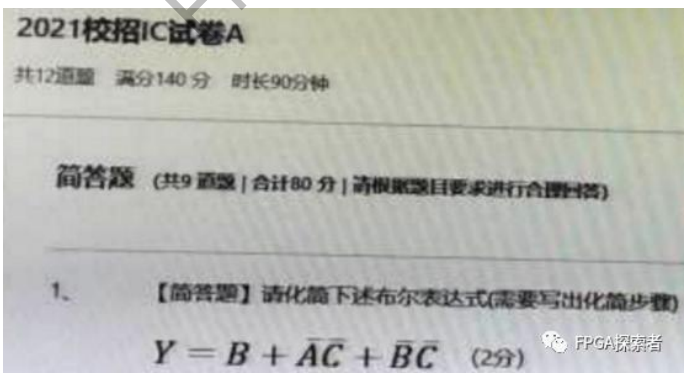
联发科技 2021 校招 IC 笔试题全部解析【数字 IC 设计验证】【MTK 笔试】

2020 年 7 月，校招 IC。
PDF 版在【FPGA 探索者】公众号回复【联发科笔试题】获取，打印后阅读。

简答题

1. 逻辑化简【公式化简】【卡诺图化简】

化简 Y = B + (~A)&(~C) + (~B)&(~C)。



卡诺图化简

卡诺图中，每个方格是一个 **最小项**，相邻方格的最小项只有 1 位不同。

n 个变量的逻辑函数，有 2^n 个最小项，对应卡诺图 2^n 个方格。

（2021 校招华为 FPGA 逻辑，第 33 题）。

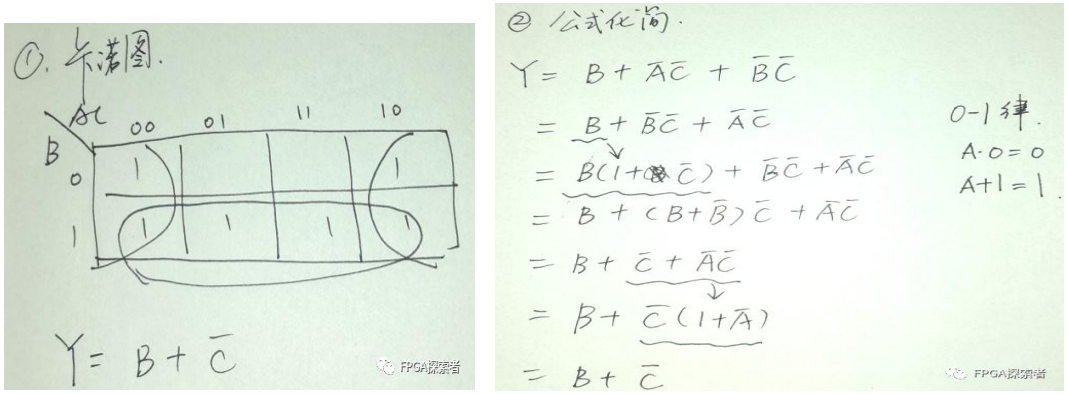
【华为 2021 秋招】FPGA 逻辑笔试解析【独家】【数字 IC】【FPGA 逻辑】

