511 | -1.67845 |
| …… | …… | …… |
| 00\_1010\_0011 | 163 | -1169.89 |
| 00\_1010\_0010 | 162 | -1173.25 |
| …… | …… | …… |
| 00\_0000\_0000 | 0 | -1717.91 |

图 2 某款ADC数据电流通道输出量化表 （有方向）

|  |  |  |
| --- | --- | --- |
| 模拟输出[9:0] | ADC value | 电压值 |
| 11\_1111\_1111 | 1023 | 924.4629 |
| 11\_1111\_1110 | 1022 | 923.3887 |
| …… | …… | …… |
| 11\_0111\_0101 | 885 | 776.2207 |
| 11\_0111\_0100 | 884 | 775.1465 |
| …… | …… | …… |
| 10\_1110\_1001 | 745 | 625.8301 |
| 10\_1110\_1000 | 744 | 624.7559 |
| …… | …… | …… |
| 10\_0111\_0101 | 629 | 501.2207 |
| 10\_0111\_0100 | 628 | 500.1465 |
| …… | …… | …… |
| 10\_0000\_0000 | 512 | 375.5371 |
| 01\_1111\_1111 | 511 | 374.4629 |
| …… | …… | …… |
| 00\_1010\_0011 | 163 | 0.632266 |
| 00\_1010\_0010 | 162 | -0.43945 |
| …… | …… | …… |
| 00\_0000\_0000 | 0 | -174.463 |

图 2 某款ADC数据电压通道输出量化表 （无方向）

**参考文献**

**附 录**