【HDL系列】Kogge-Stone加法器原理与设计

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X1,X2,X3,…,Xn，其中Xi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Xi，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Ai和Bi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X1=b1,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ar系数使用的是连乘符合，之后与bj相乘后使用的是累积和符号。

所以，

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

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

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

$$\cdots\cdots$$

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

$$\cdots\cdots$$

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

上式又可以重新写成如下形式：

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

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

$$\cdots\cdots$$

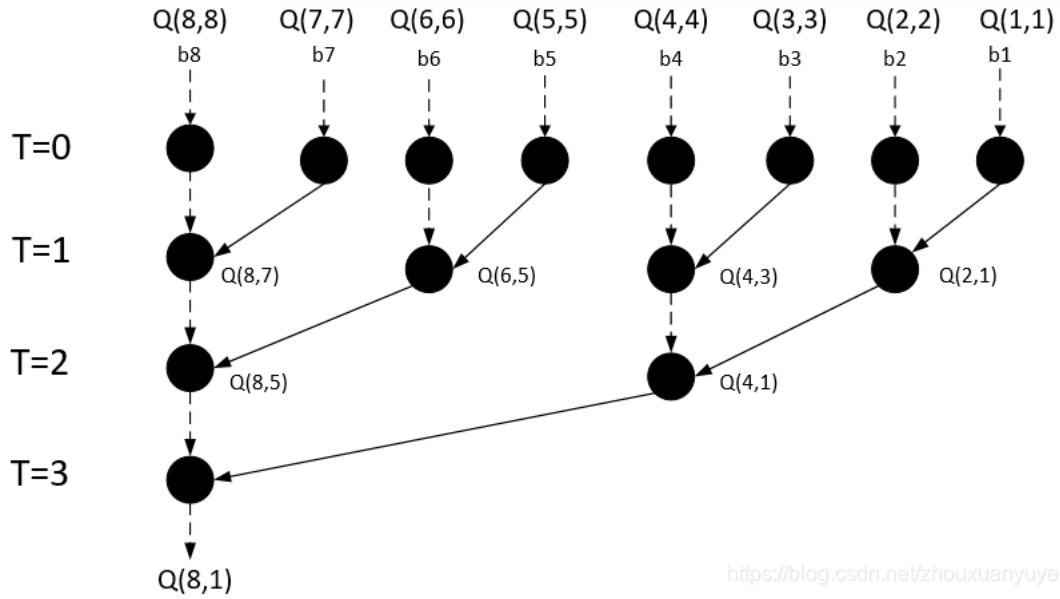
$$\begin{aligned}\hat{Q}(4,1) = x_4 &= a_4 x_3 + b_4 \\ &= a_4 a_3 x_2 + a_4 b_3 + b_4 \\ &= a_4 a_3 \hat{Q}(2,1) + \hat{Q}(4,3)\end{aligned}$$

$$\cdots\cdots$$

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。



x8计算示意图

TABLE I

Processor	$A(i)$	$T=0$ $B(i)$	$A(i)$	$T=1$ $B(i)$	$A(i)$	$T=2$ $B(i)$	$A(i)$	$T=3$ $B(i)$
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	$a_2x_1 + b_2 = x_2 = \hat{Q}(2,1)$	a_2	$X_2 = \hat{Q}(2,1)$	a_2	X_2
3	a_3	$b_3 = \hat{Q}(3,3)$	a_3a_2	$a_3b_2 + b_3 = \hat{Q}(3,2)$	a_3a_2	$(a_3a_2)X_1 + (a_3b_2 + b_3) = X_3 = \hat{Q}(3,1)$	a_3a_2	X_3
4	a_4	$b_4 = \hat{Q}(4,4)$	a_4a_3	$a_4b_3 + b_4 = \hat{Q}(4,3)$	$a_4a_3a_2$	$(a_4a_3)X_2 + (a_4b_3 + b_4) = X_4 = \hat{Q}(4,1)$	$a_4a_3a_2$	X_4
5	a_5	$b_5 = \hat{Q}(5,5)$	a_5a_4	$a_5b_4 + b_5 = \hat{Q}(5,4)$	$a_5a_4a_3a_2$	$a_5a_4(a_3b_2 + b_3) + a_5b_4 + b_5 = \hat{Q}(5,2)$ $= \sum_{w=2}^{5}\left(\prod_{m=w+1}^{5} a_m\right) b_w$	$\prod_{m=2}^{5} a_i^*$	X_5
6	a_6	$b_6 = \hat{Q}(6,6)$	a_6a_5	$a_6b_5 + b_6 = \hat{Q}(6,5)$	$a_6a_5a_4a_3$	$\sum_{w=3}^{6}\left(\prod_{m=w+1}^{6} a_m\right) b_w = \hat{Q}(6,3)$	$\prod_{m=2}^{6} a_i^*$	X_6
7	a_7	$b_7 = \hat{Q}(7,7)$	a_7a_6	$a_7b_6 + b_7 = \hat{Q}(7,6)$	$a_7a_6a_5a_4$	$\sum_{w=4}^{7}\left(\prod_{m=w+1}^{7} a_m\right) b_w = \hat{Q}(7,4)$	$\prod_{m=2}^{7} a_i^*$	X_7
8	a_8	$b_8 = \hat{Q}(8,8)$	a_8a_7	$a_8b_7 + b_8 = \hat{Q}(8,7)$	$a_8a_7a_6a_5$	$\sum_{w=5}^{8}\left(\prod_{m=w+1}^{8} a_m\right) b_w = \hat{Q}(8,5)$	$\prod_{m=2}^{8} a_i^*$	X_8