公式化简

主要利用 0-1 律，很常用的数字运算规律。

经常需要根据需要，将 1 变成 1+ X 形式 进行化简。

公式化简如下，和卡诺图化简结果一致。



数字逻辑运算定律

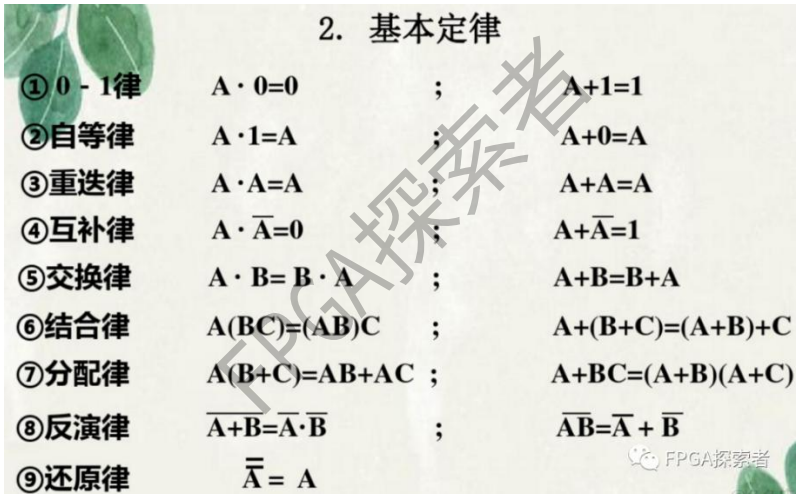
常考的：

（1）0-1 律；

（2）反演律。

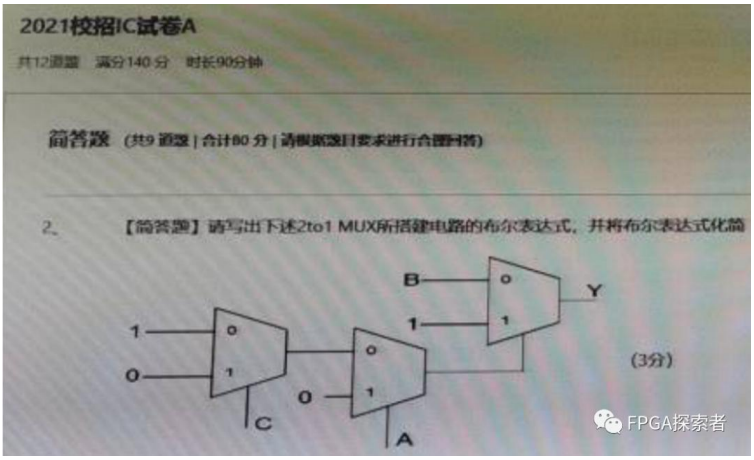
反演律：华为 2021 校招 FPGA 逻辑，第 17 题

【华为 2021 秋招】FPGA 逻辑笔试解析【独家】【数字 IC】【FPGA 逻辑】

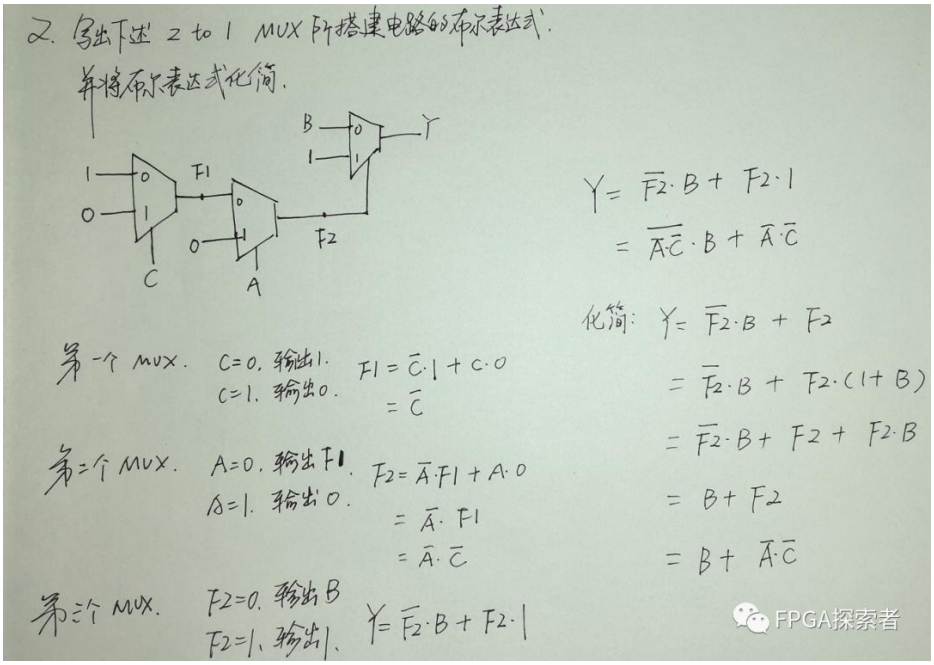


2. 逻辑表达式

请写出下述 2 to 1 MUX 所搭建电路的布尔表达式，并将布尔表达式化简。



作答：



3. glitch free 时钟无缝切换电路

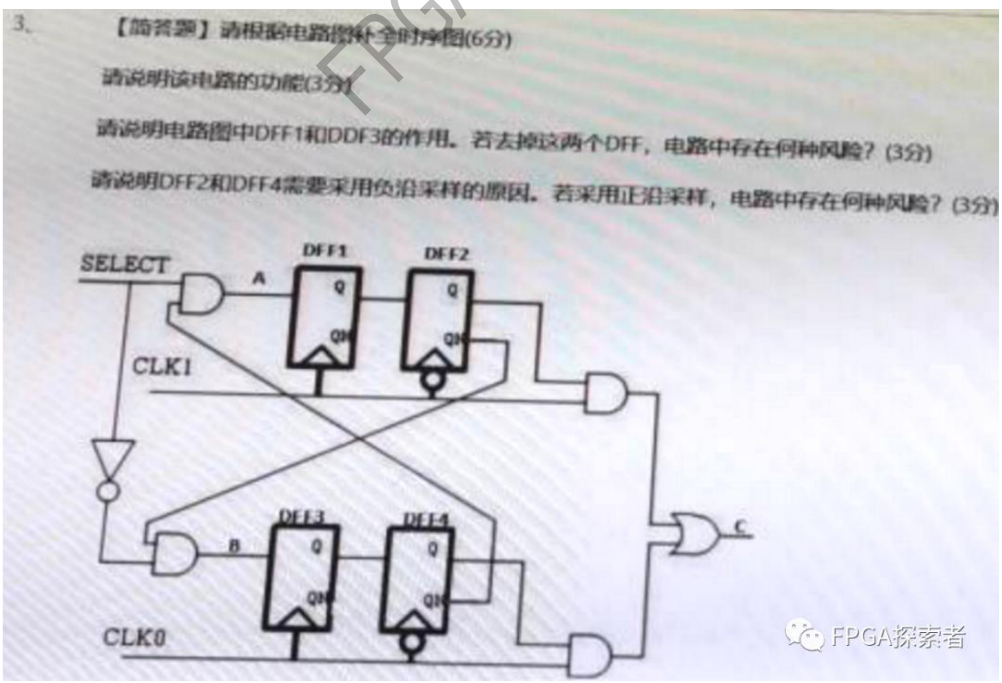
（联发科技-2021 校招 IC 卷 A——时钟无毛刺切换技术，15 分）

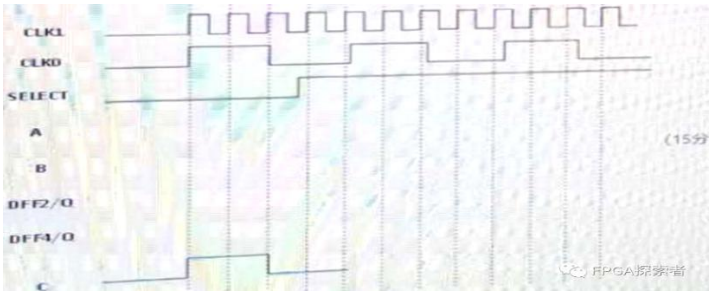
（1）根据电路图补全时序图；（6 分）

（2）说明电路的功能；（3 分）

（3）说明 DFF1 和 DFF3 的作用，去掉后有什么风险？（3 分）

（4）说明 DFF2 和 DFF4 为什么采用负沿采样？若采用正沿，会存在何种风险？（3 分）





问题分析

（1）该电路是**时钟无毛刺切换电路**；

参考：

联发科笔试题——Glitch free 无毛刺时钟切换电路、时钟无缝切换、时钟无毛刺切换技术

（2）数字电路中的风险问题：

组合逻辑：竞争冒险，出现毛刺

时序逻辑：亚稳态

（2）~（4）问题解答

第一问时序图比较复杂，先解答（2）~（4）问，后面具体分析（1）。

（2）说明电路的功能；

时钟无毛刺切换（glitch free）。

FPGA探索者

（3）说明DFF1和DFF3的作用，去掉后有什么风险？

DFF1 和 DFF3 用于同步打拍，降低亚稳态发生的概率。
不能去掉，若去掉，则存在亚稳态风险。

FPGA探索者

（4）说明DFF2和DFF4为什么采用负沿采样？
若采用正沿，会存在何种风险？

主要考虑后面的与门。

下降沿采样可以保证：
DFF2 和 DFF4 的输出 Q 在跳变时，
一定发生在时钟 CLK 的低电平区域。

FPGA探索者

如果采用正沿，则可能出现毛刺。

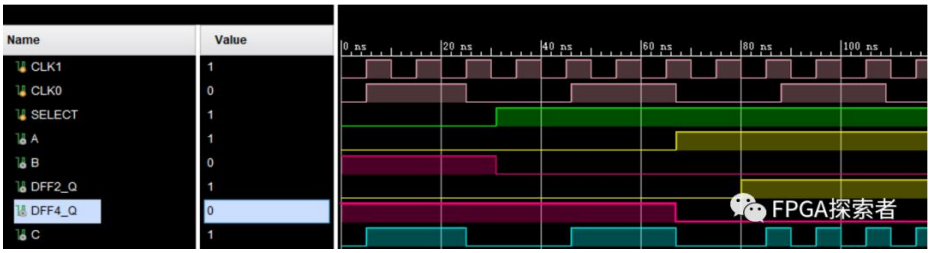
在时钟的下降沿处寄存选择控制信号，保证了控制信号不会在 2 个时钟源的高电平处进行跳变，这样就防止对输出时钟进行截断（截断导致毛刺）。

参考：

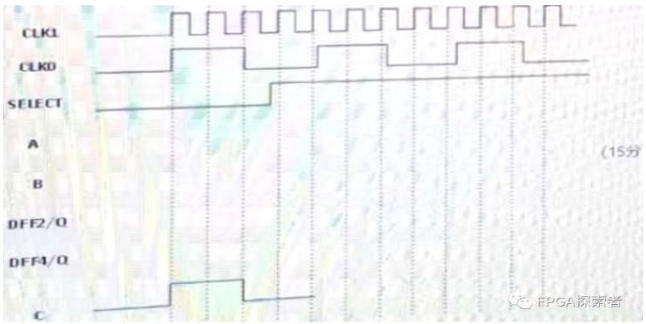
门控时钟与控制信号电平、与门门控、或门门控、上升沿门控、下降沿门控

（1）的时序图

如图所示，包括 A、B、DFF2 的 Q 输出、DFF4 的 Q 输出、C。



问题的关键在 C 的波形。



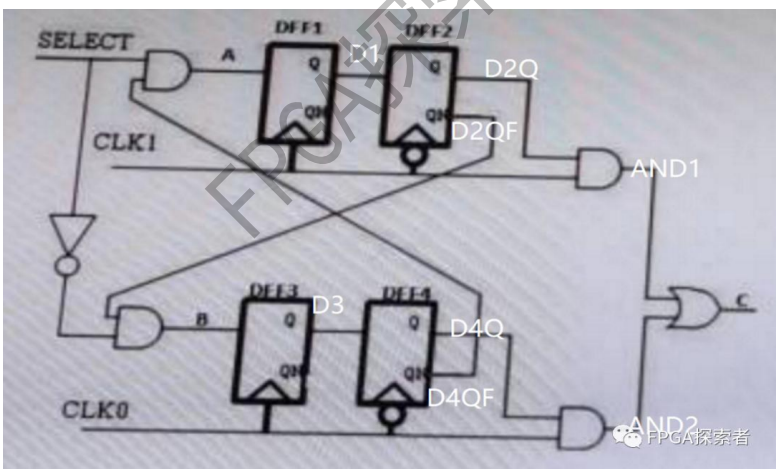
在 CLK0 的第一个周期内，C 的波形与 CLK0 完全一致，说明

（1）C 来自 AND2，**且 D4Q 为 1，D2Q 为 0**；

（2）向前推导，D3Q 的初始值为 1，D1Q 的初始值为 0；

（3）D2QF = 1，D4QF = 0；

（4）D4QF = 0，A = 0；



对 B 的状态：

（1）SELECT 初始为 0，后面变成 1，B 来自 D2QF 和 ~SELECT，只有有 1 个为 0，则 B 为 0，所以 B 的状态比较好判断，当 SELECT 从 0 变为 1 以后，B 一定是 0；

