

一种双门限语音端点检测算法的 ASIC 实现

张 羊,姜汉钧,王志华
(清华大学 微电子所, 北京 100084)

摘 要: 设计了一种基于短时幅度和过零率双门限判决算法的语音端点检测 ASIC 实现电路,采用基于异步 FIFO 的动态分帧和 CSD 编码技术大大节省了硬件资源开销,在 Modelsim 和 FPGA 上分别进行了采集的语音信号的检测验证,结果表明此算法具有硬件实现简单、资源消耗小、检测正确率高的优点.

关键词: 双门限;语音端点检测;ASIC;FIFO;CSD;FPGA

中图分类号: TN492

文献标识码: A

文章编号: 1000-7180(2016)11-0069-05

A Double Threshold Speech Endpoint Detection Algorithm Implementation in ASIC

ZHANG Yang,JIANG Han-jun,WANG Zhi-hua

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: A kind of speech endpoint detection ASIC circuit is designed, which is based on double threshold detection algorithm of short-term magnitude and zero-crossing rate. We can greatly save the cost of hardware resources by using dynamic frame partition technology based on asynchronous FIFO and CSD coding technology. We have completed the verification using the collected speech signal in Modelsim and FPGA platform respectively. The results show that this algorithm has the advantages of simple hardware implementation, low resource consumption and high detection accuracy.

Key words: aouble threshold; VAD; ASIC; FIFO; CSD; FPGA

1 引言

语音信号是人与人之间交流信息的载体,它是人类社会活动的基本手段,同时也是实现人机交互的重要前提.经过麦克风采集的语音信号经过前置放大和模数转换后进入数字域进行处理,但是采集的语音信号中并不都是完全有用的语音片段.由于说话人语音的短暂停顿或是突发性的如撞击、爆破等背景噪声的存在,对语音信号的处理效率和抗干扰提出了挑战.语音端点检测是一种通过特定的算法在背景噪声下区分出语音信号中的有声片段和无声片段,确定出实际有用的语音信号的开始和结束的位置,它是后续完成语音识别、语音增强和压缩编码系统中的重要组成部分.语音端点检测算法类型多样,主要有基于短时能量和过零率的、基于倒谱特

征的、基于信息熵的.这些检测算法在检测性能方面都有各自的优势,然而语音通信需要达到实时性,目前主流的 DSP 芯片并行度不高^[1],采用 ASIC 来实现可以最终在一块芯片上集成语音采集、信号处理、压缩编码等算法功能,具有研究价值.出于 ASIC 实现节省面积和成本的考虑,所采用的实现算法在保证足够高的检测准确率的前提下应实现简单、硬件资源消耗要小.基于上述原因,本文改进了短时幅度和过零率的双门限语音信号端点检测算法,采用了动态分帧和 CSD 编码技术降低了硬件实现资源,并给出了检测结果.

2 双门限检测算法

语音信号一般可分为无声段、清音段和浊音段.无声段短时幅度最低,浊音段短时幅度最高,清音段

短时幅度居于前两者之间,而清音的短时过零率最大,浊音和噪声次之,且浊音和噪声短时平均过零率相当。双门限算法首先利用短时幅度差别将无声段和浊音段区分开,其次再利用过零率差别将无声段和清音段区分开,弥补了清音的短时幅度比较小而容易被误检为无声段的错误,通过这样两级检测,最终将语音信号的起止点准确地检测出来。

