

一种 LZ77 解码电路及其工作方法

技术领域

本发明公开一种 LZ77 解码电路及其工作方法，属于数据解码的技术领域。

背景技术

所述 LZ77 是一种基于字典的算法，就是将已经编码的信息作为字典，如果要编码的字符曾经出现过，就用该字符串的出现位置以及长度来替代这个字符串，从而达到压缩的目的。几乎我们日常使用的所有通用压缩工具，如 ARJ, PKZip, WinZip, LHArc, RAR, GZip, 和许多硬件中内置的压缩算法，都有 LZ77 算法的应用。

经过 LZ77 压缩后的数据有两种形式，literal、pair (length, distance)，literal 是未经压缩处理的字符，pair 是一个二维元组，包含 length 和 distance 两个信息，代表的是不定长的字符串序列，在进行数据解压时 literal 不需要进行特殊处理，而 pair 则需要通过查找窗 (search_window) 进行数据恢复，所以电路对 pair 的处理速度决定了整个电路的解码速度。

中国专利文献：CN109716659A 高性能单流 LZ77 压缩技术用于高性能单流数据压缩的技术包括基于输入数据流更新索引数据结构的计算设备。输入数据流被划分为多个块。每个块具有预定的长度，例如 136 个字节，并且与先前的块重叠预定的量，例如 8 个字节。计算设备使用索引数据并行处理多个块以生成多个令牌流。令牌包括文字令牌和引用令牌，所述引用令牌指的是输入数据流中较早的匹配数据。该文献能够有效的提高现有 Gzip 解压缩算法核心组成部分 LZ77 算法的解码效率。但是，该文献并没有给出相应的地址映射电路功能模块及输出预处理电路功能模块。因此对数据恢复时电路对 pair 的处理速度难以得到直观的控制。

发明内容

为了解决上述技术问题，本发明公开一种 LZ77 解码电路。

本发明还公开上述解码电路的工作方法。

本发明通过改进的查找窗结构、对输入、输出数据分别进行预处理，进而有效提高 LZ77 解码速度。

本发明详细的技术方案如下：

一种 LZ77 解码电路，其特征在于，其中待解压数据为可被恢复为包含 literal 和 pair 的数据，由于 LZ77 压缩通常会和其它压缩算法配合使用，对数据进行多级压缩，但是在进行 LZ77 解码时待解压数据必须可以被恢复成 literal 或者 pair 的形式才能利用本发明所述的解码电路进行工作，见图 10、11，以 GZIP 解压为例，待解压数据先经 Huffman 解码电路进行解压缩后，再进行 LZ77 解压缩；

按照 pair 数据流方向，所述 LZ77 解码电路包括：

输入/输出控制电路、输入缓存、输入预处理电路、地址映射电路、RAM 输入/输出控制电路、RAM、输出预处理电路、输出缓存，所述 RAM 和输出缓存共同构成了查找窗；其中，通过所述 RAM 输入/输出控制电路向查找窗的 RAM 查找 pair 需要的数据；

按照 literal 数据流方向，所述 LZ77 解码电路包括：

输入缓存、输入预处理电路和输出缓存，待解压数据中的 literal 数据代表未被 LZ77 压缩的部分，通过所述输入预处理电路将所述 literal 数据直接放到输出缓存中；

输出缓存中的数据更新所述 RAM，所有解压出的数据和直接被放到输出缓存的 literal 数据，被先储存在输出缓存中，同时输出缓存对数据进行计数，当数据量到达一定阈值后，将对应数据通过 RAM 输入/输出控制电路写入查找窗的 RAM 进行保存，进而实现对 RAM 的更新，如图 1 所示；

其中，所述输入/输出控制电路用于控制待解压数据的输入和解压后的数据输出；

所述输入缓存，用于缓存通过输入/输出控制电路输入的数据，在输入缓存中需要限制输入缓存大小的下限，但并不是一个固定的值，而是跟 $\max(\text{literal_length}, \text{pair_length})$ 有关，即 literal bit 长度或者 pair bit $(\text{distance} + \text{length})$ bit 长度中的最大值，进一步说，所述输入缓存应至少放下 literal 或者 pair 中最长的一个；

所述输入预处理电路，用于将待解压数据预处理为 literal 数据和/或 pair 数据；在本技术领域，在具体的压缩算法中，会将 literal 和 pair 用某种“标记”加以区分，以 GZIP 压缩算法中 LZ77 的应用为例，literal 用 0-255 进行表示，而 length (pair 的第一个分量) 用 257~285 外加扩展位进行表示，解压时通过数据的范围就可以区分 literal 和 length，而 length 和 distance 成对出现，length 后面的数据就是 distance；

所述地址映射电路，用于对 pair 数据进行解析，如果 pair 数据所需的数据全部或部分位于 RAM 中（当所需数据全部存在于输出缓存中时，直接从输出缓存中 copy length 个数据）：将所述 pair 数据中的 distance 映射为 RAM 可识别的地址 (ram_rd_st_addr)，根据所述 pair 数据中的 length 计算出需要去 RAM 中读取数据的次数 (ram_rd_cnt)，根据输出缓存中数据的个数 (out_buf_cnt)，根据 RAM 当前指向的地址 (ram_cur_addr)，以确定待查找数据的位置、读取次数和长度信息；

所述输出预处理电路，根据数据的位置、读取次数和长度信息从 RAM 读回的数据，并对数据进行裁剪、拼接，最终形成解压后的数据；

所述输出缓存，用于缓存解压后的数据；

所述 RAM 输入/输出控制电路，用于控制 RAM 内的数据写入和读出；

所述查找窗，包括：RAM 和输出缓存，所述输出缓存对 RAM 进行更新。

根据本发明优选的，所述查找窗包括双口 RAM 和输出缓存；所述查找窗的最小架构中双口 RAM 的与使用 LZ77 压缩时的“字典”的大小一致，比如 GZIP 算法中在进行 LZ77 压缩时使用的是 32KByte 大小的“字典”，RAM 的大小应该跟“字典”大小一致；所述输出缓存本身的大小大于等于 RAM 每个地址对应的数据宽度。其中，RAM 每个地址对应的数据宽度为 nByte，比如 32KByte 的 RAM 使用的是 4K*8Byte 的形式，此时 n=8，则输出缓存的大小应大于等于 8Byte，解压过程中，输出缓存对解压后的数据进行计数，每当计数值大于等于 8Byte 后就将这 8Byte 的数据通过 RAM 输入/输出控制模块写入 RAM，同时将这一笔数据通过输入/输出控制模块输出，在这之后这一笔数据会被输出缓存以移位的方式丢弃，计数值相应减 8。此技术特征避免了频繁读写 RAM 造成的时钟资源的浪费，并且在解压 pair 时可以在一个时钟周期内读出多 Byte 数据，充分利用了 RAM 和输出缓存电路，进而提高了解码速度。

上述解码电路的工作方法，其特征在于，包括：

- 1) 解码电路读入一笔待解压数据放至输入缓存;
- 2) 将待解压数据预处理为 literal 数据和/或 pair 数据, 并将 literal 数据送入输出缓存、将 pair 数据送入地址映射电路;
- 3) 将 pair 数据中的 distance 映射为查找窗可识别的地址 (ram_rd_st_addr), 根据 pair 数据中的 length 计算出需要去查找窗中读取数据的次数 (ram_rd_cnt), 根据输出缓存中数据的个数 (out_buf_cnt), 根据 RAM 当前指向的地址 (ram_cur_addr), 以确定待查找数据的位置、读取次数和长度信息;
- 4) 按照待查找数据的位置、读取次数和长度信息从查找窗中读回数据;
- 5) 对读回的数据进行输出预处理: 进行裁剪、拼接, 最终得到解压后的数据;
- 6) 重复步骤 1) -5), 直至所有待解压数据解压完成。