（2）在 SELECT 保持为 0 的阶段，B 的值取决于 D2QF，在此阶段内，D2QF 恒为 1，所以 B 恒为 1；

从 B 向后分析 D3，在 CLK0 的上升沿，D3 随 B 变化：

（1）前文推导 D3 初始 1；

（2）CLK0 的第 2 个上升沿，D3 变为 0，且从此一直为 0；

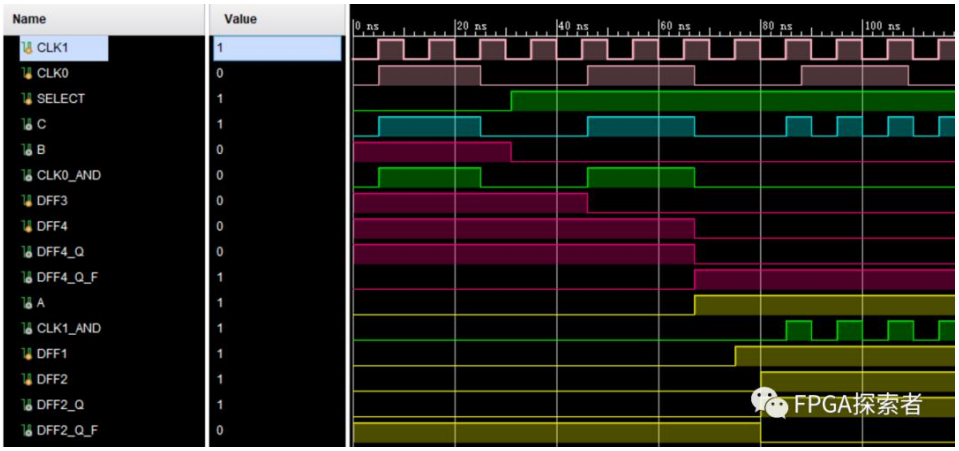
从 D3 向后分析 D4Q，在 CLK1 的下升沿，D4Q 随 D3 变化：
（1）前文推导 D4Q 初始 1；
（2）CLK0 的第 2 个下降沿，D4Q 变为 0，且从此一直为 0；
（3）与此同时，D4QF 发送相应变化，D4QF = ~ D4Q，在 CLK0 的第 2 个下降沿，D4QF 变为 1；

此时分析 A：
（1）当 SELECT 和 D4QF 同时为 1，A = 1；
（2）初始时 SELECT = 0，后面 SELECT 刚变为 1 的时候 D4QF 是 0，所以 A 一直是 0；
（3）SELECT 在 D4QF 前已经为 1，所以当 D4QF 变为 1 的时候，A 会和 D4QF 同时变为 1；

从 A 向后分析 D1，在 CLK1 的上升沿，D1 随 A 变化：
（1）前文推导 D1 初始 0；
（2）A 拉高后，CLK1 的第 1 个上升沿，D1 变为 1，且从此一直为 1；

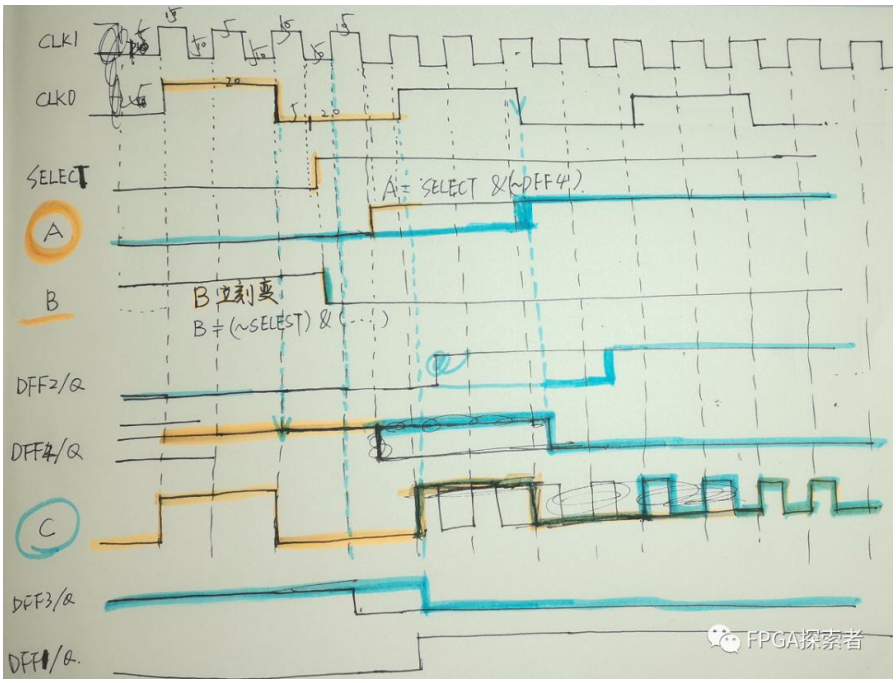
从 D1 向后分析 D2Q，在 CLK0 的下升沿，D2Q 随 D1 变化：
（1）前文推导 D2Q 初始 0；
（2）A 拉高后，CLK0 的第 1 个下降沿，D2Q 变为 1，且从此一直为 1；
（3）与此同时，D2QF 发送相应变化，D2QF = ~ D2Q，由于 B 已经变为 0，所以此时 D2QF 变成 0 不影响 B 仍然为 0；

根据 D2Q 和 CLK1 得到 AND1 的波形；根据 D4Q 和 CLK0 得到 AND2 的波形；由此得到 C 的波形。



上图中：
DFF4_Q 代表 DFF4 的 Q 输出；
DFF4_Q_F 代表 DFF4 的 Q 取反输出；
其他类似表示。

手绘版。



门控时钟、时钟切换相关文章：

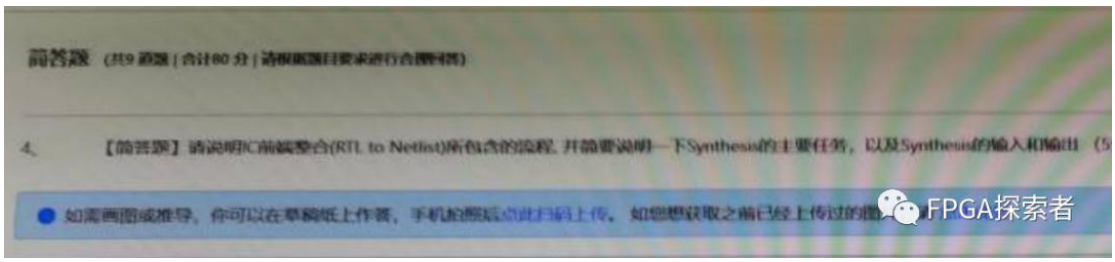
[门控时钟与控制信号电平、与门门控、或门门控、上升沿门控、下降沿门控]

[联发科笔试题——Glitch free 无毛刺时钟切换电路、时钟无缝切换、时钟无毛刺切换技术]

[来看个联发科的大题（6）——联发科技-2021 校招 IC 卷 A——时钟无毛刺切换技术]

