

【HDL系列】Kogge-Stone加法器原理与设计

原创 纸上谈芯 于 2020-02-09 10:04:41 发布 阅读量5.1k 收藏 41 点赞数 5

分类专栏: 纸上谈芯 文章标签: verilog Kogge-Stone加法器 树形加法器 数字IC设计



纸上谈芯 专栏收录该内容

129 订阅 31 篇文章 订阅

目录

- 一、Kogge-Stone并行算法
- 二、Kogge-Stone加法器
- 三、Verilog设计

Kogge-Stone加法器是利用Peter M. Kogge和Harold S.Stone于1972年提出的一种并行算法生成的一种树形加法器。

一、Kogge-Stone并行算法

Kogge和Stone根据一般m阶递归问题提出一种并行算法。本文介绍其一阶递归问题的并行结构，详细请阅读其论文。

对于序列 $X_1, X_2, X_3, \dots, X_n$ ，其中 X_i 是其之前m个数的函数，也就是：

$$x_i = f(x_{i-1}, x_{i-2}, \dots, x_{i-m})$$

以上是一个常见的问题，如线性时变系统，对于i时刻的 X_i ，可由下式计算而来：

$$\begin{aligned} x_1 &= B_1 \\ x_2 &= A_2 x_1 + B_2 \\ x_3 &= A_3 x_2 + B_3 \\ &\dots \dots \dots \\ x_i &= A_i x_{i-1} + B_i \\ &\dots \dots \dots \\ x_N &= A_N x_{N-1} + B_N \end{aligned}$$

其中 A_i 和 B_i 表示系统内部动态系数，可为实数或者复数，常量，时变矩阵等。

使用简化的并行算法解决以下一阶递归问题，给定 $X_1=b_1$ ，则

$$x_i = a_i x_{i-1} + b_i$$

为解决如上问题，论文中定义了下式：

$$\hat{Q}(m, n) = \sum_{j=n}^m \left(\prod_{r=j+1}^m a_r \right) b_j$$

其中对 a_r 系数使用的是连乘符合，之后与 b_j 相乘后使用的是累积和符号。

所以，

觉得还不错? 一键收藏



纸上谈芯

关注

5



41

1



专栏

$$x_1 = b_1 = \hat{Q}(1,1)$$

$$x_2 = a_2 x_1 + b_2 = a_2 b_1 + b_2 = \hat{Q}(2,1)$$

$$x_3 = a_3 x_2 + b_3 = a_3 a_2 b_1 + a_3 b_2 + b_3 = \hat{Q}(3,1)$$

... ..

$$x_i = a_i x_{i-1} + b_i = \hat{Q}(i, 1)$$

... ..

$$x_N = a_N x_{N-1} + b_N = \hat{Q}(N, 1)$$

上式又可以重新写成如下形式：

$$\hat{Q}(1,1) = x_1 = b_1$$

$$\hat{Q}(2,1) = x_2 = a_2 x_1 + b_2 = a_2 \hat{Q}(1,1) + b_2$$

... ..

$$\hat{Q}(4,1) = x_4 = a_4 x_3 + b_4$$

$$= a_4 a_3 x_2 + a_4 b_3 + b_4$$

$$= a_4 a_3 \hat{Q}(2,1) + \hat{Q}(4,3)$$

... .. <https://blog.csdn.net/zhouxuanyuyue>

其一般形式为：

$$\hat{Q}(2i, 1) = x_{2i} = \left(\prod_{r=i+1}^{2i} a_r \right) \hat{Q}(i, 1) + \hat{Q}(2i, i+1)$$

$Q(i,1)$ 和 $Q(2i,i+1)$ 可以独立计算， $Q(i,1)$ 和 $Q(2i,i+1)$ 又可以单独分解独立计算，每个层级都可以独立计算，因此加快了运算速度。以下是x8的计算图例。