X1至X8的计算步骤、系数与公式，来源Kogge-Stone论文

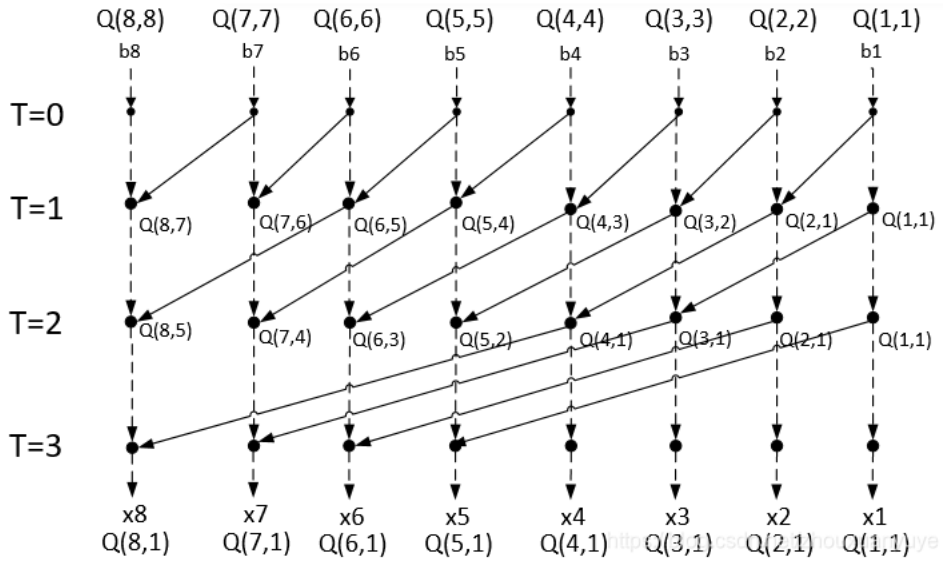
上图为x1-x8的计算步骤，系数和公式，分三个时间步骤完成：

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：



X1至X8的计算图示

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：

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

其中：

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

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：

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

所以ci算式写成Kogge-Stone形式：

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

所以，如对于C1-C4等的生成，其算式如下：

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

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

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

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

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

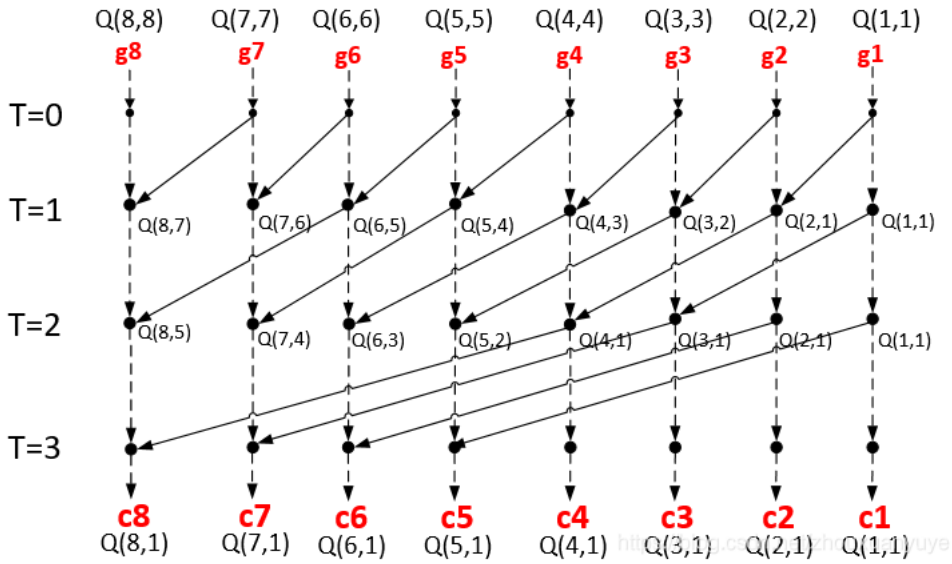
$$\hat{Q}(4,1) = c_4 = p_4 c_3 + g_4$$

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

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

$$\cdots\cdots$$

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

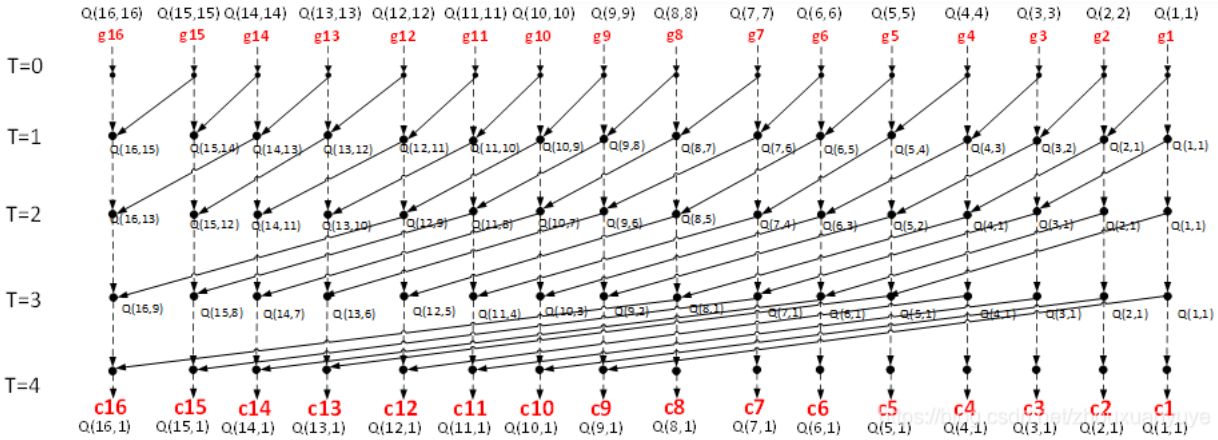
设计一个16比特的Kogge-Stone加法器，分三部分：