双门限端点检测算法首先为短时幅度设置两个门限,幅度高的门限 ITU 和幅度低的门限 ITL,为短时过零率设置门限 ZCR。合理地设置门限值,当短时幅度超过 ITL 或者过零率大于 ZCR 的地方即为清音开始的地方。由于一些撞击或爆破类的突发性噪声也可能造成短时幅度或过零率很高,但这些和真正的语音信号相比持续时间都很短,为此还要设置最大静音帧长 maxsilence 和最小语音帧长 minlen。整个语音端点检测主要通过四个状态即静音态、过渡态、语音态和结束态来完成。在静音态,如果某帧的短时幅度超过低门限 ITL 或者过零率超过门限 ZCR,则开始进入过渡态。在过渡态,其短时幅度和过零率比较小,不能确信是否真正处在语音态,当短时幅度值低于 ITL 而且过零率低于 ZCR 时则回到静音态。无论是在静音态或过渡态,如果其短时幅度值高于 ITU,则真正进入了语音态。在语音态如果短时幅度低于 ITL 而且过零率低于 ZCR 时开始记录静音帧帧长 silence,当 silence 达到 maxsilence 时语音将结束,判断这时记录的语音帧长 count 是否大于 minlen,如果大于 minlen,则表示记录的是真正的语音,否则就是突发噪声,应当跳过这部分噪声继续扫描后面的语音数据。

3 电路实现

3.1 预加重

前端采集得到的语音信号经过 AD 量化之后一般通过一阶高通 FIR 滤波器进行预加重,可使得语音信号的高频分量得到提升,其频谱更加平坦。预加重的滤波函数中的系数为小数,硬件实现起来比较麻烦,本文通过用分数 $15/16$ 代替 0.94 ,因为 $15/16$ 可以非常方便地通过移位来实现。这样由滤波函数得到的时域一阶差分表达式为

$$\begin{aligned} y[n] &= x[n] - \frac{15}{16}x[n-1] \\ &= x[n] - (x[n-1] - \frac{x[n-1]}{16}) \quad (1) \end{aligned}$$

式(1)中除以 16 可以用右移 4 位来实现,这样

通过简单的加减法、延时和移位运算即可硬件实现预加重,消耗的硬件资源非常小。后文提到当用 MATLAB 中的 wavrecord 函数配合电脑上的麦克风进行语音数据源采集量化时采用了 int16 数据类型,它是 16 bit 有符号数,在进行 16 bit 移位的时候需要扩展符号位,否则移出位自动补零,将会导致负数移位的结果变成正数。

3.2 分帧

分帧是将非平稳的语音信号分成时间在 10 到 30 ms 之间短时平稳的各帧进行处理。为了保证帧与帧之间的连续平滑性,设计采用了交叠分帧的方式。本设计中帧长为 256,帧移为 128,采用动态分帧的方式实现。使用了两个异步 FIFO 去实现,异步 FIFO 的实现框图如图 1 所示。异步 FIFO 用于跨时钟域的数据传输。它是基于双端口的 RAM 实现,在读写时钟的控制下分别产生读写地址,在读写信号有效时把数据从读地址指向的 RAM 中读出或将数据写入到写地址指向的 RAM 中。异步 FIFO 的设计关键在于空满标志位的判断,当写地址追上读地址则 FIFO 满,当读地址追上写地址,则 FIFO 空。由于读写地址属于不同的时钟域,所以在比较之前需要先将读写地址进行时钟同步处理,另外同步打拍解决亚稳态只适用于传输单比特信号,而在判断 FIFO 空满状态时需要比较读地址和写地址这两个异步时钟域的多比特情形,设计中采用了格雷码,这种编码相对于二进制编码可以保证任意相邻两个数之间转化只有一位发生变化,大大减小了同步过程中地址出错的概率。

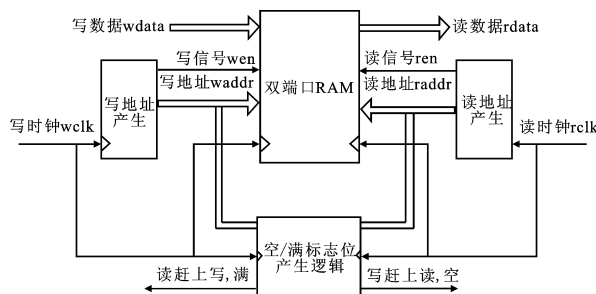


图1 异步 FIFO 实现框图

动态分帧的实现框图如图 2 所示,其中 fifo256 的大小为一帧语音长度 256,分成两个数据块, fifo128 的大小为 128。设计的 fifo256 的读时钟频率为写时钟频率的 2 倍,而 fifo128 的读写时钟频率都与 fifo256 的读时钟频率相同。具体的 fifo256 和 fifo128 的读写信号时序如图 3 所示(设计的 fifo 读写信号低电平有效),复位 rst 时, fifo256 的读信号