觉得还不错? 一键收藏



纸上谈芯

关注

5



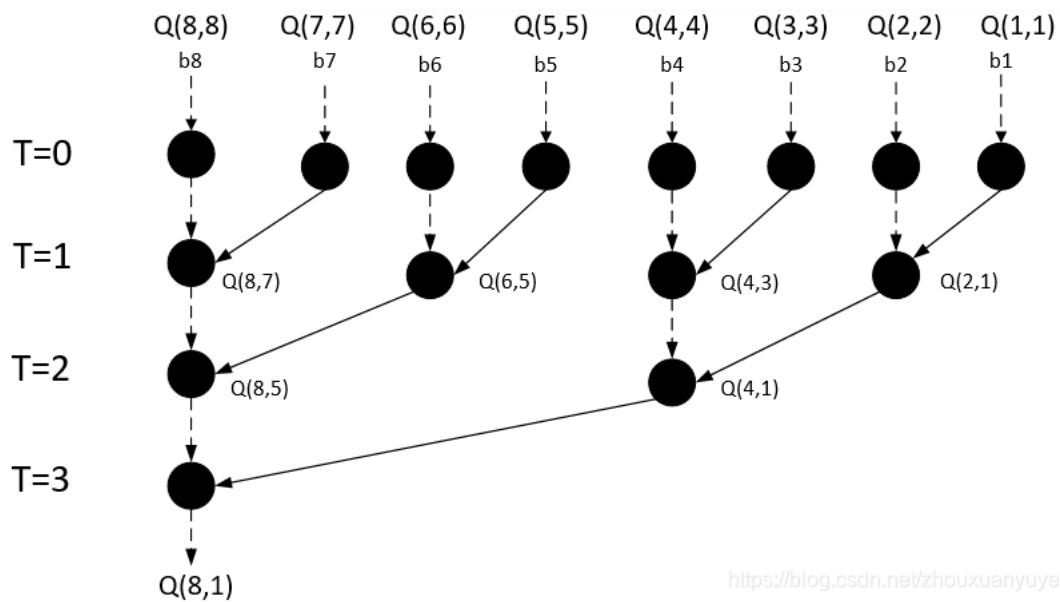
41



1



专栏



x8计算示意图

TABLE I

| Processor | $T = 0$ | | $T = 1$ | | $T = 2$ | | $T = 3$ | |
|-----------|---------|-----------------------------|-----------|---------------------------------------|-------------------|---|-----------------------|--------|
| | $A(i)$ | $B(i)$ | $A(i)$ | $B(i)$ | $A(i)$ | $B(i)$ | $A(i)$ | $B(i)$ |
| 1 | ** | $b_1 = X_1 = \hat{Q}(1, 1)$ | ** | $X_1 = \hat{Q}(1, 1)$ | ** | $X_1 = \hat{Q}(1, 1)$ | ** | X_1 |
| 2 | a_2 | $b_2 = \hat{Q}(2, 2)$ | a_2 | $a_2 x_1 + b_2 = x_2 = \hat{Q}(2, 1)$ | a_2 | $X_2 = \hat{Q}(2, 1)$ | a_2 | X_2 |
| 3 | a_3 | $b_3 = \hat{Q}(3, 3)$ | $a_3 a_2$ | $a_3 b_2 + b_3 = \hat{Q}(3, 2)$ | $a_3 a_2$ | $(a_3 a_2) X_1 + (a_3 b_2 + b_3) = X_3 = \hat{Q}(3, 1)$ | $a_3 a_2$ | X_3 |
| 4 | a_4 | $b_4 = \hat{Q}(4, 4)$ | $a_4 a_3$ | $a_4 b_3 + b_4 = \hat{Q}(4, 3)$ | $a_4 a_3 a_2$ | $(a_4 a_3) X_2 + (a_4 b_3 + b_4) = X_4 = \hat{Q}(4, 1)$ | $a_4 a_3 a_2$ | X_4 |
| 5 | a_5 | $b_5 = \hat{Q}(5, 5)$ | $a_5 a_4$ | $a_5 b_4 + b_5 = \hat{Q}(5, 4)$ | $a_5 a_4 a_3 a_2$ | $a_5 a_4 (a_3 b_2 + b_3) + a_5 b_4 + b_5 = \hat{Q}(5, 2)$ | $\prod_{m=2}^5 a_m^*$ | X_5 |
| 6 | a_6 | $b_6 = \hat{Q}(6, 6)$ | $a_6 a_5$ | $a_6 b_5 + b_6 = \hat{Q}(6, 5)$ | $a_6 a_5 a_4 a_3$ | $\sum_{w=3}^6 \left(\prod_{m=w+1}^6 a_m \right) b_w = \hat{Q}(6, 3)$ | $\prod_{m=2}^6 a_m^*$ | X_6 |
| 7 | a_7 | $b_7 = \hat{Q}(7, 7)$ | $a_7 a_6$ | $a_7 b_6 + b_7 = \hat{Q}(7, 6)$ | $a_7 a_6 a_5 a_4$ | $\sum_{w=4}^7 \left(\prod_{m=w+1}^7 a_m \right) b_w = \hat{Q}(7, 4)$ | $\prod_{m=2}^7 a_m^*$ | X_7 |
| 8 | a_8 | $b_8 = \hat{Q}(8, 8)$ | $a_8 a_7$ | $a_8 b_7 + b_8 = \hat{Q}(8, 7)$ | $a_8 a_7 a_6 a_5$ | $\sum_{w=5}^8 \left(\prod_{m=w+1}^8 a_m \right) b_w = \hat{Q}(8, 5)$ | $\prod_{m=2}^8 a_m^*$ | X_8 |