（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：



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

```
//level3
    //gen coe
    generate
        for(i=1;i<=16;i=i+1) begin
            assign level3_p[i] = level2_p[i] & level2_p[i-2];
        end
    endgenerate

    //gen Q value
    generate
        for(i=1;i<=4;i=i+1)
            assign level3_Q[i] = level2_Q[i];
        for(i=5;i<=16;i=i+1) begin
            assign level3_Q[i] = (level3_p[i] & level2_Q[i-4]) | level2_Q[i];
        end
    endgenerate
```


16比特Kogge-Stone加法器level3生成逻辑

```
//level4
    //gen coe
    generate
        for(i=1;i<=16;i=i+1) begin
            assign level4_p[i] = level3_p[i] & level3_p[i-4];
        end
    endgenerate

    //gen Q value
    generate
        for(i=1;i<=8;i=i+1)
            assign level4_Q[i] = level3_Q[i];
        for(i=9;i<=16;i=i+1) begin
            assign level4_Q[i] = (level4_p[i] & level3_Q[i-8]) | level3_Q[i];
        end
    endgenerate
```


16比特Kogge-Stone加法器level4生成逻辑

```
//sum genertator
generate
    for(i=1; i<=16; i=i+1) begin
        assign S[i] = c[i-1] ^ p[i];
    end
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: