# 一种用于HEVC标准中帧内预测的预测单元的硬件编址寻址方法

申请号：201410590929.4
申请日：2014-10-29

申请(专利权)人　复旦大学
地址　200433 上海市杨浦区邯郸路220号
发明(设计)人　范益波　黄磊磊　刘聪　金怡泽　曾晓洋

主分类号　H04N19/593(2014.01)I
分类号　H04N19/593(2014.01)I　H04N19/105(2014.01)I　H04N19/129(2014.01)I
公开(公告)号　104363458A
公开(公告)日　2015-02-18
专利代理机构　上海正旦专利代理有限公司 31200
代理人　陆飞　盛志范

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104363458 A
(43)申请公布日 2015.02.18

(21)申请号 201410590929.4

(22)申请日 2014.10.29

(71)申请人 复旦大学
地址 200433 上海市杨浦区邯郸路 220 号

(72)发明人 范益波 黄磊磊 刘聪 金怡泽 曾晓洋

(74)专利代理机构 上海正旦专利代理有限公司 31200
代理人 陆飞 盛志范

(51)Int.Cl.
*H04N 19/593*(2014.01)
*H04N 19/105*(2014.01)
*H04N 19/129*(2014.01)

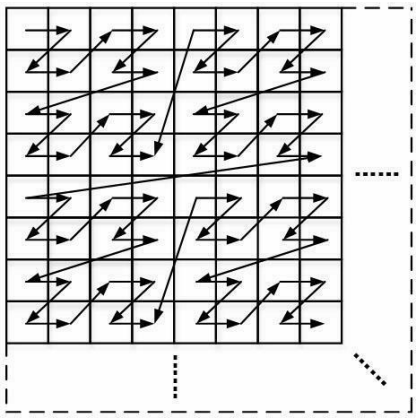
权利要求书2页 说明书5页 附图2页

(54)发明名称

一种用于 HEVC 标准中帧内预测的预测单元的硬件编址寻址方法

(57)摘要

本发明属于高清数字视频压缩编解码技术领域，具体为一种用于 HEVC 标准中帧内预测的预测单元的硬件编址寻址方法。在 HEVC 帧内编码的过程中需要依据当前的搜索模式，对预测单元进行不同顺序的寻址；假设当前编码的最大单位为一个 64×64 块，那么对于当前的处理单位，其中所有 4×4 大小的预测单元按照 Z 字顺序编址；对于其他预测单元，将以其左上角的 4×4 块表征其地址；在此编址基础上，对于不同顺序搜索，给出不同的寻址的公式。本发明以较低的成本完成对于预测单元编址寻址的硬件实现。



CN 104363458 A

1. 一种用于 HEVC 标准中帧内预测预测单元的硬件编址寻址方法，在 HEVC 帧内编码的过程中需要依据当前的搜索模式，对预测单元（PU）进行不同顺序的寻址；其特征在于：

假设当前编码的最大单位（LCU）为一个 64×64 块，那么对于当前的处理单位，其中所有 4×4 大小的预测单元（PU）按照 Z 字即 Zig-Zag 顺序编址；对于其他预测单元（PU），将以其左上角的 4×4 块表征其地址；

在此编址基础上，如果帧内预测按照前序遍历的顺序搜索，那么对于下一块预测单元（PU）的寻址的公式为：

$$Addr_{nxt} = \begin{cases} Addr_{cur} & , Size_{nxt} < Size_{cur} \\ Addr_{cur} + 1 & , Size_{nxt} >= Size_{cur} \end{cases} \quad (1)$$

此处，$Addr_{nxt}$ 是下一个 PU 的地址，$Addr_{cur}$ 是当前 PU 的地址，$Size_{nxt}$ 是下一个的 PU 的大小，$Size_{cur}$ 是当前的 PU 的大小，$Addr_{nxt}$ 和 $Addr_{cur}$ 的单位是 4×4 大小的预测单元（PU），$Size_{nxt}$ 和 $Size_{cur}$ 取的是预测单元（PU）的边长，单位是像素；

如果帧内预测按照后序遍历的顺序搜索，那么对于下一块预测单元（PU）的寻址的公式为：

$$Addr_{nxt} = \begin{cases} Addr_{cur} + Offset & , Size_{nxt} == Size_{cur} \\ Addr_{cur} - 3 * Offset & , Size_{nxt} != Size_{cur} \end{cases} \quad (2)$$

$$Offset = \begin{cases} 1 & , Size_{cur} == 4 \\ 4 & , Size_{cur} == 8 \\ 16 & , Size_{cur} == 16 \\ 64 & , Size_{cur} == 32 \end{cases}$$

此处，$Addr_{nxt}$ 是下一个 PU 的地址，$Addr_{cur}$ 是当前 PU 的地址，$Offset$ 是需要加上的偏移地址，$Size_{nxt}$ 是下一个的 PU 的大小，$Size_{cur}$ 是当前的 PU 的大小，$Addr_{nxt}$、$Addr_{cur}$ 和 $Offset$ 的单位是 4×4 大小的预测单元（PU），$Size_{nxt}$ 和 $Size_{cur}$ 取的是预测单元（PU）的边长，单位是像素；

如果帧内预测按照给定分块（Partition）的顺序搜索，那么对于下一块预测单元（PU）的寻址的公式为：

$$Addr_{nxt} = Addr_{cur} + Offset \quad (3)$$

$$Offset = \begin{cases} 1 & , Size_{cur} == 4 \\ 4 & , Size_{cur} == 8 \\ 16 & , Size_{cur} == 16 \\ 64 & , Size_{cur} == 32 \end{cases}$$

此处，$Addr_{nxt}$ 是下一个 PU 的地址，$Addr_{cur}$ 是当前 PU 的地址，$Offset$ 是需要加上的偏移地址， $Size_{cur}$ 是当前的 PU 的大小 $Addr_{nxt}$、$Addr_{cur}$ 和 $Offset$ 的单位是 4×4 大小的预测单元（PU），$Size_{cur}$ 取的是预测单元（PU）的边长，单位是像素。

# 一种用于 HEVC 标准中帧内预测的预测单元的硬件编址寻址方法

## 技术领域

[0001] 本发明属于高清数字视频压缩编解码技术领域，针对 HEVC 视频编解码标准，具体涉及一种用于 HEVC 标准中帧内预测的预测单元的硬件编址寻址方法。

## 背景技术

[0002] HEVC（High Efficiency Video Coding）是由国际电信组织（ITU）和运动图像专家组（MPEG）联合成立的组织 JCTVC 提出的下一代视频编解码标准。目标是在相同的视觉效果的前提下，相比于上一代标准，即 H.264/AVC 标准，压缩率提高一倍。

[0003] 基于 HEVC 的视频编码器，主要由以下模块组成：帧内预测、帧间预测、变换、量化、反量化、反变换、重建、去方块滤波器、自适应样点补偿等。其中，帧内预测利用同一帧图像内相邻像素之间的相关性，采用合适的方法进行预测，以减小空间冗余度，从而达到压缩的效果。为了提高预测的准确性，HEVC 引入了基于四叉树的块结构，具体地，图像处理块的最大单位（LCU）可以是一个 64×64 块，而该 64×64 块可以被划分成 4 个 32×32 块，每个 32×32 块又可以被划分为 4 个 16×16 块，依次类推直到 4×4 块的层次，这些块被统一地称为预测单元（PU）。可以说，帧内预测的过程就是搜索预测单元（PU）的过程，而在这样的搜索过程中，对于当前预测单元（PU）的寻址是必不可少的，寻址的复杂直接影响了编码的效率和性能。

## 发明内容

[0004] 本发明的目的在于提出一种可以克服现有技术不足的、高效的、用于 HEVC 标准中帧内预测的预测单元的硬件编址寻址方法。

[0005] 假设当前编码的最大单位（LCU）为一个 64×64 块，那么对于当前的处理单位，其中所有 4×4 大小的预测单元（PU）将按照 Z 字（Zig-Zag）顺序编址，如图 1 所示。该顺序实际上也是 HEVC 标准中所采用的处理顺序，如图 2 所示。

[0006] 而对于其他预测单元（PU），将以其左上角的 4×4 块表征其地址，8×8 的情况如图 3 所示，16×16 的情况如图 4 所示。

[0007] 在此编址基础上，如果帧内预测按照前序遍历的顺序搜索，如图 5 所示，其中，圆圈内包含的数字表征了搜索顺序，那么对于下一块预测单元（PU）的寻址可以利用公式（1）完成：

$$Addr_{nxt} = \begin{cases} Addr_{cur} & , Size_{nxt} < Size_{cur} \\ Addr_{cur} + 1 & , Size_{nxt} >= Size_{cur} \end{cases} \quad (1)$$