X1至X8的计算步骤、系数与公式，来源Kogge-Stone论文

上图为x1-x8的计算步骤，系数和公式，分三个时间步骤完成：

T=0: 输入系数b1-b7

T=1: 根据其对应的a1-a8系数计算相邻两项

T=2: 再次根据算式合并Q值

根据以上x1至x8的计算方式，画出其图例如下，如果看过本号之前的文章，相信你对此结构已经非常熟悉：

觉得还不错？ [一键收藏](#)



纸上谈芯

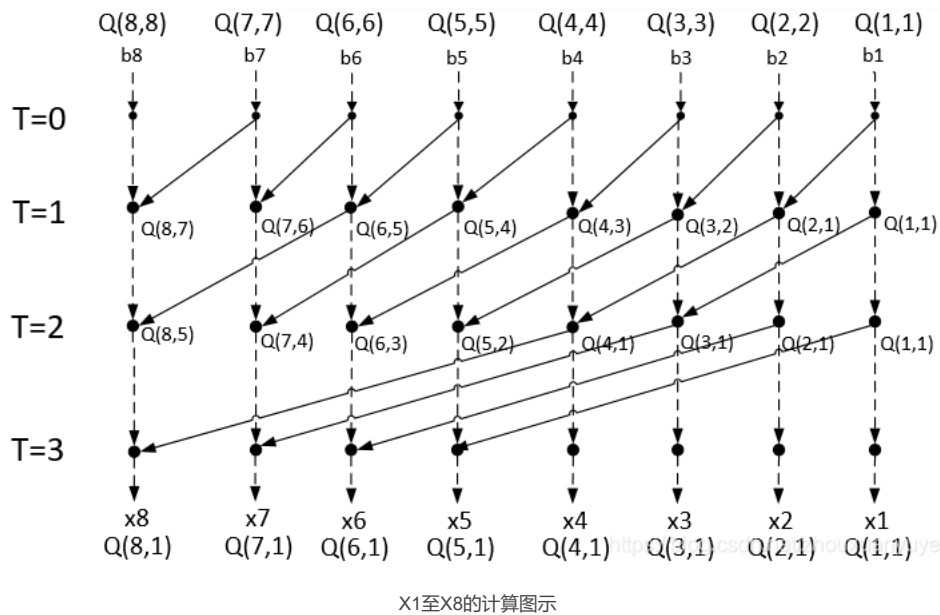
关注

5

41

1

专栏



二、Kogge-Stone加法器

基于以上介绍的Kogge-Stone的一阶递归问题的并行结构，如何应用到加法器中呢？我们先回顾下超前进位加法器中进位链的计算。

对于N比特的A和B两数，结果为N比特S和1比特进位Cout，超前进位加法器的进位链与和公式的计算公式如下：

$$\begin{cases} S_i = p_i \oplus c_{i-1}, & i = 1, \dots, N \\ c_i = g_i + c_{i-1}p_i, & i = 1, \dots, N \\ c_0 = c_{in} \\ c_{out} = c_N \end{cases}$$

其中：

$$\begin{cases} p_i = A_i \oplus B_i, & i = 1, 2, \dots, N \\ g_i = A_i B_i, & i = 1, 2, \dots, N \end{cases}$$

重点关注进位链ci的生成算式，发现其属于Kogge-Stone概括的一阶递归问题：

$$c_i = p_i c_{i-1} + g_i, \quad i = 1, \dots, N$$

所以ci算式写成Kogge-Stone形式：

$$\hat{Q}(i, 1) = p_i c_{i-1} + g_i, \quad i = 1, \dots, N$$

所以，如对于C1-C4等的生成，其算式如下：

觉得还不错？ [一键收藏](#)



