# 流水线设计

# 前言

本文从四部分对流水线设计进行分析，具体如下：

一 什么是流水线

二 什么时候用流水线设计

三 使用流水线的优缺点

四 流水线加法器举例

# 第一 什么是流水线

流水线设计就是将组合逻辑系统地分割，并在各个部分（分级）之间插入寄存器，并暂存中间数据的方法。

目的是将一个大操作分解成若干的小操作，每一步小操作的时间较小，所以能提高频率，各小操作能并行执行，所以能提高数据吞吐率（提高处理速度）。

# 第二 什么时候用流水线设计

使用流水线一般是时序比较紧张，对电路工作频率较高的时候。典型情况如下：

1）功能模块之间的流水线，用乒乓 buffer 来交互数据。代价是增加了 memory 的数量，但是和获得的巨大性能提升相比，可以忽略不计。

2） I/O 瓶颈，比如某个运算需要输入 8 个数据，而 memroy 只能同时提供 2 个数据，如果通过适当划分运算步骤，使用流水线反而会减少面积。

3）片内 sram 的读操作，因为 sram 的读操作本身就是两极流水线，除非下一步操作依赖读结果，否则使用流水线是自然而然的事情。

4）组合逻辑太长，比如(a+b)\*c，那么在加法和乘法之间插入寄存器是比较稳妥的做法。

# 第三 使用流水线的优缺点

1）优点： 流水线缩短了在一个时钟周期内给的那个信号必须通过的通路长度，增加了数据吞吐量，从而可以提高时钟频率，但也导致了数据的延时。举例如下：

例如：一个 2 级组合逻辑，假定每级延迟相同为 Tpd，

1.无流水线的总延迟就是 2Tpd，可以在一个时钟周期完成，但是时钟周期受限制在 2Tpd；

2.流水线：每一级加入寄存器（延迟为 Tco）后，单级的延迟为 Tpd+Tco，每级消耗一个时钟周期，流水线需要 2 个时钟周期来获得第一个计算结果，称 为首次延迟，它要 2\*（ Tpd+Tco），但是执行重复操作时，只要一个时钟周期来获得最后的计算结果，称为吞吐延迟（ Tpd+Tco）。可见只要 Tco 小于 Tpd，流水线就可以提高速度。 特别需要说明的是，流水线并不减小单次操作的时间，减小的是整个数据的操作时间，请大家认真体会。

2） 缺点：功耗增加，面积增加，硬件复杂度增加，特别对于复杂逻辑如 cpu 的流水线而言，流水越深，发生需要 hold 流水线或 reset 流水线的情况时，时间损失越大。 以使用流水线并非有利无害，大家需权衡考虑。

# 第四 一个 8bit 流水线加法器的小例子

非流水线：

module add8(

a,

b,

c);

input [7:0] a;

input [7:0] b;

output [8:0] c;

assign c[8:0] = {1'd0, a} + {1'd0, b};

endmodule

采用两级流水线：第一级低 4bit,第二级高 4bit，所以第一个输出需要 2 个时钟周期有效，后面的数据都是 1 个周期

之后有效。

module adder8\_2(

input clk,

input cin,

input [7:0] cina,

input [7:0] cinb,

output reg [7:0] sum,

output reg cout

);

reg cout1; //插入的寄存器

reg [3 :0 ] sum1 ; //插入的寄存器

reg [3:0] cina\_reg;

reg [3:0] cinb\_reg;//插入的寄存器

always @(posedge clk) //第一级流水

begin

{cout1 , sum1} <= cina[3:0] + cinb [3:0] + cin ;

end

// 打这一拍似乎没什么必要

always @(posedge clk)

begin

cina\_reg <= cina[7:4];

cinb\_reg <= cinb[7:4];

end

always @(posedge clk) //第二级流水

begin

{cout ,sum[7:0]} <= {{1'b0,cina\_reg[3:0]} + {1'b0,cinb\_reg[3:0]} + cout1 ,sum1[3:0]} ;

end

endmodule

这里讲到的流水线，主要是一种硬件设计的算法，如第一条中表述的流水线设计就是将组合逻辑系统地分割，并在各个部分（分级）之间插入寄存器，并暂存中间数据的方法。

针对处理器中的流水线结构。比如，比如 5—6 个不同功能的电路单元组成一条指令处理流水线，然后将一条指令分成 5—6 步后再由这些电路单元分别执行，这样就能实现在一个 CPU 时钟周期完成一条指令，因此提高 CPU 的运算速度。 一般的 CPU 中,每条整数流水线都分为四级流水， 即指令预取、 译码、 执行、 写回结果， openrisc采用的是 5 级整数流水线。

当然它们的核心思想都是利用并行执行提高效率。

总结一下，流水线就是插入寄存器，以面积换取速度。