4. IC 前端流程

请说明 IC 前端整合（RTL To Netlist）所包含的流程，并简要说明一下 Synthesis 的主要任务，以及 Synthesis 的输入和输出。



以门级网表（Netlist）生成为分界线，之前称为前端，之后称为后端。

布局布线之前可以认为是前端，布局布线到流片是后端。

前端：逻辑设计，RTL ——》 Netlist 门级网表；

后端：物理设计，Netlist 门级网表 ——》 物理版图；

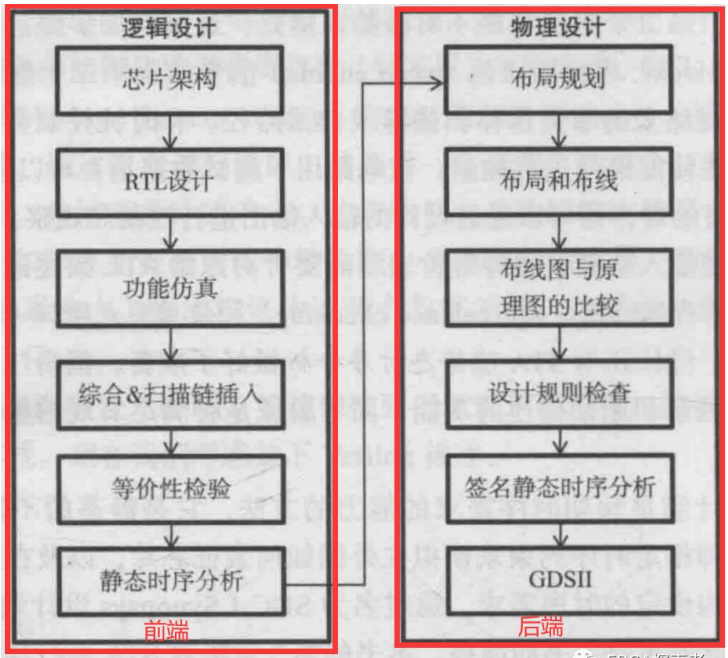
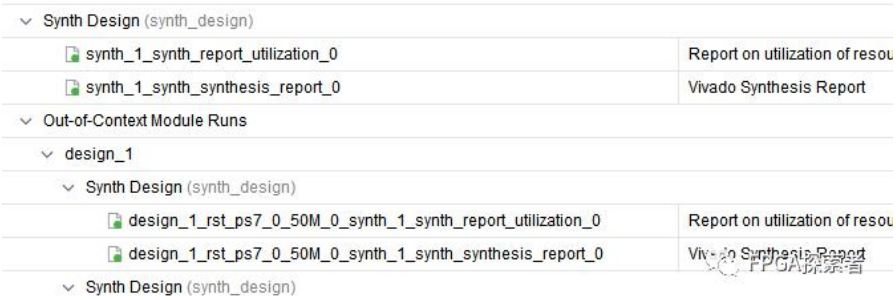


图 1.2　ASIC 设计流程

数字前端设计（front-end）：
以生成可以布局布线的网表为终点；

数字后端设计（back-end）：
以生成可以进行流片的 GDSII 文件为终点；

Synthesis：综合，主要任务是将 RTL 代码 转成 门级网表；
典型的网表文件由单元（Cell）、引脚（Pin）、端口（Port）、网络（Net）组成。
Synthesis 输入：RTL 代码，工艺库，约束
Synthesis 输出：Netlist 门级网表（用于布局布线），标准延迟文件（用于时序仿真）；综合后的报告；



功能仿真

验证 RTL 代码设计的**功能正确性**，**没有加入延时信息**，又叫**前仿真**，工具有 Mentor 的 **Modelsim**，Synopsys 的 **VCS**，Candence 的 NC-Verilog。
在综合、布局布线以后，有加入延时的后仿真（时序仿真）。

Synthesis 综合

逻辑综合的结果（目的）是把 HDL 代码翻译成门级网表 netlist，工具有 Synopsys 的 **Design Compiler（简称 DC）**，门级网表拿去布局布线。

DFT 可测性设计

DFT（Design for Test）可测性设计，为了测试而加入的设计，常见技术 ：

（1）**Scan Chain（扫描链），针对时序电路，测试寄存器（Flip-Flop）和组合逻辑**；

（2）**MBIST（Memory Bulit-in Self Test，内建自测试），测试芯片中存储资源**， rom 和 ram，在设计中插入内建自测试逻辑；

（3）**Boundary Scan（边界扫描），测试封装与 IO、芯片间互联**，主要逻辑有 TAP Controller 和 Boundary Scanchain）、JTAG（JTAG 是 boundary scan design 中用到的一个基本结构）。

ATPG（Automatic Test Pattern Generation，自动测试向量生成，基于扫描链，根据算法推算出应该加载到扫描链上的激励序列和期望序列，这样的序列称为测试向量）；

DFT 构建硬件结构，ATPG 生成测试向量。

形式验证

形式验证，属于验证范畴，从 **功能上** 对综合后的网表进行验证，常用的是**等价性检验**，以功能验证后的 HDL 设计为参考，对比综合后的网表功能，检验是否在功能上存在等价性，**保证综合后没有改变原先 HDL 描述的功能**。

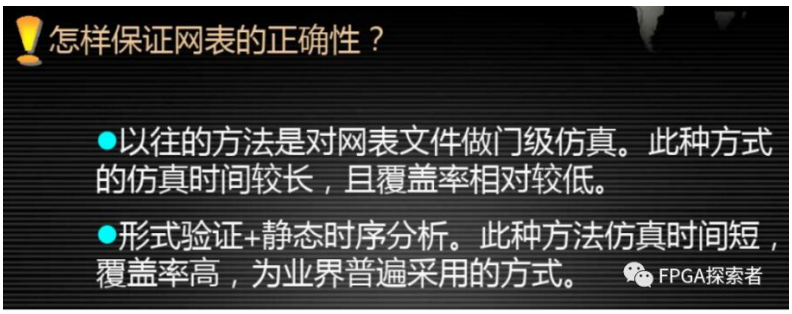
形式验证工具有 Synopsys 的 Formality。

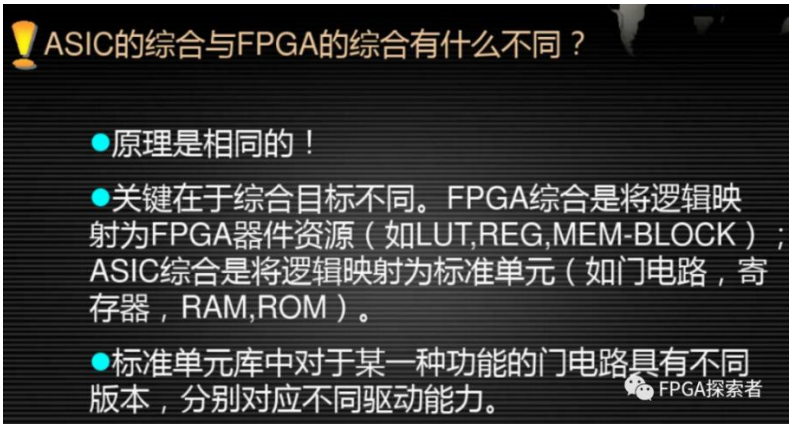
STA 静态时序分析

STA 静态时序分析（Static Timing Analyse），属于验证范畴，从**时序上**对综合后的网表进行验证，检查电路是否存在建立时间、保持时间等违例。