纸上谈芯

关注



5



41



1



专栏

$$\hat{Q}(1,1) = c_1$$

$$\hat{Q}(2,1) = c_2 = p_2 c_1 + g_2 = p_2 \hat{Q}(1,1) + g_2$$

$$\hat{Q}(3,1) = c_3 = p_3 c_2 + g_3$$

$$= p_3 p_2 c_1 + p_3 g_2 + g_3$$

$$= p_3 p_2 \hat{Q}(1,1) + \hat{Q}(3,2)$$

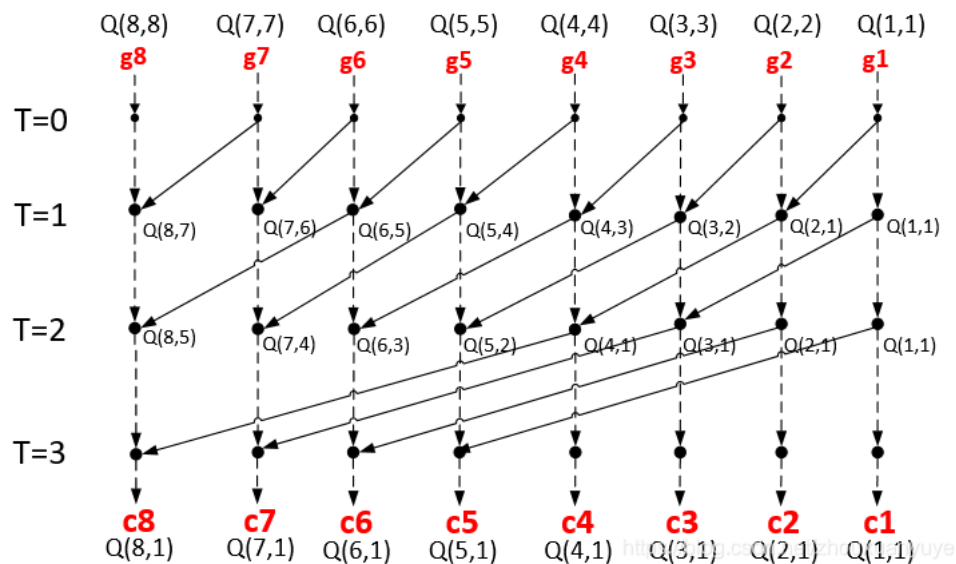
$$\hat{Q}(4,1) = c_4 = p_4 c_3 + g_4$$

$$= p_4 p_3 c_2 + p_4 g_3 + g_4$$

$$= p_4 p_3 \hat{Q}(2,1) + \hat{Q}(4,3)$$

... .. <https://blog.csdn.net/zhouxuanyu>

根据以上原理，8比特的加法器进位链并行计算的Kogge-Stone形式的树形结构如下图所示：



8位加法器进位链Ci生成树形图

三、Verilog设计

设计一个16比特的Kogge-Stone加法器，分三部分：

- (1) 生成g, p信号，由半加器模块组成；
- (2) 每个节点处理单元实现，即每个节点中的系数生成和每个独立的Q值计算；
- (3) 生成输出进位Cout和S信号。

该16比特Kogge-Stone位加法器进位链的树形结构如下图所示：

觉得还不错？ [一键收藏](#)



纸上谈芯

关注



5



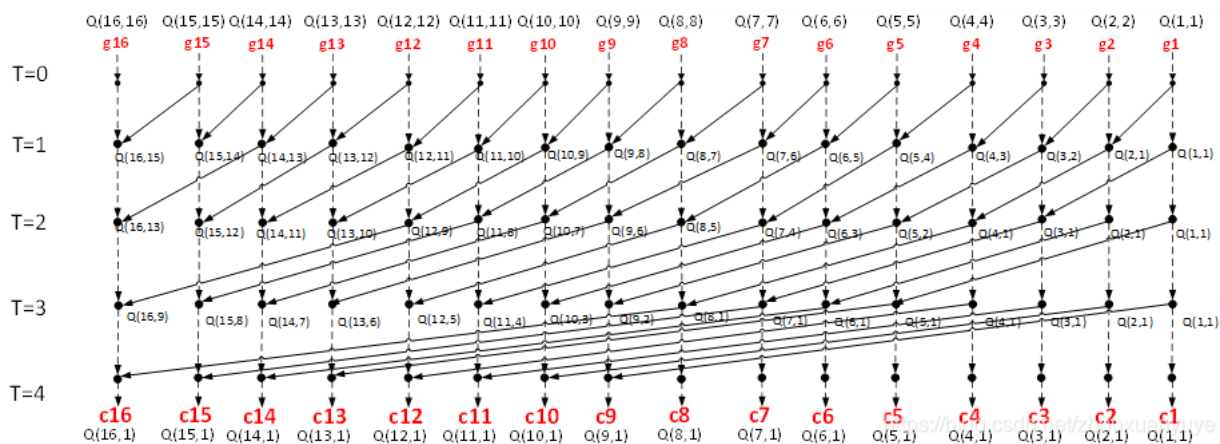
41



1



专栏



16位加法器进位链生成树形图

从图中足以见Kogge-Stone结构加法器布线尤其拥塞，但其好处在于进位链生成的延迟单位少。在 `verilog` 中根据其规律使用generate和endgenerate语句各级信号如下。

```
20 //pg generator
21 genvar i;
22 generate
23     for(i=1; i<=16; i=i+1)begin
24         pg_gen u_pg_gen(
25             .a( A[i] ),
26             .b( B[i] ),
27             .g( g[i] ),
28             .p( p[i] )
29         );
30     end
31 endgenerate
```

16比特Kogge-Stone加法器PG生成逻辑

```
35 //level1
36 //gen coe
37 assign level1_p = p[16:1];
38
39 //gen Q value
40 generate
41     for(i=1; i<=16; i=i+1) begin
42         assign level1_Q[i] = (level1_p[i] & g[i-1]) | g[i];
43     end
44 endgenerate
```

16比特Kogge-Stone加法器level1生成逻辑

```
46 //level2
47 //gen coe
48 generate
49     for(i=1; i<=16; i=i+1) begin
50         assign level2_p[i] = level1_p[i] & level1_p[i-1];
51     end
52 endgenerate
53
54 //gen Q value
55 generate
56     for(i=1; i<=2; i=i+1)
57         assign level2_Q[i] = level1_Q[i];
58
59     for(i=3; i<=16; i=i+1) begin
60         assign level2_Q[i] = (level2_p[i] & level1_Q[i-2]) | level1_Q[i];
61     end
62 endgenerate
```