fifo256_ren、写信号 fifo256_wen, fifo128 的读信号 fifo128_ren、写信号 fifo128_wen 都为高电平. 复位后 fifo256_wen 信号一直为低电平(写允许), 等到 fifo256 写完第一块的 128 个数据之后 fifo256_ren 变成低电平允许读出 fifo256 的第一块数据, 同时 fifo128_wen 信号变成低电平将 fifo256 的第一块数据写入到 fifo128 中, 这时 fifo256 也在同时写入第二块数据, 当 fifo256 的第一块数据读完之后, 此时 fifo128 的数据写完, 而 fifo256 的第二块数据写了一半, 这时 fifo256_ren 和 fifo128_wen 变成高电平, 而 fifo128_ren 变成低电平, 允许将 fifo128 中的数据读出, fifo128 读完后 fifo256 中第二块数据也写完, 此时 fifo256_ren 和 fifo128_wen 变成低电平, fifo128_ren 变成高电平, 如此循环. 数据选择器(MUX)实现选通输出, 第一次选择输出 fifo256 读出的数据, 第二次选择输出 fifo128 读出的数据, 这样实现将一块 128 点的数据重复输出两次. 分帧以后的数据流在图中按标号 1 和 2 组成第一帧, 2 和 3 组成第二帧, 以此类推得到分帧后的语音数据. 分帧后, 波形平滑了许多, 虽然略微有延时, 但这样实现的实时分帧, 与静态分帧相比, 速度提高了很多, 有效地减少了存储空间浪费.

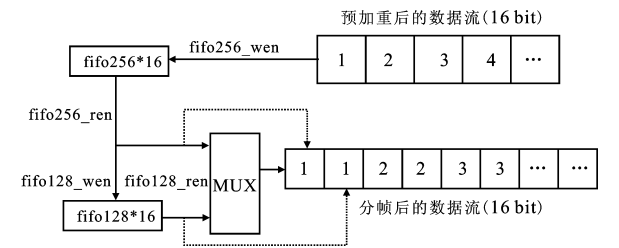


图 2 动态分帧实现框图

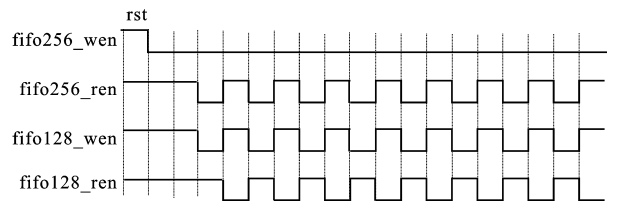


图 3 两块 FIFO 读写信号时序

3.3 加窗

为了减小前述语音分帧的截断效应, 需要加窗处理. 矩形窗的主瓣宽度小, 频谱分辨率较高, 但其旁瓣衰减不够, 而汉明窗的旁瓣衰减较大, 因此窗函数选用汉明窗, 即将分帧后的数据每帧乘上对应的窗函数的值. 窗函数的值是一系列小数, 具体到硬件实现时, 先将窗函数的值进行 16 bit 量化后再与每

帧的数据相乘, 16 bit 乘法器实现起来占用很多的硬件资源, 本文采用 CSD(canonic signed digit)编码的技术^[2-3], CSD 编码将只用 0 和 1 两个数表示的二进制编码按 0, 1 和 -1 三个数表示的方式进行编码, 这种编码方式得到的编码值不存在连续两个非零位, 而且一个数的 CSD 编码是所有编码方式中含有非零位最少的. 将二进制编码转化为 CSD 编码过程如图 4 所示: 对于二进制数 0111_0110_1011_1111, 从最低位起向最高位移动查找到连续两个非零位 1, 停下来将“11”转化为“0 $\bar{1}$ ”, 其中 $\bar{1}$ 代表 -1, 同时向高位进位 1, 将这个进位 1 加到高位数上得到第一次的编码数据, 第二次再继续从最低位开始寻找连续的非零位“11”, 将它按同样的方法转化为“0 $\bar{1}$ ”, 同时向高位进位 1 并把这个进位加到高位上去, 如此循环编码几次, 直到没有找到连续的非零位“11”为止, 此即为最终的 CSD 编码值. 可以看到原来的二进制数有 12 个非零位, 而通过 CSD 编码后的结果只有 5 个非零位. CSD 编码后通过加减法和移位的方式代替乘法, 可以大大节省 16 bit 乘法算法所消耗的硬件资源, 最后再将计算的结果进行右移 16 位. 如前文所述, 右移过程中仍然需要扩展符号位.