注意 STA 和 形式验证的不同，STA 从时序上验证，形式验证从功能上验证。

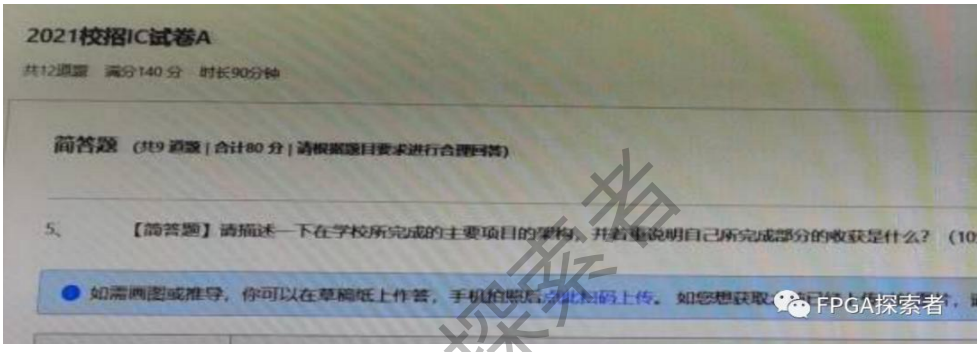
STA 工具有 Synosys 的 Prime Time。





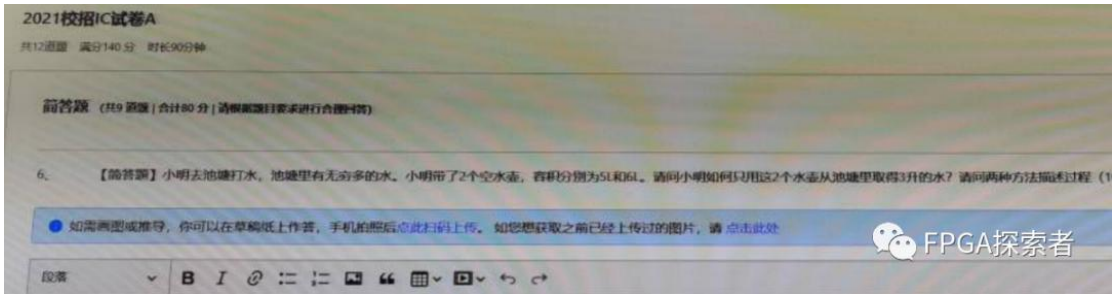
PDF 版在【FPGA 探索者】公众号回复【联发科笔试题】获取。

5. 所做项目



6. 逻辑思维题

小明去池塘打水，池塘里有无穷多的水，小明带了 2 个空水壶，容积分别为 5L 和 6L。请问小明如何用这 2 个水壶从池塘里取得 3 升的水？请给出两种方法。



答案：

第一种方法：

（1）6L 装满，倒进 5L，这样 6L 壶里还剩 1L，将 5L 壶里的全部倒掉，将 6L 壶里剩下的 1L 倒进 5L 壶里，这样 5L 壶里还能装 4L；

（2）6L 装满，倒进 5L 壶里 4L 水，这样 6L 壶里还剩 2L，将 5L 壶里的全部倒掉，将 6L 壶里剩下的 2L 倒进 5L 壶里，这样 5L 壶里还能装 3L；

（3）6L 装满，倒进 5L 壶里 3L 水，这样 6L 壶里还剩 3L；

第二种方法：

5L 装满，往 6L 里倒，不再详述。

7. Perl 语言哈希表

有下面四个人，他们名字分别是 Li Fei，Liu Qiang，Zhuang Ming，Tian Hua，写一个 Perl 程序，做到输入他们的姓，就能告诉这个人的名。

【思路】

```
哈希是 key/value 对的集合。
Perl中哈希变量以百分号 (%) 标记开始。
访问哈希元素格式：${key}。
```

代码：

```
#!/usr/local/bin/perl

%hash = ();                          # 哈希表
$hash{Li} = "Fei";
$hash{Liu} = "Qiang";
$hash{Zhang} = "Ming";
$hash{Tian} = "Hua";

my $a;
while( 1 ){

    print "Enter Xing:";
    $a = <STDIN>;                    # 获取键盘输入
    chomp $a;                        # 去掉输入的换行符

    if( $a ~~ "Exit") {              # 如果输入是 Exit
        die "Exit System\n";         # 退出系统
    }
    else {                           # 输入的是姓
        print $hash{$a};             # 打印名
        print "\n";                  # 打个换行
    }
}
```


【测试结果】

输入正确的姓，输出对应的名；

输入不正确的，输出为空；

输入 Exit，退出。

```
Enter Xing:Li
Fei
Enter Xing:Li
Fei
Enter Xing:Liu
Qiang
Enter Xing:Zhang
Ming
Enter Xing:Hua

Enter Xing:Tian
Hua
Enter Xing:LL
```

Perl 语言文章：

来看个联发科秋招的一个大题（4）——2021 校招 Perl 语言哈希表

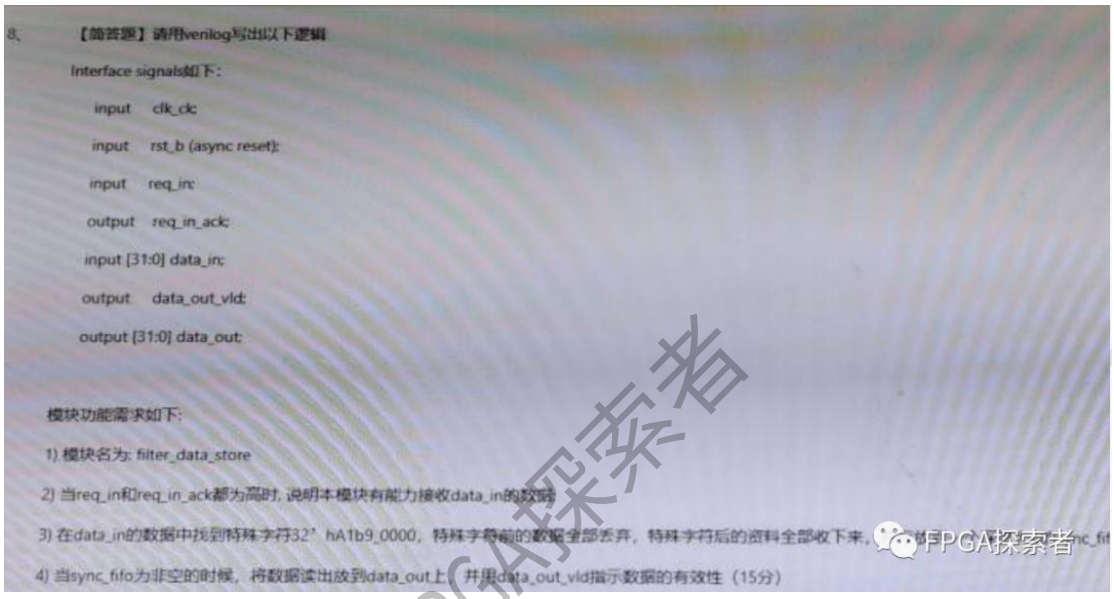
来看个联发科秋招的一个大题（3）——必考的 Perl 语言文件读写

来看个联发科秋招的一个大题（2）——必考的 Perl 语言正则匹配和文件读写

8. Verilog 编程

请用 Verilog 写出以下逻辑。

在 data_in 的数据中找到特殊字符 32'hA1b9_0000，特殊字符前的数据全部丢掉，特殊字符后的数据全部收下来，并且保存在 sync_fifo 中，当 sync_fifo 为非空的时候，将数据读出放到 data_out 上，并用 data_out_vld 指示数据的有效性。



分析：

问题涉及：

（1）输入检测；

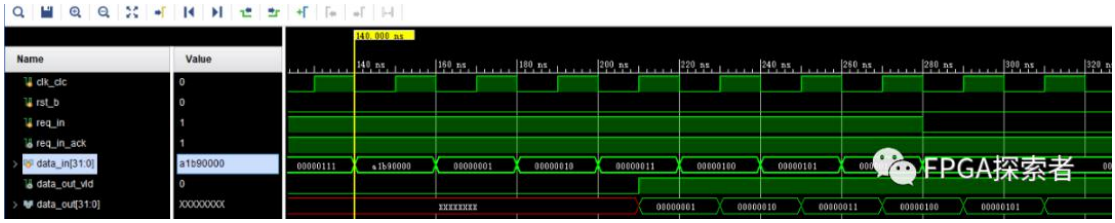
（2）同步 FIFO；

```
/*****************************************
// 公众号：FPGA 探索者
*****************************************/
module filter_data_store(
    clk_clc, rst_b, req_in, req_in_ack,
    data_in, data_out_vld, data_out
    );

input               clk_clc;
input               rst_b;              // async reset
input               req_in;
output              req_in_ack;
input   [31:0]  data_in;
output              data_out_vld;
```

```
output    [31:0]    data_out;

reg req_in_ack_reg;
assign req_in_ack = req_in_ack_reg;
wire in_valid;
assign in_valid = req_in & req_in_ack;
reg in_valid_reg;

reg [31:0] data_in_reg;
reg data_start = 1'b0;
always @ (posedge clk_clc or posedge rst_b)
begin
      if( rst_b ) begin
            data_start <= 1'b0;
            req_in_ack_reg <= 1'b0;
      end
      else begin
            in_valid_reg <= in_valid;
            data_in_reg <= data_in;
            req_in_ack_reg <= 1'b1;
            if( in_valid ) begin
                  if( data_in == 32'hA1b9_0000 ) begin
                        data_start <= 1'b1;
                  end
            end
            else begin
                  data_start <= data_start;
            end
      end
end

reg data_start2;
always @ (posedge clk_clc)
begin
      data_start2 <= data_start;
end

// 公众号：FPGA 探索者
reg [31:0] fifo_ram [3:0];    // reg [data_width-1:0] fifo_ram [data_depth-1:0]
reg [2:0] count;
wire full;
wire empty;
reg [1:0] wr_ptr;
reg [1:0] rd_ptr;
```