根据本发明优选的, 所述工作方法还包括:

将所述 literal 数据和所述解压后的数据送入所述输出缓存, 对数据进行计数, 当数据量到达预设阈值后, 输出缓存将对应数据写入 RAM 进行保存, 同时对 RAM 进行更新;

进行解压时单个所匹配的字符串位置存在以下三种情形:

仅存在于 RAM 中;

或仅存在于输出缓存中;

或同时存在于 RAM 和输出缓存中。

根据本发明优选的, 将所述 literal 数据和所述解压后的数据送入所述输出缓存, 当一次待解压数据全部处理完成时, 将输出缓存中的数据输出。

根据本发明优选的, 对 RAM 进行更新是滚动更新, 包括:

将输出缓存中的数据写入 RAM 时, 按照由低地址向高地址, 或者由高地址向低地址的方向将每一笔数据依次写入。以 4K*8Byte 的 RAM, 按照由高地址向低地址的方向写入时举例说明, 用位宽为 12bit 的 ram_cur_addr 代表下一笔数据要写入的地址, 解压开始时, ram_cur_addr 应该指向最高地址 4095, 每写入一笔 8Byte 的数据后, 地址减 1, 当 ram_cur_addr 减到 0 时, 由于二进制的计数特性, ram_cur_addr 再减 1 回到 4095, 这样就实现了对 RAM 的滚动更新。同理, 也可以由低地址 0 开始写, 不过每次写入一次数据后, ram_cur_addr 应该加 1, 也可以实现对 RAM 滚动更新的效果。

根据本发明优选的, 所述步骤 5) 输出预处理包括:

对数据进行裁剪, 裁剪掉冗余数据;

对数据进行拼接, 将裁剪后的数据和后续数据重新拼接成对齐的格式。

由于 pair 数据所代表的字符串是不定长的, 比如在 GZIP 算法中, length 的范围为 3 到 258Byte, 而每次通过 RAM 从查找窗中读回的数据都是该地址全部的数据, 所以需要裁剪掉冗余数据。如图 9, 如果 pair 数据所匹配到的数据只有 c、d、e、f, 这 4 个 Byte, 但是对 RAM 进行读操作的时候是把这个地址所有的 Byte 的 a、b、c、d、e、f、g、h 都读出来的, 所以需要把除有效数据 c、d、e、f 外的数据裁剪掉。裁剪后的数据拼接在输出缓存的高位, 输出缓存中已经有 x、y、z 3 个 Byte, 所以拼接后的输出缓存为 x、y、z、c、d、e、f。以进行 RAM 的更新和数据的输出, 这时对单个 pair 数据的解压操作就已经完成。

根据本发明优选的，所述步骤2)地址映射电路的映射规则包括：

1)地址映射电路对pair数据中的distance的大小进行判断：

1-1)若distance小于等于out_buf_cnt时，则直接从输出缓存中拷贝length个数据；

1-2)若distance大于out_buf_cnt时，则从RAM中拷贝数据，并对length的大小进行判断：

1-2-1)若 $\text{length} \leq \text{distance} - \text{out_buf_cnt}$ ，则待查找的数据只存在于RAM中，

起始地址(ram_rd_st_addr)= $\text{ram_cur_addr} - \lceil (\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数} \rceil$ ，读取次数(rd_cnt)：

若 $\text{length} \leq \lceil (\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数} \rceil$ ，只进行1次读RAM操作， $\text{rd_cnt}=1$ ；

若 $\text{length} > \lceil (\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数} \rceil$ ， $\text{rd_cnt} = \lceil \text{length} - \lceil (\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数} \rceil \rceil / \text{RAM 每个地址对应的字节数}$ ，向上取整；

读取过程为，从起始地址开始对RAM进行读操作，每读一次， ram_rd_st_addr 减1，再对新的 ram_rd_st_addr 进行读操作，直至读够 rd_cnt 次，对RAM的第一次读操作和最后一次读操作所读出的数据可能有冗余；