此处，$Addr_{nxt}$ 是下一个 PU 的地址，$Addr_{cur}$ 是当前 PU 的地址，$Size_{nxt}$ 是下一个的 PU 的大小，$Size_{cur}$ 是当前的 PU 的大小，$Addr_{nxt}$ 和 $Addr_{cur}$ 的单位是 4×4 大小的预测单元（PU），

$Size_{nxt}$ 和 $Size_{cur}$ 取的是预测单元(PU)的边长,单位是像素。

[0008] 如果帧内预测按照后序遍历的顺序搜索,如图 6 所示,其中,圆圈内包含的的数字表征了搜索顺序,那么对于下一块预测单元(PU)的寻址可以利用公式(2)完成:

$$Addr_{nxt} = \begin{cases} Addr_{cur} + Offset & , Size_{nxt} == Size_{cur} \\ Addr_{cur} - 3 * Offset & , Size_{nxt} != Size_{cur} \end{cases} \quad (2)$$

$$Offset = \begin{cases} 1 & , Size_{cur} == 4 \\ 4 & , Size_{cur} == 8 \\ 16 & , Size_{cur} == 16 \\ 64 & , Size_{cur} == 32 \end{cases}$$

此处,$Addr_{nxt}$ 是下一个 PU 的地址,$Addr_{cur}$ 是当前 PU 的地址,$Offset$ 是需要加上的偏移地址,$Size_{nxt}$ 是下一个的 PU 的大小,$Size_{cur}$ 是当前的 PU 的大小,$Addr_{nxt}$、$Addr_{cur}$ 和 $Offset$ 的单位是 4×4 大小的预测单元(PU),$Size_{nxt}$ 和 $Size_{cur}$ 取的是预测单元(PU)的边长,单位是像素。

[0009] 如果帧内预测按照给定分块(Partition)的顺序搜索,那么对于下一块预测单元(PU)的寻址可以利用公式(3)完成:

$$Addr_{nxt} = Addr_{cur} + Offset \quad (3)$$

$$Offset = \begin{cases} 1 & , Size_{cur} == 4 \\ 4 & , Size_{cur} == 8 \\ 16 & , Size_{cur} == 16 \\ 64 & , Size_{cur} == 32 \end{cases}$$

此处,$Addr_{nxt}$ 是下一个 PU 的地址,$Addr_{cur}$ 是当前 PU 的地址,$Offset$ 是需要加上的偏移地址, $Size_{cur}$ 是当前的 PU 的大小 $Addr_{nxt}$、$Addr_{cur}$ 和 $Offset$ 的单位是 4×4 大小的预测单元(PU),$Size_{cur}$ 取的是预测单元(PU)的边长,单位是像素。

[0010] 使用本方法,可以有效地结合不同的搜索方式和不同块大小的预测单元(PU),简化对于预测单元(PU)的寻址操作,降低硬件复杂度,提高编码器的性能。

## 附图说明

[0011] 图 1:4×4 大小预测单元(PU)的编址。

[0012] 图 2:Z 字(Zig-Zag)顺序示意。

[0013] 图 3:8×8 大小预测单元(PU)的编址。

[0014] 图 4:16×16 大小预测单元(PU)的编址。

[0015] 图 5 :前序遍历全搜索。

[0016] 图 6 :后序遍历全搜索。

[0017] 图 7 :给定分块(Partition)方式搜索的一个例子。

## 具体实施方式

[0018] 下面通过实例并结合附图,进一步具体描述本发明方法。

[0019] 如对于图 5 所示的前序遍历的搜索顺序,帧内预测将 :

