从电路设计的角度入门VerilogHDL

邸志雄，西南交通大学

www.dizhixiong.cn

CONTENTS

目录

一 概述

二 编写方法

三 功能仿真

知识点列表

流水线

场景设计：敌人太多，队友协助装弹，顺溜只完成发射动作

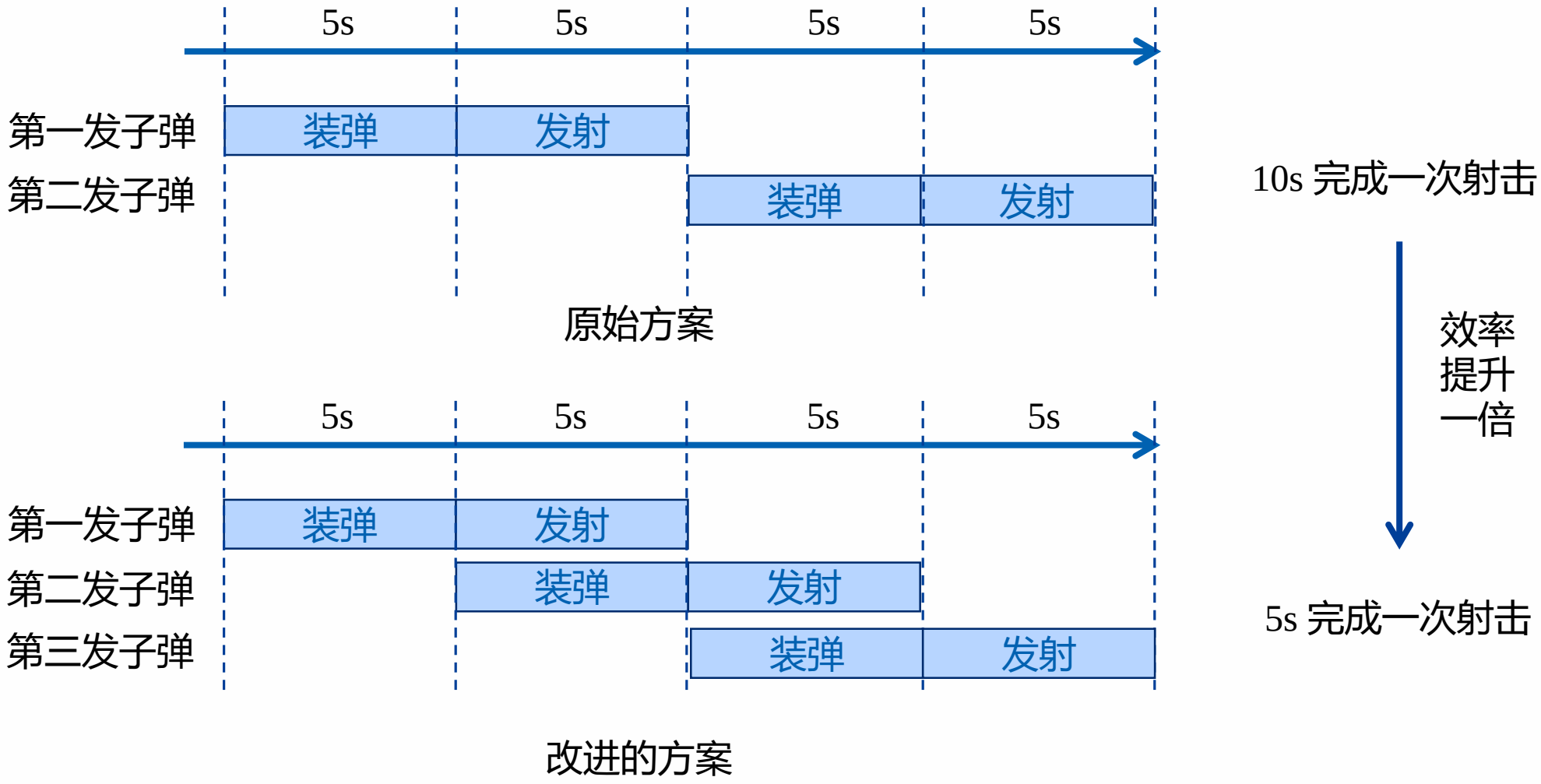
假设：

- 射击流程分为两步：先装弹，再发射。
- 装弹 5s，发射 5s
- 顺溜的队友枪法太差，命中率为0；顺溜命中率为100%

从量化的角度来分析消灭敌人的效率是否有提升?

[万木杨, 大话处理器出版社, 清华大学出版社, 2011-10-01]

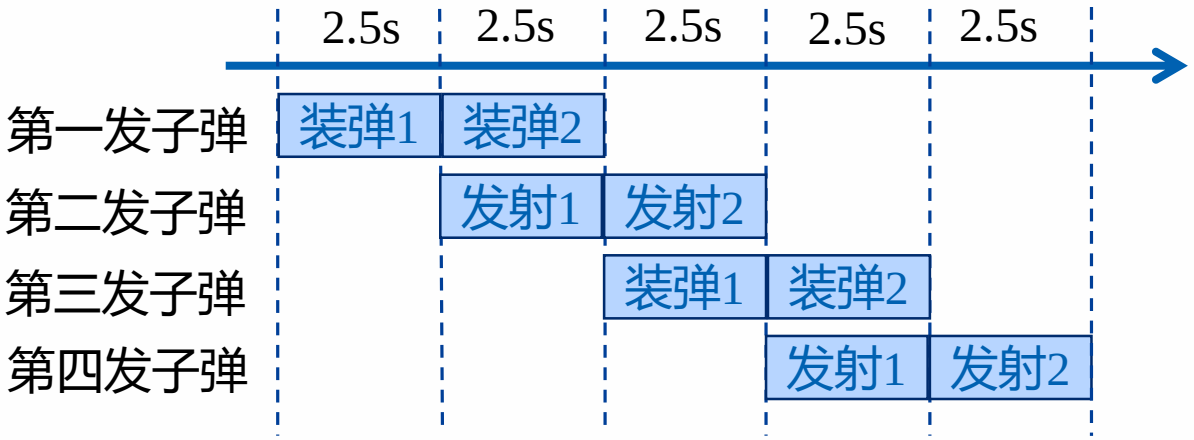
流水线



子弹如同流水一样，在各个环节并行流动，这个过程被形象地成为“流水线”