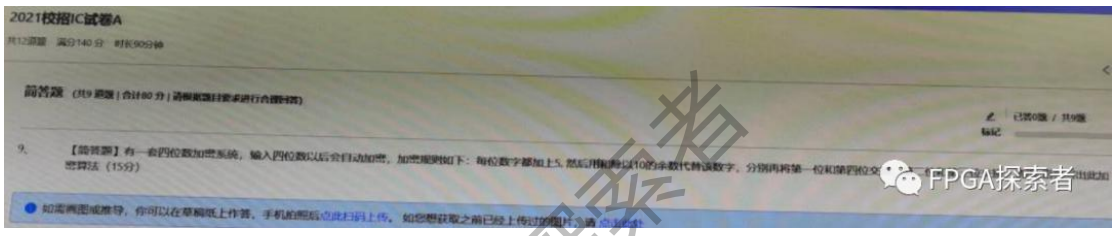
```
assign full = (count == 3'd4);
assign empty = (count == 3'd0);

reg [31:0]    data_out_reg;
reg           data_out_vld_reg;
always @ (posedge clk_clc or posedge rst_b)
begin
    if( rst_b ) begin
        count <= 3'b0;
        wr_ptr <= 0;
        rd_ptr <= 0;
        data_out_vld_reg <= 1'b0;
    end
    else begin
        if( !empty ) begin                                  // 不空
            if( data_start2 & in_valid_reg) begin           // 读+写
                fifo_ram[wr_ptr] <= data_in_reg;
                data_out_reg <= fifo_ram[rd_ptr];
                data_out_vld_reg <= 1'b1;
                wr_ptr <= wr_ptr + 1'b1;
                rd_ptr <= rd_ptr + 1'b1;
            end
            else begin                                      // 读
                data_out_reg <= fifo_ram[rd_ptr];
                data_out_vld_reg <= 1'b1;
                rd_ptr <= rd_ptr + 1'b1;
                if( count >= 1)
                    count <= count - 1;
                else
                    count <= 0;
            end
        end
        else begin                                          // 空，写
            data_out_vld_reg <= 1'b0;
            if( data_start2 & in_valid_reg) begin
                fifo_ram[wr_ptr] <= data_in_reg;
                wr_ptr <= wr_ptr + 1'b1;
                if(count >= 3'd4)
                    count <= 3'd4;
                else
                    count <= count + 1'b1;
            end
        end
```

```
    end
end
assign data_out = data_out_reg;
assign data_out_vld = data_out_vld_reg;
endmodule
```



9. C 语言编程

有一套四位数加密系统，输入四位数以后会自动加密。加密规则如下：每位数字都加上 5，然后用和除以 10 的余数代替该数字，分别再将第一位和第四位交换、第二位和第三位交换，请用 C 语言写出此加密算法。



分析要点

1. 准备使用多次循环输入，while 循环，并且指定一个输入退出机制，用 break 退出外部的 while；
2. sacnf 输入时，一定注意对于 int、unsigned int 等类型的数据需要使用 & 取地址符号，而对于字符数组或者字符串是不需要用 &，直接给变量名；

```
scanf("%d",&data_in); // 注意 & 取地址符号 scanf("%s",data_in); // 对于字符串，字符数组，变量名就是数组首地址
```

3. 对一个四位数取每一位的数据，应该依次取模后取除法；

4. 输出要按指定格式输出，考虑输出结果是 0 或者 12 等不足 4 位数的情况，需要在前面补零，即输出 0000、0012 等；

```
printf("data_out = %04d\n\n",data_out);
```

注意 printf 输出和 scanf 输入的不同，printf 直接是变量名，scanf 是 &+变量名；

编程结果

```
#include <stdio.h>
int main()
{
```

```
    int data_in;
    char A, B, C, D;
    char A1, B1, C1, D1;
    int data_out;

    while(1) {
        printf("please input data_in: ");
        scanf("%d",&data_in);

        // 输入 11111 表示退出
        if( data_in == 11111 ) {
            printf("Exit\n");
            break;   // break 退出了 while 循环
        }

        // 取出 四位数
        A = data_in / 1000;
        B = (data_in % 1000) / 100;
        C = (data_in % 100) / 10;
        D = (data_in % 10);

        // 加 5 求 余数
        A1 = (A + 5) % 10;
        B1 = (B + 5) % 10;
        C1 = (C + 5) % 10;
        D1 = (D + 5) % 10;

        // 位交换 + 拼接
        data_out = D1*1000 + C1*100 + B1*10 + A1;
        // 指定格式输出，输出 4 位，不够 4 位的前面补零到 4 位
        // 比如 0，指定格式输出 0000
        printf("data_out = %04d\n\n",data_out);
    }
    return 0;
}
```



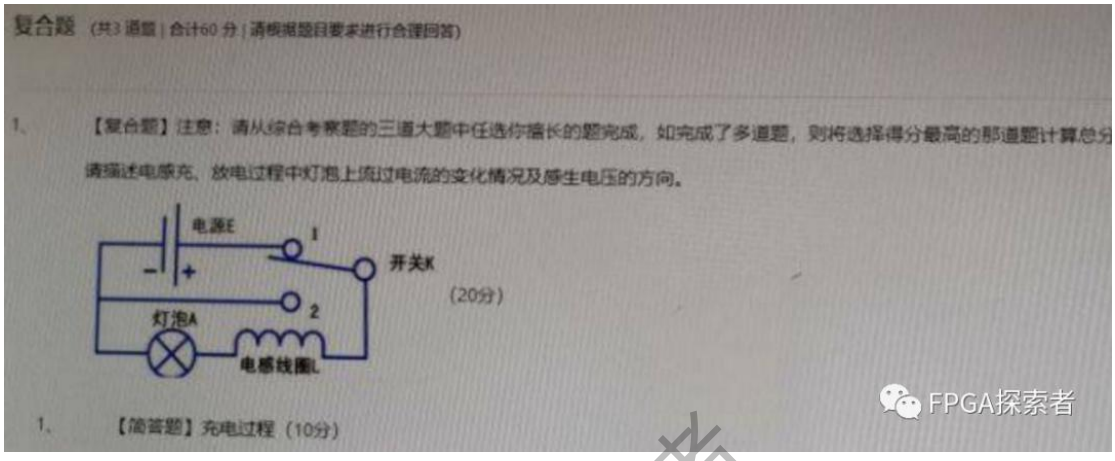
```
please input data_in: 0000
data_out = 5555

please input data_in: 1234
data_out = 9876

please input data_in: 5555
data_out = 0000

please input data_in: 2356
data_out = 1087

please input data_in: 7895
data_out = 0432

please input data_in: 11111
Exit
```

复合题

清楚综合考察的三道大题中任选你擅长的题完成，如完成了多道题，则将选择得分最高的那道题计算总分。

1. 电感

请描述电感充、放电过程中灯泡上流过的电流的变化情况及感生电压的方向。



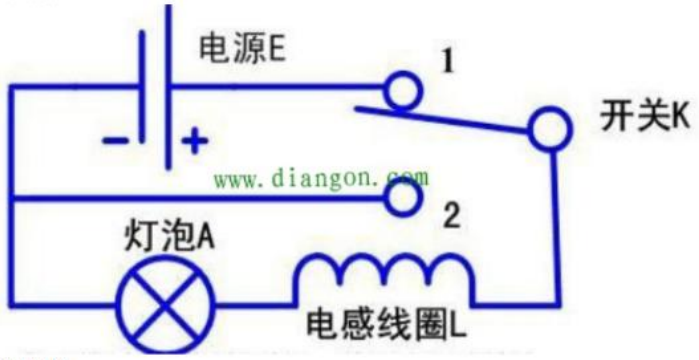
电感充放电的原理

2019-11-6 09:49 | 编辑:电工学习网 | 查看: 7864 | 评论: 1

电感这个元件在电子电路中是经常见到的，我们炒菜用的电磁炉里面有线圈盘它是特制的电感、电源变压器、电流互感器以及扼流圈都是电感。它在电路中一般起到滤波、扼流、调谐、延时、耦合、补偿等很多作用，今天我们来说说电感是如何进行充放电的。