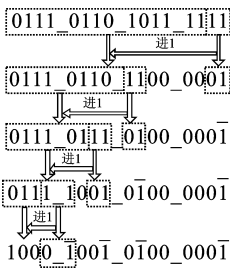


图 4 CSD 编码示意图

3.4 短时幅度与过零率计算

将加窗后的各帧数据进行绝对值求和即可得到各帧的短时幅度 Amp, 求绝对值时通过判断加窗后的数据的最高位, 如果最高位为 0, 代表数据为正数, 求绝对值时保持不变; 如果最高位为 1, 代表数据为负数, 求绝对值时通过将数据的各位取反加 1. 求完绝对值之后需要将各帧数据进行累加输出各帧短时幅度. 累加在每帧数据内进行, 一帧数据累加完后将累加的结果输出同时将累加器清零开始下一帧数据的累加.

过零率 Zero 的计算先将加窗后的数据进行一个时钟的延时, 然后将延时前的每帧数据和延时后的每帧数据在每个时钟周期内进行最高位的判别,

在数据都不为 0 的前提下如果数据最高位不同记判别结果结果为 1, 否则记判别结果为 0. 最后将这些判别结果在一帧数据范围内进行累加.

3.5 端点检测

端点检测的算法采用有限状态机实现, 重点在于判别门限的选取. 传统的算法选取的判别门限要求最大值、最小值及方差^[4-9], 这些都不利于硬件实现, 本设计在大量实验的基础上得到了适于节约硬件资源实现的短时幅度和过零率双门限. 门限的设定基于背景噪声, 在背景噪声不是很大的情况下, 取前 8 帧数据的各帧短时幅度和各帧过零率求平均值得到 amp_8_average, amp_8_average, 则短时幅度高低门限 ITU 和 ITL 以及过零率门限 ZCR 设为

$$ITU = 128 * \text{amp_8_average} \tag{2}$$

$$ITL = 64 * \text{amp_8_average} \tag{3}$$

$$ZCR = 2 * \text{zero_8_average} \tag{4}$$

状态机分为 5 个状态, S0 代表初始态, S1 代表静音态, S2 代表过渡态, S3 代表语音态, S4 代表结束态. Zhen_count 代表帧计数器, count 代表语音帧计数器, silence 代表静音帧计数器, 状态机的状态转移图如图 5 所示.

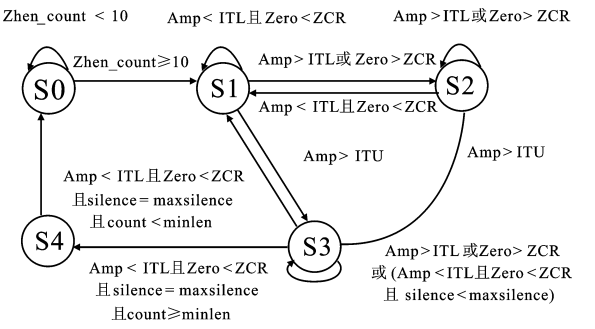


图 5 端点检测状态转移图

状态机的具体动作如下: 在初始态 S0, 复位 count 和 silence. 当 zhen_count 大于等于 10 帧时, 状态跳转到 S1, 否则继续留在 S0 态. 在静音态 S1, 如果短时幅度高于 ITU, 则直接跳转到 S3 语音态, 同时 count 的值加 1; 如果短时幅度大于 ITL 且小于 ITU 或者过零率大于 ZCR, 则跳转到 S2 过渡态, 同时 count 的值加 1; 如果短时幅度小于 ITL 而且过零率小于 ZCR, 则停留在 S1 态, 同时 count 清零. 在过渡态 S2, 如果短时幅度高于 ITU, 则跳转到 S3 语音态, 同时 count 的值加 1; 如果短时幅度大于 ITL 且小于 ITU 或者过零率大于 ZCR, 则保持在 S2 态, 同时 count 的值加 1; 如果短时幅度