1-2-2)若 $\text{length} > \text{distance} - \text{out_buf_cnt}$ 时，则从RAM中查找 $\text{distance} - \text{out_buf_cnt}$ 个数据，剩下的 $\text{length} - \text{distance} - \text{out_buf_cnt}$ 个数据从输出缓存中获得。

以上为地址映射规则，此外从RAM中读出的数据还需要经过输出预处理电路的裁剪、拼接才能最终进入到输出缓存中。

本发明的技术优势在于：

1、本发明可广泛的的应用于各种解压缩算法中的LZ77解码部分，本发明仅需根据不同的算法中的literal和pair的“标记”形式来设计对应的输入预处理电路，同时根据压缩时使用“字典”的大小，来选择不同大小的RAM，其它电路部分可以实现通用，方便移植，增加了电路的可重用性。

2、本发明所述查找窗的组成结构及更新方式：使用RAM和输出缓存共同构成查找窗，当输出缓存中的数据达到一定的阈值时才对RAM进行一次写操作，这种操作方式避免了频繁读写RAM造成的时钟资源的浪费，并且在解压pair数据时可以在一个时钟周期内读出多Byte数据，充分利用了RAM和输出缓存电路，提高了解码速度。

3、本发明通过提出与查找窗结构相匹配的地址映射电路的映射规则和输出预处理电路，可以快速得到解压后的数据。

附图说明

图 1 是本发明所述 LZ77 解码电路结构；
图 2 是本发明针对数据解压流程图；
图 3 是本发明输入预处理电路；
图 4 是本发明 RAM 单次写入示意图；
图 5 是本发明 RAM 滚动更新示意图；
图 6 是本发明所述查找窗示意图；
图 7-1 是本发明 pair 所匹配数据仅存在于 RAM 时的示意图；
图 7-2 是本发明 pair 所匹配数据仅存在于输出缓存时的示意图；
图 7-3 是本发明 pair 所匹配数据同时存在于 RAM 和输出缓存时的示意图；
图 8 是本发明的地址映射电路示意图；
图 9 是本发明对读回数据进行裁剪、拼接的示意图；
图 10 是本发明中 GZIP 解码中的 LZ77 解码电路的应用场景示意图；
图 11 是本发明中 GZIP 解码中的 LZ77 解码电路的应用场景流程图。

具体实施方式

下面结合实施例和说明书附图对本发明做详细的说明，但不限于此。

实施例 1、

如图 1、2、3 所示，

一种 LZ77 解码电路，其中待解压数据为可被恢复为包含 literal 和 pair 的数据；

按照 pair 数据流方向，所述 LZ77 解码电路包括：

输入/输出控制电路、输入缓存、输入预处理电路、地址映射电路、RAM 输入/输出控制电路、RAM、输出预处理电路、输出缓存，所述 RAM 和输出缓存共同构成了查找窗；其中，通过所述 RAM 输入/输出控制电路向查找窗的 RAM 查找 pair 需要的数据；

按照 literal 数据流方向，所述 LZ77 解码电路包括：

输入缓存、输入预处理电路和输出缓存，输出缓存中的数据更新所述 RAM；

其中，所述输入/输出控制电路用于控制待解压数据的输入和解压后的数据输出；

所述输入缓存，用于缓存通过输入/输出控制电路输入的数据；

所述输入预处理电路，用于将待解压数据预处理为 literal 数据和/或 pair 数据；

所述地址映射电路，用于对 pair 数据进行解析，将所述 pair 数据中的 distance 映射为 RAM 可识别的地址 (ram_rd_st_addr)，根据所述 pair 数据中的 length 计算出需要去 RAM 中读取数据的次数 (ram_rd_cnt)，根据输出缓存中数据的个数 (out_buf_cnt)，根据 RAM 当前指向的地址 (ram_cur_addr)，以确定待查找数据的位置、读取次数和长度信息；