16比特Kogge-Stone加法器level2生成逻辑

觉得还不错? 一键收藏



纸上谈芯

关注



5



41



1



专栏

```

64 //level3
65 //gen coe
66 generate
67     for(i=1;i<=16;i=i+1) begin
68         assign level3_p[i] = level2_p[i] & level2_p[i-2];
69     end
70 endgenerate
71
72 //gen Q value
73 generate
74     for(i=1;i<=4;i=i+1)
75         assign level3_Q[i] = level2_Q[i];
76     for(i=5;i<=16;i=i+1) begin
77         assign level3_Q[i] = (level3_p[i] & level2_Q[i-4]) | level2_Q[i];
78     end
79 endgenerate

```

16比特Kogge-Stone加法器level3生成逻辑

```

81 //level4
82 //gen coe
83 generate
84     for(i=1;i<=16;i=i+1) begin
85         assign level4_p[i] = level3_p[i] & level3_p[i-4];
86     end
87 endgenerate
88
89 //gen Q value
90 generate
91     for(i=1;i<=8;i=i+1)
92         assign level4_Q[i] = level3_Q[i];
93     for(i=9;i<=16;i=i+1) begin
94         assign level4_Q[i] = (level4_p[i] & level3_Q[i-8]) | level3_Q[i];
95     end
96 endgenerate

```

16比特Kogge-Stone加法器level4生成逻辑

```

85 //sum generator
86 generate
87     for(i=1; i<=16; i=i+1) begin
88         assign S[i] = c[i-1] ^ p[i];
89     end
90 endgenerate

```

16比特Kogge-Stone加法器和S生成逻辑

欢迎批评指正，更多阅读，关注“[纸上谈芯](#)”，不定期更新，共同学习：



Kogge-Stone 树形加法器

Blog Life

Kogge-Stone 树形加法器1. Kogge-Stone2. 超前进位加法器3. Kogge-Stone 并行算法4. 树形结构 1. Kogge-Stone Kogge-Stone 加法器是利用 Peter M. Kogge 和 Harold S. Stone

< Verilog实现加法器 > koggle-stone加法器设计——超前进位加法器改进

IC跳跳鱼的博客

一，内容介绍 加法器是数字电路中的最基础电路之一，也是CPU的核心功能之一。在这个专栏，我会把所有我知道的数字电路的加法器相关模型都实现一遍并解释其原理。 编

halfstone 原理_【HDL系列】Kogge-Stone加法器原理与设计

从图中足以见Kogge-Stone结构加法器布线尤其拥塞,但其好处在于进位链生成的延迟单位少。在verilog中根据其规律使用generate和endgenerate语句生成各级信号如下。 16比

各种加法器的比对分析与Verilog实现(3)_kogge-stone加法器

1. Kogge-Stone加法器的Verilog实现 module koggle_stone (input[3:0] a_i,b_i, inputc_i, outputwire [3:0] c_o); wire [3:0] q,p; wire [4:0] carry; assignp=a_i&b_i; ...

verilog加法器_【HDL系列】Kogge-Stone加法器原理与设计

weixin_39940154的博客

Kogge-Stone加法器是利用Peter M. Kogge和Harold S. Stone于1972年提出的一种并行算法生成的一种树形加法器。一、Kogge-Stone并行算法Kogge和Stone根据一般m阶递归

四位行波进位加法器_【HDL系列】硬件加法器原理与设计小结

weixin_39634898的博客

硬件加法器种类繁多，对于不同的设计，加法器的需求也不一样。在前端设计中，使用符号“+”便可轻而易举地实现加法器。只 觉得还不错？ 一键收藏 0 法器类型，或许可



纸上谈芯

关注

5

41

1



专栏