小于 ITL 而且过零率小于 ZCR, 则回到静音态 S1, 同时 count 清零. 在语音态 S3, 如果短时幅度高于 ITL 或过零率大于 ZCR, 则仍然保持在 S3 态, 同时 count 加 1; 如果短时幅度小于 ITL 而且过零率小于 ZCR, 则将 silence 加 1, 如果 silence 小于 maxsilence, 静音帧数没达到静音的最大帧数, 状态仍然留在在 S3, 同时 count 加 1; 如果 silence 等于 maxsilence, 静音帧达到了最大帧数, 而且 count 小于 minlen, 语音帧数小于最小语音长度, 认为是噪声引起的波动, 并非实际的语音信号, 状态回到 S1 态, 同时 count 和 silence 将清零; 如果 silence 等于 maxsilence, 静音帧达到了最大帧数, 而且 count 大于 minlen, 语音帧达到了最小语音长度, 是语音信号, 状态跳转到 S4 结束态. 在结束态 S4, count 和 silence 都清零, 在下一帧到来时回到状态 S0. 语音开始点 startpoint 为在状态 S1 或 S2 状态即将进入状态 S3 时的 zhen_count 的值减去 count 的值; 语音结束点为在状态 S3 即将进入状态 S4 时的 count 的值加上 startpoint 的值.

4 检测结果

借助于 matlab 里的 wavrecord 函数用电脑上的 mic 可以采集一段语音信号用来验证该算法的检测正确性. 在采样率为 16 kb/s 下采集 10 s 的数据, 数据中有效语音的地方为“徐良宵”, 在 model-sim 上的仿真结果如图 6 所示, “徐良宵”三个字开始的帧数分别为 147~216 帧, 327~383 帧, 505~556 帧. 在 matlab 中的检测结果如图 7 所示, 二者对比起止点可以确定本实现算法的正确性. 最后利用起止点帧数绘制出去除无声段的语音信号如图 8 所示.

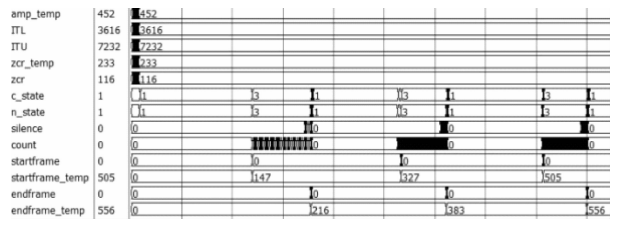


图 6 modelsim 上端点检测的结果

为进一步验证端点检测的正确性, 利用深圳依元素科技有限公司的 XC3SD1800A 的 FPGA 开发平台进行了验证, 将采集的语音信号源数据存入到 RAM 缓存中, 在 ISE 环境里调用 RAM IP 核提供语音数据输入, ISE 环境下综合的结果如表 1 所示.