所述输出预处理电路，根据数据的位置、读取次数和长度信息从 RAM 读回的数据，并对数据进行裁剪、拼接，最终形成解压后的数据；

所述输出缓存，用于缓存解压后的数据；

所述 RAM 输入/输出控制电路，用于控制 RAM 内的数据写入和读出；

所述查找窗，包括：RAM 和输出缓存，所述输出缓存对 RAM 进行更新，如图 6 所示。

实施例 2、

如图 5 所示。如实施例 1 所述一种 LZ77 解码电路，所述查找窗包括双口 RAM 和输出缓存；所述查找窗的最小架构中双口 RAM 的与使用 LZ77 压缩时的“字典”的大小一致，比如 GZIP 算法中在进行 LZ77 压缩时使用的是 32KByte 大小的“字典”，RAM 的大小应该跟“字典”大小一致；所述输出缓存本身的大小大于等于 RAM 每个地址对应的数据宽度。其中，RAM 每个地址对应的数据宽度为 nByte，比如 32KByte 的 RAM 使用的是 4K*8Byte 的形式，此时 n=8，则输出缓存的大小应大于等于 8Byte，解压过程中，输出缓存对解压后的数据进行计数，每当计数值大于等于 8Byte 后就将这 8Byte 的数据通过查找窗输入/输出控制模块写入 RAM，同时将这一笔数据通过输入/输出控制模块输出，在这之后这一笔数据会被输出缓存以移位的方式丢弃，计数值相应减 8。此技术特征避免了频繁读写 RAM 造成的时钟资源的浪费，并且在解压 pair 时可以在一个时钟周期内读出多 Byte 数据，充分利用了 RAM 和输出缓存电路，进而提高了解码速度。

实施例 3、

如实施例 1、2 所述解码电路的工作方法，包括：

- 1) 解码电路读入一笔待解压数据放至输入缓存；
- 2) 将待解压数据预处理为 literal 数据和/或 pair 数据，并将 literal 数据送入输出缓存、将 pair 数据送入地址映射电路；
- 3) 将 pair 数据中的 distance 映射为查找窗可识别的地址 (ram_rd_st_addr)，根据 pair 数据中的 length 计算出需要去查找窗中读取数据的次数 (ram_rd_cnt)，根据输出缓存中数据的个数 (out_buf_cnt)，根据 RAM 当前指向的地址 (ram_cur_addr)，以确定待查找数据的位置、读取次数和长度信息；如图 8 所示；
- 4) 按照待查找数据的位置、读取次数和长度信息从查找窗中读回数据；
- 5) 对读回的数据进行输出预处理：进行裁剪、拼接，最终得到解压后的数据；
- 6) 重复步骤 1) -5)，直至所有待解压数据解压完成。

实施例 4、

如实施例 3 所述解码电路的工作方法还包括：

将所述 literal 数据和所述解压后的数据送入所述输出缓存，对数据进行计数，当数据量到达预设阈值后，输出缓存将对应数据写入 RAM 进行保存，同时对 RAM 进行更新；

进行解压时单个所匹配的字符串位置存在以下三种情形：

仅存在于 RAM 中；

或仅存在于输出缓存中；

或同时存在于 RAM 和输出缓存中。如图 7-1、7-2、7-3 所示。

将所述 literal 数据和所述解压后的数据送入所述输出缓存，当一次待解压数据全部处理完成时，将输出缓存中的数据输出。

对 RAM 进行更新是滚动更新，如图 4、5 所示，包括：

将输出缓存中的数据写入 RAM 时,按照由低地址向高地址,或者由高地址向低地址的方向将每一笔数据依次写入。以 4K*8Byte 的 RAM,按照由高地址向低地址的方向写入时举例说明,用位宽为 12bit 的 ram_cur_addr 代表下一笔数据要写入的地址,解压开始时,ram_cur_addr 应该指向最高地址 4095,每写入一笔 8Byte 的数据后,地址减 1,当 ram_cur_addr 减到 0 时,由于二进制的计数特性,ram_cur_addr 再减 1 回到 4095,这样就实现了对 RAM 的滚动更新。同理,也可以由低地址 0 开始写,不过每次写入一次数据后,ram_cur_addr 应该加 1,也可以实现对 RAM 滚动更新的效果。

所述步骤 5) 输出预处理包括:

对数据进行裁剪,裁剪掉冗余数据;

对数据进行拼接,将裁剪后的数据和后续数据重新拼接成对齐的格式。

由于 pair 数据所代表的字符串是不定长的, , 比如在 GZIP 算法中, length 的范围为 3 到 258Byte, 而每次通过 RAM 从查找窗中读回的数据都是该地址全部的数据,所以需要裁剪掉冗余数据。如图 9, 如果 pair 数据所匹配到的数据只有 c、d、e、f, 这 4 个 Byte, 但是对 RAM 进行读操作的时候是把这个地址所有的 Byte 的 a、b、c、d、e、f、g、h 都读出来的,所以需要把除有效数据 c、d、e、f 外的数据裁剪掉。裁剪后的数据拼接在输出缓存的高位,输出缓存中已经有 x、y、z 3 个 Byte, 所以拼接后的输出缓存为 x、y、z、c、d、e、f。以进行 RAM 的更新和数据的输出,这时对单个 pair 数据的解压操作就已经完成。

实施例 5、

根据实施例 3 所述的步骤 2) 地址映射电路的映射规则包括:

1) 地址映射电路对 pair 数据中的 distance 的大小进行判断:

1-1) 若 distance 小于等于 out_buf_cnt 时,则直接从输出缓存中拷贝 length 个数据;

1-2) 若 distance 大于 out_buf_cnt 时,则从 RAM 中拷贝数据,并对 length 的大小进行判断:

1-2-1) 若 $length \leq distance - out_buf_cnt$, 则待查找的数据只存在于 RAM 中,

起始地址 (ram_rd_st_addr) = ram_cur_addr - $\lceil (distance - out_buf_cnt) / RAM \text{ 每个地址对应的字节数} \rceil$ 向上取整, 读取次数 (rd_cnt):

若 $length \leq \lceil (distance - out_buf_cnt) / RAM \text{ 每个地址对应的字节数} \rceil$ 取余, 只进行 1 次读 RAM 操作, rd_cnt=1;

若 $length > \lceil (distance - out_buf_cnt) / RAM \text{ 每个地址对应的字节数} \rceil$ 取余, $rd_cnt = \lceil [length - \lceil (distance - out_buf_cnt) / RAM \text{ 每个地址对应的字节数} \rceil \text{ 取余}] / RAM \text{ 每个地址对应的字节数} \rceil$ 向上取整;

读取过程为,从起始地址开始对 RAM 进行读操作,每读一次,ram_rd_st_addr 减 1,再对新的 ram_rd_st_addr 进行读操作,直至读够 rd_cnt 次,对 RAM 的第一次读操作和最后一次读操作所读出的数据可能有冗余;

1-2-2) 若 $\text{length} > \text{distance} - \text{out_buf_cnt}$ 时, 则从 RAM 中查找 $\text{distance} - \text{out_buf_cnt}$ 个数据, 剩下的 $\text{length} - \text{distance} - \text{out_buf_cnt}$ 个数据从输出缓存中获得。

以上为地址映射规则, 此外从 RAM 中读出的数据还需要经过输出预处理电路的裁剪、拼接才能最终进入到输出缓存中。

本发明中提出的地址映射电路及映射规则, 使原本 pair 数据中的 length 和 distance 可以适配本发明所述的查找窗结构, 同时通过输出预处理电路对这种查找窗结构带来的冗余信息进行裁剪和拼接, 从而快速得到 pair 所匹配的字符串。