电感的充电原理

为了能够清楚表述充电的原理，我们可以用下面的电路模型来进行说明问题。当我们把开关拨到1的位置的时候，由于电感的自感应原理，会建立一个左正右负的感应电动势来阻碍电源对线圈的充电电流，此时电感线圈L里的电流会慢慢增大，与电感线圈的灯泡此时的亮度会慢慢变亮。从这个渐亮的过程我们可以推测到电感总是阻碍自身电流的变化的。



电感的放电原理

当我们把开关拨到2的位置时候会发现灯泡A并不会立即熄灭，而是慢慢熄灭的。这时候电感线圈相当于一个电动势，其电感的左边是正极右边是负极。电感储存的电能逐渐被灯泡消耗完。所以我们看到灯的亮度由亮慢慢变暗最后熄灭这样一个渐进的过程。从这个电感的充电放电过程可以看出，电感中的电流是不能突变的，因此我们常称电感元件是"惯性元件"。

2. FIR 滤波器

4 阶 FIR，单位冲激响应 h[n] =[0.4922， 0.9297， 0.9297， 0.4922]，

（1）群时延 Group Delay

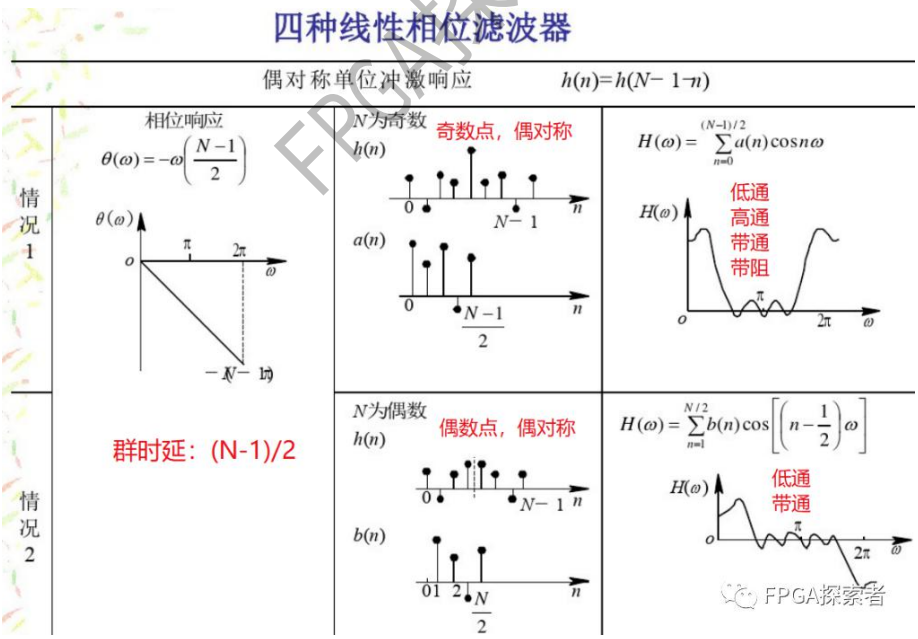
（2）叙述 FIR 滤波器相比 IIR 滤波器的特点

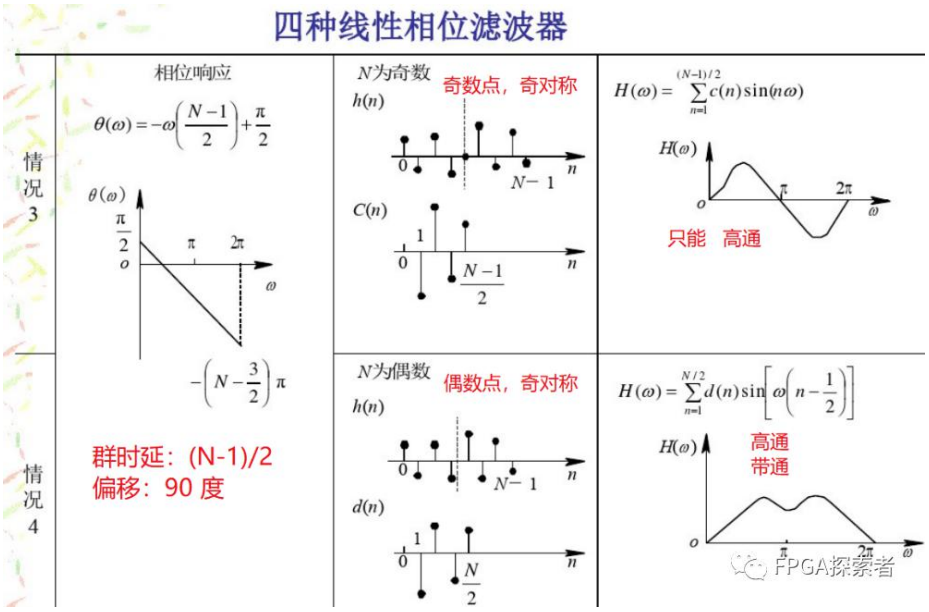
（3）不知道；

（4）请用乘法器、加法器、寄存器画出 FIR 的直接型和转置型结构框图。

N 点 FIR 滤波器的群时延是 (N-1)/2。

群时延是相位特性函数 f(w) 对相位 w **求导再取负**，而 **FIR 具有线性相位特性**，即 f(w) 是关于 w 的线性函数，即 f(w) = aw + b 形式，导数就是斜率 a，取负数为 -a。



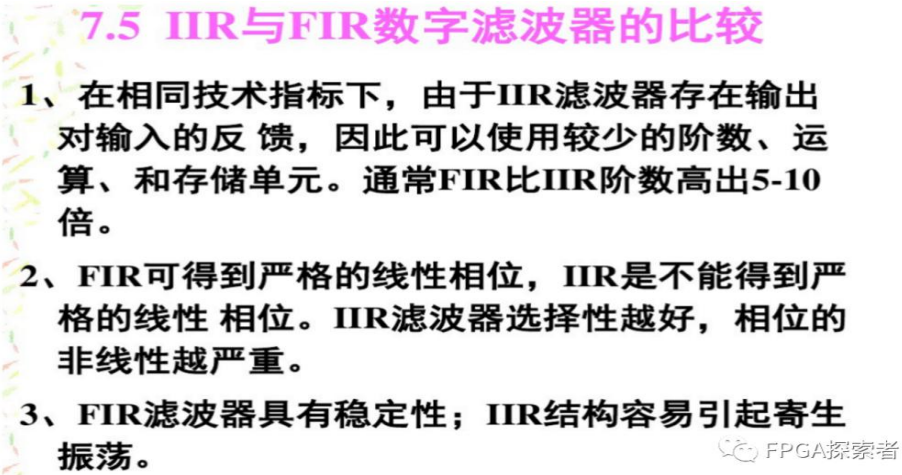


（a）FIR 稳定，零点均在单位圆内，没有极点（零极点相消），系统稳定，IIR 零极点可能在单位圆外，不稳定，容易振荡，在实际的 FPGA 实现中，有可能因为有限字长效应和量化误差导致单位圆内的零极点偏移到单位圆外；

（b）**FIR 具有严格的线性相位，群时延固定**，IIR 没有线性相位，其选择性越好，相位的非线性越严重；

（c）实现相同滤波性能的条件下，**FIR 需要更多的阶数**，一般 FIR 比 IIR 阶数多 5 - 10 倍；

（d）FIR 具有有限长冲击响应（Finite Impulse Response），**可以使用 FFT 进行快速计算**，IIR 滤波器是无限冲激响应（Infinite），不能使用 FFT 进行快速计算；