1. 搜索 64×64 块,其地址为 0 ;
2. 搜索该 64×64 块中的第 1 个 32×32 块,其地址为 1 ;
3. 搜索该 32×32 块中的第 1 个 16×16 块,其地址为 1 ;
4. 搜索该 16×16 块中的第 1 个 8×8 块,其地址为 1 ;
5. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 1 ;
6. 搜索该 8×8 块中的 2 个 4×4 块,其地址为 2 (1+1);
7. 搜索该 8×8 块中的 3 个 4×4 块,其地址为 3 (2+1);
8. 搜索该 8×8 块中的 4 个 4×4 块,其地址为 4 (3+1);
9. 搜索上一层 16×16 块中的第 2 个 8×8 块,其地址为 5 (4+1);
10. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 5 ;
11. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 6 (5+1);
12. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 7 (6+1);
13. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 8 (7+1);
14. 搜索上一层 16×16 块中的第 3 个 8×8 块,其地址为 9 (8+1);
15. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 9 ;
16. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 10 (9+1);
17. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 11 (10+1);
18. 搜索该 8×8 块中的 1 个 4×4 块,其地址为 12 (11+1);

……

列出该顺序后,可以得到在下一个预测单元(PU)的大小小于当前预测单元(PU)的大小的情况下,预测单元(PU)的地址不变 ;在下一个预测单元(PU)的大小大于等于当前预测单元(PU)的大小的情况下,预测单元(PU)的地址加 1。此处,地址仍以一个 4×4 大小的预测单元(PU)作为基本单位。

[0020] 又对图 6 所示的后序遍历的搜索顺序,帧内预测将 :

1. 搜索第 1 个 4×4 块,其地址为 1 ;
2. 搜索第 2 个 4×4 块,其地址为 2 (1+1);
3. 搜索第 3 个 4×4 块,其地址为 3 (2+1);
4. 搜索第 4 个 4×4 块,其地址为 4 (3+1);
5. 搜索第 1 个 8×8 块,其地址为 1 (4-3*1);
6. 搜索第 5 个 4×4 块,其地址为 5 (1+4);
7. 搜索第 6 个 4×4 块,其地址为 6 (5+1);
8. 搜索第 7 个 4×4 块,其地址为 7 (6+1);

9. 搜索第 8 个 4×4 块,其地址为 8（7+1）;

10. 搜索第 2 个 8×8 块,其地址为 5（8-3*1）;

11. 搜索第 9 个 4×4 块,其地址为 9（5+4）;

12. 搜索第 10 个 4×4 块,其地址为 10（9+1）;

13. 搜索第 11 个 4×4 块,其地址为 11（10+1）;

14. 搜索第 12 个 4×4 块,其地址为 12（11+1）;

15. 搜索第 3 个 8×8 块,其地址为 9（12-3*1）;

16. 搜索第 13 个 4×4 块,其地址为 13（9+4）;

17. 搜索第 14 个 4×4 块,其地址为 14（13+1）;

18. 搜索第 15 个 4×4 块,其地址为 15（13+1）;

……

列出该顺序后,可以得到在下一个预测单元(PU)的大小大于当前预测单元(PU)的大小的情况下,下一个预测单元(PU)的地址等与当前地址减去之前所述的 *Offset* 变量的 3 倍 ;在下一个预测单元(PU)的大小小于等于当前预测单元(PU)的大小的情况下,下一个预测单元(PU)的地址等于当前地址加上 *Offset* 变量。此处,地址仍以一个 4×4 大小的预测单元(PU)作为基本单位。

[0021] 再如对图 7 所示的基于给定分块方式(Partition)的搜索,在这个示例下,帧内预测将 :

1. 搜索第 1 个 32×32 块,其地址为 1 ;

2. 搜索下一个 8×8 块,其地址为 65（1+64）;

3. 搜索下一个 8×8 块,其地址为 69（65+4）;

4. 搜索下一个 8×8 块,其地址为 73（69+4）;

5. 搜索下一个 8×8 块,其地址为 77（73+4）;

6. 搜索下一个 16×16 块,其地址为 81（77+4）;

7. 搜索下一个 16×16 块,其地址为 97（81+16）;

8. 搜索下一个 16×16 块,其地址为 113（97+16）;

9. 搜索下一个 16×16 块,其地址为 129（113+16）;

10. 搜索下一个 16×16 块,其地址为 145（129+16）;

11. 搜索下一个 8×8 块,其地址为 161（145+16）;

12. 搜索下一个 8×8 块,其地址为 165（161+4）;

13. 搜索下一个 8×8 块,其地址为 169（165+4）;

14. 搜索下一个 8×8 块,其地址为 173（169+4）;

