课程作业报告2

[Kogge-Stone加法器设计验证 1](#_Toc30644)

[设计原理 2](#_Toc10791)

[设计细节 2](#_Toc1053)

[验证 4](#_Toc28478)

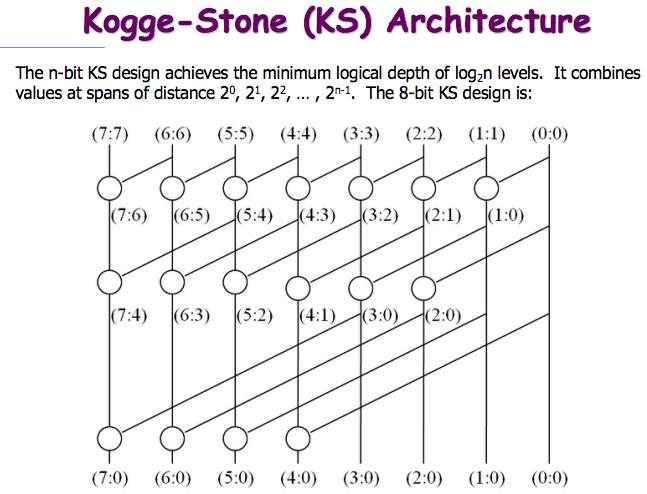
[总结 4](#_Toc7165)

# 

# Kogge-Stone加法器设计验证

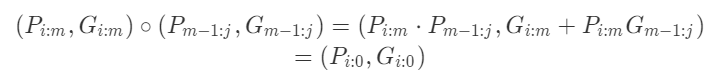
张朕银20212020089 薛玥20212020163

## 设计原理



根据上图的拓扑关系，对点操作进行有序映射，从而节省组合逻辑运算时间，以及逻辑资源消耗。

点操作：



## 设计细节

定义点操作模块。

module dot(Pi1,Gi1,Pi0,Gi0,Po,Go);  
//P"i:j" = P"i:m" & P"m-1:j"  
//G"i:j" = G"i:m" | P"i:m" & G"m-i:j"  
//Pi0: P"m-1:j"  
//Gi0: G"m-1:j"  
//Pi1: P"i:m"  
//Gi1: G"i:m"  
//Po: P"i:j"  
//Go: G"i:j"  
input Pi0,Gi0,Pi1,Gi1;  
output Po,Go;  
   
assign Po = Pi1 & Pi0;  
assign Go = Gi1 | (Pi1 & Gi0);  
endmodule

定义一个名为KS\_Adder的模块，为顶层模块，并定义其端口和位宽：

module KS\_Adder(A,B,Ci,Co,S);  
input [15:0]A;  
input [15:0]B;  
input Ci;  
output Co;  
output [15:0]S;

然后定义内部线路C，用来表示每一位的进位。其中表示，表示。

wire [16:0]C;  
assign C[0]=Ci;  
assign Co=C[16];

然后根据KS模型，定义所有信号（）。

wire P\_0\_0,G\_0\_0;  
wire P\_1\_1,G\_1\_1;  
wire P\_2\_2,G\_2\_2;  
......  
wire P\_15\_12,G\_15\_12;  
wire P\_15\_8,G\_15\_8;  
wire P\_15\_0,G\_15\_0;

然后给所有信号加上驱动信号。

assign P\_0\_0 = A[0] ^ B[0];  
assign P\_1\_1 = A[1] ^ B[1];  
assign P\_2\_2 = A[2] ^ B[2];  
......  
assign P\_15\_15 = A[15] ^ B[15];  
   
assign G\_0\_0 = A[0] & B[0];  
assign G\_1\_1 = A[1] & B[1];  
assign G\_2\_2 = A[2] & B[2];  
......  
assign G\_15\_15 = A[15] & B[15];

然后给所有信号添加点操作。

dot dot\_1\_0(.Pi1(P\_1\_1),.Gi1(G\_1\_1),.Pi0(P\_0\_0),.Gi0(G\_0\_0),.Po(P\_1\_0),.Go(G\_1\_0));  
dot dot\_2\_1(.Pi1(P\_2\_2),.Gi1(G\_2\_2),.Pi0(P\_1\_1),.Gi0(G\_1\_1),.Po(P\_2\_1),.Go(G\_2\_1));  
dot dot\_2\_0(.Pi1(P\_2\_1),.Gi1(G\_2\_1),.Pi0(P\_0\_0),.Gi0(G\_0\_0),.Po(P\_2\_0),.Go(G\_2\_0));  
......  
dot dot\_15\_0(.Pi1(P\_15\_8),.Gi1(G\_15\_8),.Pi0(P\_7\_0),.Gi0(G\_7\_0),.Po(P\_15\_0),.Go(G\_15\_0));

再给添加驱动信号。

assign C[1] = G\_0\_0 | (P\_0\_0 & C[0]);  
assign C[2] = G\_1\_0 | (P\_1\_0 & C[0]);  
......  
assign C[16] = G\_15\_0 | (P\_15\_0 & C[0]);  
   
assign S[0] = P\_0\_0 ^ C[0];  
assign S[1] = P\_1\_1 ^ C[1];  
......  
assign S[15] = P\_15\_15 ^ C[15];

最后语法收尾

endmodule

## 验证

由于16位加法器的全覆盖验证需要进行次验证，而这样庞大的验证量会消耗过多时间。因此最终采用的testbench方案中，采用了随机测试的方法，最终通过验证。验证覆盖率为$\cfrac{1}{2048}$。

## 总结

进行了KS\_Adder的设计和验证，进一步加深了对SystemVerilog和组合逻辑硬件设计验证的认识。