如果 RAM 的数据宽度为 1Byte, 那么如果处理 length 长度的 pair 数据则至少需要 length 个时钟周期, 因为即使是采用了流水线结构的电路, 每次读操作也至少要消耗一个时钟周期, 这样就限制了解码的速度上限会小于 1Byte/cycle。本发明适当增大了 RAM 每个地址对应的数据宽度, 以提高解码速度上限, 同时提出了相应的查 RAM 更新方法来提高解码速度。例如: 如果 RAM 的大小为 32KByte (4K*8Byte)。由于 RAM 每个地址的数据位宽为 8Byte, 如果每解压出 1 个 Byte 就对 RAM 更新一次, 则必须在写该地址之前进行一次读操作。例如连续解压出 8 个 literal, 如 “abcdefgh” 这个序列, 更新 RAM 的过程会按照以下顺序进行 写入 a→读出 a→写入 ab→读出 ab→写入 abc→读出 abc→写入 abcd→读出 abcd→写入 abcde……→读出 abcdefg→写入 abcdefgh, 完成这样的操作共需要 15 个时钟周期, 时间成本较高。

本发明提出的“双口 RAM+输出缓存”结构, 如果解压后的数据先放入输出缓存并进行计数 (out_buf_cnt), 输出缓存大小为 8Byte, 当数据缓存够 8Byte 后再进行 RAM 的更新操作, 见图 5, 输出缓存状态即按照以下顺序变化 a→ab→abc→abcd→abcde→abcdef→abcdefg→abcdefgh→写入 abcdefgh, 如图 4, 仅花费 9 个时钟周期, 极大的提高了解码效率。

同时将这一笔解压后的数据进行输出, 清空输出缓存。通过上述说明, 可以知道查找窗包含了两个部分, 一部分是已经写入 RAM 的数据, 另一部分是未写入 RAM 的数据: 输出缓存中的数据就是未写入 RAM 的数据, 这是由于设定的阈值规定的。

1、一种 LZ77 解码电路，其特征在于，其中待解压数据为可被恢复为包含 literal 和 pair 的数据；

按照 pair 数据流方向，所述 LZ77 解码电路包括：

输入/输出控制电路、输入缓存、输入预处理电路、地址映射电路、RAM 输入/输出控制电路、RAM、输出预处理电路、输出缓存，所述 RAM 和输出缓存共同构成了查找窗；其中，通过所述 RAM 输入/输出控制电路向查找窗的 RAM 查找 pair 需要的数据；

按照 literal 数据流方向，所述 LZ77 解码电路包括：

输入缓存、输入预处理电路和输出缓存；

输出缓存中的数据更新所述 RAM；

其中，所述输入/输出控制电路用于控制待解压数据的输入和解压后的数据输出；

所述输入缓存，用于缓存通过输入/输出控制电路输入的数据；

所述输入预处理电路，用于将待解压数据预处理为 literal 数据和/或 pair 数据；

所述地址映射电路，用于对 pair 数据进行解析：将所述 pair 数据中的 distance 映射为 RAM 可识别的地址 (ram_rd_st_addr)，根据所述 pair 数据中的 length 计算出需要去 RAM 中读取数据的次数 (ram_rd_cnt)，根据输出缓存中数据的个数 (out_buf_cnt)，根据 RAM 当前指向的地址 (ram_cur_addr)，以确定待查找数据的位置、读取次数和长度信息；

所述输出预处理电路，根据数据的位置、读取次数和长度信息从 RAM 读回的数据，并对数据进行裁剪、拼接，最终形成解压后的数据；

所述输出缓存，用于缓存解压后的数据；

所述 RAM 输入/输出控制电路，用于控制 RAM 内的数据写入和读出；

所述查找窗，包括：RAM 和输出缓存，所述输出缓存对 RAM 进行更新。

2、根据权利要求 1 所述的一种 LZ77 解码电路，其特征在于，所述查找窗包括双口 RAM 和输出缓存；所述查找窗的最小架构中双口 RAM 的与使用 LZ77 压缩时的“字典”的大小一致；所述输出缓存本身的大小大于等于 RAM 每个地址对应的数据宽度。