15. 搜索下一个 16×16 块,其地址为 177（173+4）;

16. 搜索下一个 16×16 块,其地址为 193（177+16）;

17. 搜索下一个 16×16 块,其地址为 209（193+16）;

18. 搜索下一个 16×16 块,其地址为 225（209+16）;

19. 搜索下一个 8×8 块,其地址为 241（225+16）;

20. 搜索下一个 8×8 块,其地址为 245（241+4）;

21. 搜索下一个 8×8 块,其地址为 249（245+4）;

22. 搜索下一个 8×8 块,其地址为 253（249+4）;

列出该顺序后,可以得到在下一个预测单元(PU)的大小大于当前预测单元(PU)的大小的情况下,下一个预测单元(PU)的地址永远等于当前地址加上之前所述的 *Offset* 变量。

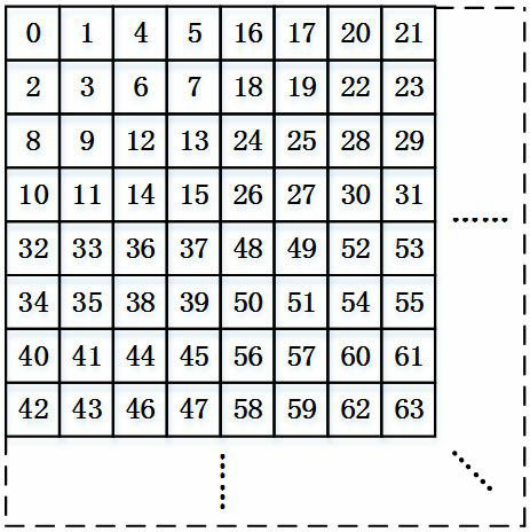

0 1 4 5 16 17 20 21
2 3 6 7 18 19 22 23
8 9 12 13 24 25 28 29
10 11 14 15 26 27 30 31
32 33 36 37 48 49 52 53
34 35 38 39 50 51 54 55
40 41 44 45 56 57 60 61
42 43 46 47 58 59 62 63

图 1

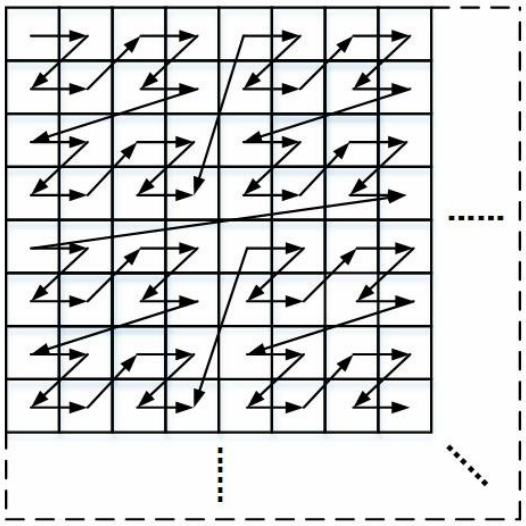


图 2

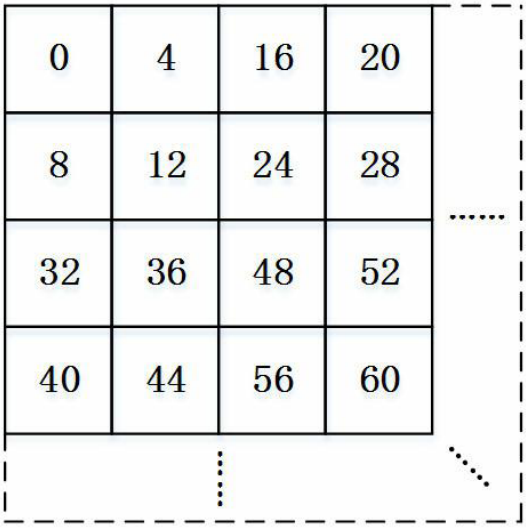

0 4 16 20
8 12 24 28
32 36 48 52
40 44 56 60

图 3

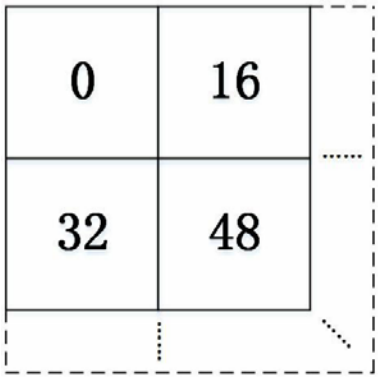

0 16
32 48

图 4

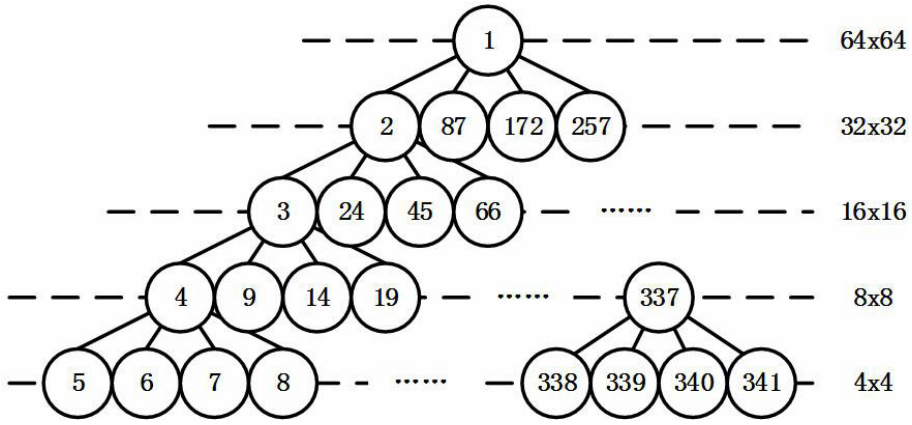

1
64x64
2
87
172
257
32x32
3
24
45
66
16x16
4
9
14
19
337
8x8
5
6
7
8
338
339
340
341
4x4

图 5

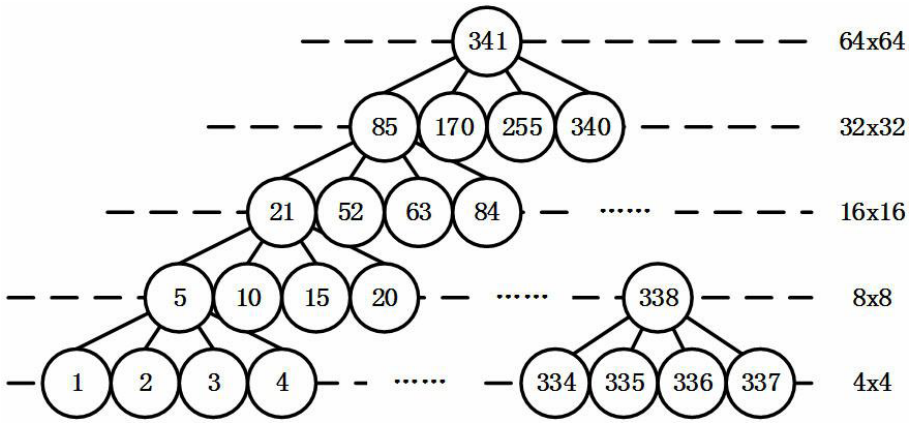

341
64x64
85
170
255
340
32x32
21
52
63
84
16x16
5
10
15
20
338
8x8
1
2
3
4
334
335
336
337
4x4

图 6

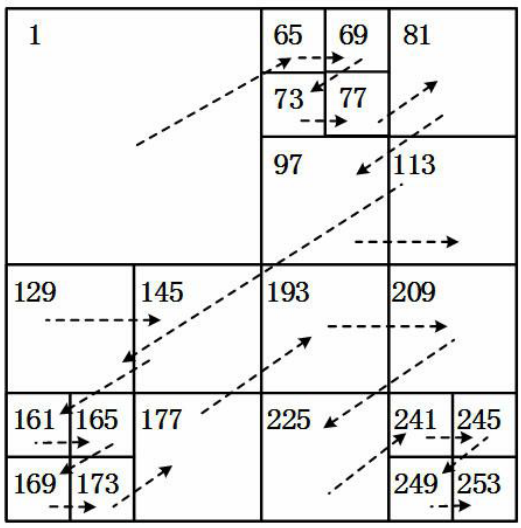

1
65
69
81
73
77
97
113
129
145
193
209
161
165
177
225
241
245
169
173
249
253

图 7