[万木杨, 大话处理器出版社, 清华大学出版社, 2011-10-01]

流水线



2.5s 完成一次射击

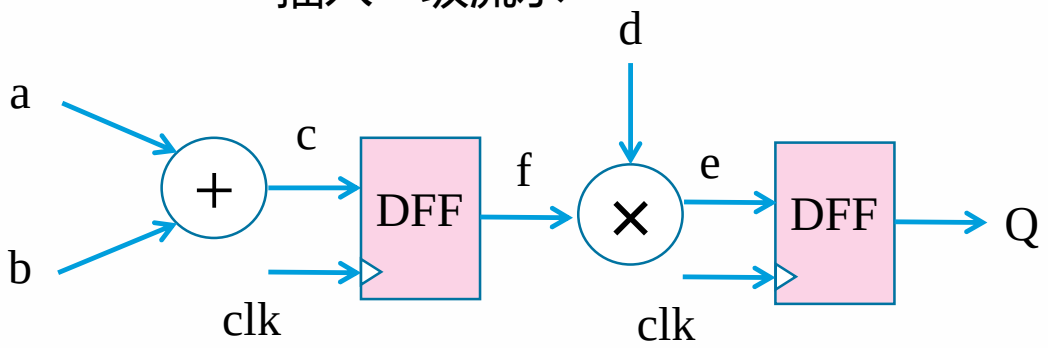
效率再提升一倍

不考虑其他因素情况下，流水线级数越多，工作效率越高

[万木杨, 大话处理器出版社, 清华大学出版社, 2011-10-01]

流水线

插入一级流水




```
always  @ (posedge clk  or  negedge rst_n )
begin
   if (rst_n == 1'b 0)
      f <= 0 ;
   else
      f  <= a + b ;
end
```

```
always  @ (posedge clk  or  negedge rst_n )
begin
   if (rst_n == 1'b 0)
      Q  <= 0 ;
   else
      Q  <= f * d   ;
end
```

Q在第二个周期才得到计算结果；逻辑路径的延迟减少为原来的二分之一（假设“+”与“*”延迟相等）

知识点列表

参数化

module 中的参数一般是用作定义其中常量的工具。

以下代码定义了半加器的"与门"和"异或门"的延时分别为 2 和 4 个时间单位。

```
module half_adder (co, sum, a, b);
output co, sum;
input a, b;
parameter and_delay = 2;
parameter xor_delay = 4;
and #and_delay u1(co, a, b);
xor #xor_delay u2(sum, a, b);
endmodule
```

参数化

```
module mux2to1_N(Sel, A, B, O);
    parameter N = 1
    input [N-1:0] A;
    input [N-1:0] B;
    input Sel;
    output [N-1:0] O;
    mux2to1 mux0[N-1:0] (Sel, A, B, O);
endmodule
…
Mux2to1_N #(4) mux1 (S, in1, in2, out)
```

参数化

- Like the C pre-processor
  - But uses ` (back-tick) instead of #
  - Constants: `define
    - No parameterized macros
    - Use ` before expanding constant macro

```
`define letter_A 8'h41
wire w = `letter_A;
```

  - Conditional compilation: `ifdef, `endif
  - File inclusion: `include

- Parameter vs `define
  - Parameter only for “per instance” constants
  - `define for “global” constants

模块实例化

```
module comp(out_port1, out_port2, in_port2);
        output out_port1, out_port2;
input in_port1, in_port2;
……·
endmodule

module demo_top1;     //位置对应调用法
comp gate1(Q,R,J,K);  //将模块定义时的端口名对应地用与之相连的信号线名称所替代
endmodule          //调用时端口名表项必须与模块定义时的端口名表项一一对应

module demo_top2;     //端口名对应调用法
comp  gate2(.in_port(K), .out_port1(Q), .out_port2®, .in_port1(J)); /*直接表示定义-调用间的对应关系，表示方式为".定义时的端口名（调用时与之相连的信号名）"*/
endmodule    //调用时端口名的排列顺序可以随意改变

module demo_top3;  //存在不连接端口的调用方式
comp gate3(Q,  , J, K);
endmodule
```

知识点列表

总结

如何书写高性能verilog HDL：

1.牢记并理解可综合“四大法宝”所对应的硬件结构

2.写前，确认电路指标是什么：性能？面积？

3.硬件思维方式，代码不再是一行行的代码而是一块一块的硬件模块；

4.对所需实现的硬件电路“胸有成竹”；

初学者的误区：把verilog代码当做了程序，把电路设计当成了编程。

参考资料

轻松成为设计高手:VerilogHDL实用精解. EDA先锋工作室. 北京航空航天大学出版社. 2012年.

中国大学MOOC平台：芯动力——硬件加速设计方法

https://www.icourse163.org/course/SWJTU-1207492806

谢谢！

邸志雄，西南交通大学

www.dizhixiong.cn