3、如权利要求 1 或 2 所述解码电路的工作方法，其特征在于，包括：

1) 解码电路读入一笔待解压数据放至输入缓存；

2) 将待解压数据预处理为 literal 数据和/或 pair 数据，并将 literal 数据送入输出缓存、将 pair 数据送入地址映射电路；

3) 将 pair 数据中的 distance 映射为查找窗可识别的地址 (ram_rd_st_addr)，根据 pair 数据中的 length 计算出需要去查找窗中读取数据的次数 (ram_rd_cnt)，根据输出缓存中数据的个数 (out_buf_cnt)，根据 RAM 当前指向的地址 (ram_cur_addr)，以确定待查找数据的位置、读取次数和长度信息；

4) 按照待查找数据的位置、读取次数和长度信息从查找窗中读回数据；

5) 对读回的数据进行输出预处理：进行裁剪、拼接，最终得到解压后的数据；

6) 重复步骤 1) -5)，直至所有待解压数据解压完成。

4、如权利要求3所述解码电路的工作方法，其特征在于，所述工作方法还包括：

将所述 literal 数据和所述解压后的数据送入所述输出缓存，对数据进行计数，当数据量到达预设阈值后，输出缓存将对应数据写入 RAM 进行保存，同时对 RAM 进行更新；

进行解压时单个所匹配的字符串位置存在以下三种情形：

仅存在于 RAM 中；

或仅存在于输出缓存中；

或同时存在于 RAM 和输出缓存中。

5、如权利要求3所述解码电路的工作方法，其特征在于，

将所述 literal 数据和所述解压后的数据送入所述输出缓存，当一次待解压数据全部处理完成时，将输出缓存中的数据输出。

6、如权利要求4或5所述解码电路的工作方法，其特征在于，对 RAM 进行更新是滚动更新，包括：

将输出缓存中的数据写入 RAM 时，按照由低地址向高地址，或者由高地址向低地址的方向将每一笔数据依次写入。

7、如权利要求3所述解码电路的工作方法，其特征在于，所述步骤5)输出预处理包括：

对数据进行裁剪，裁剪掉冗余数据；

对数据进行拼接，将裁剪后的数据和后续数据重新拼接成对齐的格式。

8、如权利要求3所述解码电路的工作方法，其特征在于，所述步骤2)地址映射电路的映射规则包括：

1) 地址映射电路对 pair 数据中的 distance 的大小进行判断：

1-1) 若 distance 小于等于 out_buf_cnt 时，则直接从输出缓存中拷贝 length 个数据；

1-2) 若 distance 大于 out_buf_cnt 时，则从 RAM 中拷贝数据，并对 length 的大小进行判断：

1-2-1) 若 $\text{length} \leq \text{distance} - \text{out_buf_cnt}$ ，则待查找的数据只存在于 RAM 中，

起始地址 (ram_rd_st_addr) = $\text{ram_cur_addr} - [(\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数}]_{\text{向上取整}}$ ，读取次数 (rd_cnt)：

若 $\text{length} \leq [(\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数}]_{\text{取余}}$ ，只进行 1 次读 RAM 操作， $\text{rd_cnt} = 1$ ；

若 $\text{length} > [(\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数}]_{\text{取余}}$ ，
 $\text{rd_cnt} = [[\text{length} - [(\text{distance} - \text{out_buf_cnt}) / \text{RAM 每个地址对应的字节数}]_{\text{取余}}] / \text{RAM 每个地址对应的字节数}]_{\text{向上取整}}$ ；

读取过程为,从起始地址开始对 RAM 进行读操作,每读一次,ram_rd_st_addr 减 1,再对新的 ram_rd_st_addr 进行读操作,直至读够 rd_cnt 次;

1-2-2) 若 $\text{length} > \text{distance} - \text{out_buf_cnt}$ 时,则从 RAM 中查找 $\text{distance} - \text{out_buf_cnt}$ 个数据,剩下的 $\text{length} - \text{distance} - \text{out_buf_cnt}$ 个数据从输出缓存中获得。