如下所示为 FIR 时域表达式，信号 x(n)，滤波器时域冲激响应 h(n)，时域卷积得到滤波后的数据 y(n)。

卷积：乘累加操作。

$$y(n)=x(n)*h(n)$$

如下所示为直接型和转置型结构，将其中的 z^(-1) 使用 reg 寄存器代替即可，根据题意，画出题目中给出的 4 阶 FIR 滤波器。

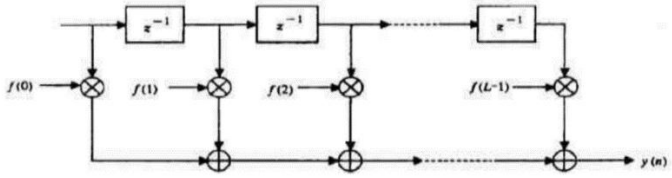


图 3-3 直接型的 FIR 滤波器

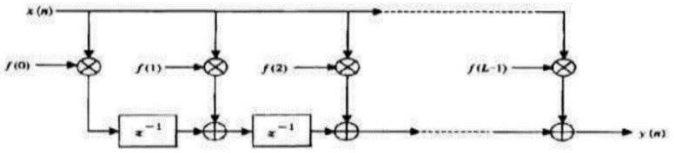
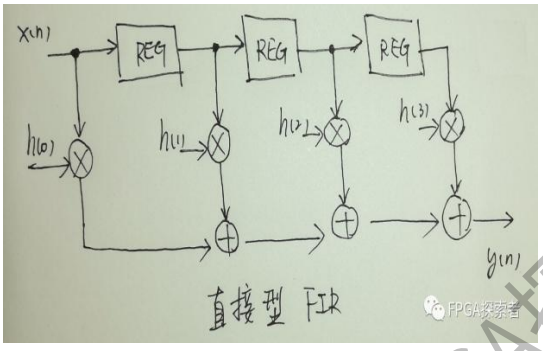
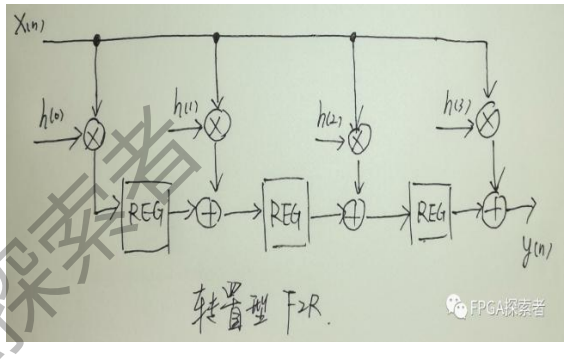


图 3-4 转置式结构的 FIR 滤波器





FIR 滤波器相关文章：

matlab 与 FPGA 数字滤波器设计（6）—— Vivado 中使用 Verilog 实现并行 FIR 滤波器/截位操作

matlab 与 FPGA 数字滤波器设计（5）—— Verilog 串行 FIR 滤波器

matlab 与 FPGA 数字滤波器设计（4）—— Vivado DDS 与 FIR IP 核设计 FIR 数字滤波器系统

matlab 与 FPGA 数字滤波器设计（3）—— Matlab 与 Vivado 联合仿真 FIR 滤波器

matlab 与 FPGA 数字滤波器设计（2）——Vivado 调用 IP 核设计 FIR 滤波器

matlab 与 FPGA 数字滤波器设计（1）——通过 matlab 的 fdatool 工具箱设计 FIR 数字滤波器

3. 信号与系统、傅里叶变换

占空比为 50% 的理想方波的偶次谐波的幅度为（0），奇次谐波的幅度为（2/(pi*N)）。

所以复指数函数形式的傅里叶级数为

$$x(t)=\sum_{n=-\infty}^{\infty}c_n e^{jn\omega_0 t}=-j\frac{A}{\pi}\sum_{n=-\infty}^{\infty}\frac{1}{n}(1-\cos n\pi)e^{jn\omega_0 t},\quad n=0,\ \pm1,\ \pm2,\ \pm3,\ \cdots。$$

$$\begin{cases}c_{nI}=-\dfrac{A}{n\pi}(1-\cos n\pi) & (n=0,\ \pm1,\ \pm2,\ \pm3,\ \cdots)\\ c_{nR}=0\end{cases}$$

$$|c_n|=\sqrt{{c_{nR}}^2+{c_{nI}}^2}=\left|\frac{A}{n\pi}(1-\cos n\pi)\right|=\begin{cases}\left|\dfrac{2A}{n\pi}\right| & n=\pm1,\pm3,\pm,\cdots \quad \text{奇次谐波}\\ 0 & n=0,\pm2,\pm4,\pm6,\cdots \quad \text{偶次谐波}\end{cases}$$

$$|c_n|=\sqrt{{c_{nR}}^2+{c_{nI}}^2}=\left|\frac{A}{n\pi}(1-\cos n\pi)\right|=\begin{cases}\left|\dfrac{2A}{n\pi}\right| & n=\pm1,\pm3,\pm,\cdots \quad \text{奇次谐波幅}\\ 0 & n=0,\pm2,\pm4,\pm6,\cdots \quad \text{偶次谐波幅}\end{cases}$$

$$\varphi_n=\arctan\frac{c_{nI}}{c_{nR}}=\begin{cases}-\dfrac{\pi}{2} & n=+1,+3,+5,\cdots \quad \text{奇次谐波正频率部分}\\ \dfrac{\pi}{2} & n=-1,-3,-5,\cdots \quad \text{奇次谐波负频率部分}\\ 0 & n=0,\pm2,\pm4,\pm6,\cdots\end{cases}$$

PDF 版在【FPGA 探索者】公众号回复【联发科笔试题】获取。

2022 届 FPGA/数字 IC 交流：

【往期精彩】

2022 届提前批汇总

华为 2022 届江苏山东高校顶尖人才计划
华为 2022 届校园招聘成渝地区-精英计划
晶晨半导体 2022 年校园招聘预热——电子科大校友交流
紫光展锐 2022 校招面试直通卡——5 月 12 日开始启动岗位推介会

最新面经汇总

乐鑫科技实习面试及基础问题解答
字节跳动实习面试及基础问题解答
2021 英伟达暑期实习面经（芯片设计前端/DFT）

笔试面试

【华为 2021 秋招】FPGA 逻辑笔试解析-2【修改】
【华为 2021 秋招】FPGA 逻辑笔试解析【独家】【数字 IC】【FPGA 逻辑】【2021 届秋招】
FPGA、数字 IC 系列（1）——乐鑫科技 2021 数字 IC 提前批笔试（上）
FPGA、数字 IC 系列（1）——乐鑫科技 2021 数字 IC 提前批笔试（下）
2020 年大疆芯片开发笔试（二）【数据无损量化问题】【定点数】【无损定点化】
2020 年大疆芯片开发（一）【FPGA 资源】【存储器问题】【Source clock latency 约束】
FPGA/数字 IC 笔试题——序列检测（FSM 状态机）【状态机序列检测】
同步后的复位该当作同步复位还是异步复位？——Xilinx FPGA 异步复位同步释放
不得不读的 FPGA 设计白皮书——Xilinx FPGA 复位策略白皮书翻译（WP272）【FPGA 探索者】
Verilog 笔记——奇数分频和小数分频
FPGA、数字 IC 系列（2）——电子科大与北航部分 Verilog 题目与解析
扭环形计数器、环形计数器、m 序列线性反馈移位寄存器、ZC 序列
Verilog 的块语句 fork...join 和 begin...end
Verilog 中负数的 % 取余数运算、C 语言、Matlab 各自的取余数运算【%】【mod】【rem】
Verilog 中用于时序验证的系统任务[setup][hold][skew][width][recovery][removal]
存储器相关问题汇总【SRAM】【DRAM】【SDRAM】【Flash】【EPROM】【EEPROM】
FPGA 设计中的优化问题——【面积优化】【速度优化】【关键路径优化】【流水线】【寄存器配平】【资源共享】
CDC 跨时钟域处理及相应的时序约束【set_clock_groups】【set_max_delay】【FPGA 探索者】

实习信息

ADI 2021 暑期实习生招聘启动！
招聘丨 Qualcomm 2021 暑期实习生招募正式启动！速来~
英特尔 2021 年暑期实习
中兴通讯 2022 研发类实习招聘丨多岗位来袭！