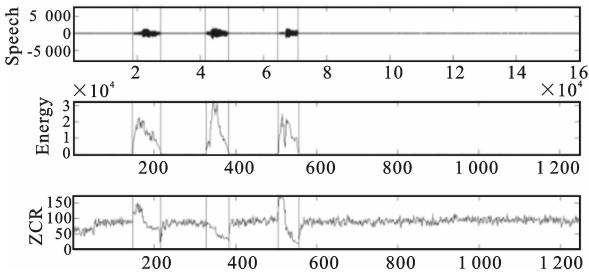


图 7 matlab 上端点检测结果

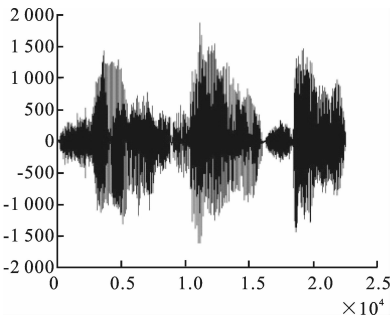


图 8 检测出的去除无声段的语音信号

表 1 ISE 综合资源利用情况

Logic Utilization	Used	Avail	Util/%
Number of Slice Registers	6 568	33 280	19
Number of Slice LUTs	7 369	33 280	22
Number of bonded IOBs	106	519	20
Number of BUFGMUXs	6	24	25

将本设计所消耗的资源与其他设计对比结果如表 2 所示。

表 2 本设计与其他文献资源消耗对比

参考文献	Silce Registers	Slice LUTs	Block RAM/FIFO	DSP48Es
文献[8]	1 199	960	16	3
文献[6]	16 923	20 302	41	30
本文	6 568	7 369	0	0

虽然比参考文献[8]中多消耗了寄存器和 LUT 资源,但是并没有用到 RAM、FIFO 和 DSP48Es 资源,使用 CSD 编码实现乘法器可以比普通乘法器节省 33%的资源,本设计在资源消耗上是非常有竞争优势的.对更多的语音信号样本进行检测表明在 200 个词源的条件具有 90%以上的准确率。

5 结束语

本文在短时幅度和过零率的双门限检测算法基础上硬件实现了语音端点检测,利用了动态分帧和 CSD 编码技术有效地减少了硬件资源开销,最后在

modelsim 和 FPGA 上进行了检测结果的验证。

参考文献:

[1] 靳月英. 语音识别 ASIC 中端点检测算法研究与实现[J]. 计算机与现代化, 2011,26(12):57-59.

[2] Sait S M, Farooqui A A. A novel technique for fast multiplication[C]//Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on Computers and Communications. Washington DC, USA: IEEE, 1995:109-114.

[3] Hashemian R. A new method for conversion of 2'S complement to canonic signed digit number system and its representation[C]//Conference Record of the Thirtieth Asilomar Conference on Signals, Systems and Computer. [s. l.]: IEEE, 1996. 904-907.

[4] Rabiner L R, Juang B H. Fundamentals of Speech recognition [M]. Boston: Prentice Hall, 1993.

[5] Verteletskaia E, Sakhnov K. Voice activity detection for speech enhancement applications[J]. Acta Polytechnica, 2010,26(11):100-105.

[6] Jung J, Jin S, Kim D, et al. A voice activity detection system based on fpga[C]// Control Automation and Systems (ICCAS), 2010 International Conference on. Berlin: IEEE, 2010: 2304-2308.

[7] Noguchi H, Takagi T, Yoshimoto M, et al. An ultralow-power VAD hardware implementation for intelligent ubiquitous sensor networks[J]. IEEE Workshop on Signal Processing Systems (SiPS), 2009,26(11): 214-219.

[8] Liang T, Zhang G J, Zhang X J. Speaker Recognition Algorithm Implementation Based FPGA[J]. Communications Technology, 2008,37(6):343-351.

[8] 周明忠,吉立新. 基于平均幅度和加权过零率的 VAD 算法及其 FPGA 实现[J]. 信息工程大学学报, 2010,11(6):713-718.

[9] 李昱,林志谋,黄云鹰,等. 基于短时能量和短时过零率的 VAD 算法及其 FPGA 实现[J]. 电子技术应用, 2006,32(9):110-113.

作者简介:

张 羊 男,(1991-),硕士研究生. 研究方向为数模混合电路设计. E-mail:zhangyang13@mails.tsinghua.edu.cn

姜汉钧 男,(1980-),博士生导师. 研究方向为数模混合电路设计.

王志华 男,教授,博士生导师. 研究方向为集成电路和系统的设计方法学、用于医疗和通信的低功耗模拟与射频集成电路设计、高速实时信号处理.