FIFO 深度计算：

一个8bit宽的AFIFO，输入时钟为100MHz，输出时钟为95MHz，设一个package为4Kbit，且两个package之间的发送间距足够大。问AFIFO的深度。

因为这位网友可能只是简述，因此信息并不完整，我的个人理解是这样的场景，一个异步FIFO，读写频率不同，读写位宽相同。发送发一次Burst突发的数据量为4Kbit，即500Word，在两次Burst突发之间有足够的时间，因此我们只用考虑在发送方Burst发送数据的时间T内，如果接受方没法将数据全部接受，其余数据均可存在FIFO内且不溢出，那么在发送方停止Burst发送数据的时间段内，接收方就可以从容的从FIFO内读取数据。首先发送方Burst发送数据的时间段为 T = 500/100MHz，发送的数据量为 B_send = 500word，而在T这段时间内，接收方能够接受的数据量为B_rec = T*95MHz = 500 * 95 / 100 word = 475word，因此 B_remain = B_send - B_rec = 500 - 475 = 25 。那么FIFO的深度至少要大于等于25才行。

https://blog.csdn.net/times_poem/article/details/51917648

ISE 实现步骤：

ISE实现的步骤

在综合之后，我们开始启动FPGA在ISE中的实现过程，整个过程包括以下几个步骤：

1.Translate - 将输入的网表文件和约束文件整合后输出到一个Xilinx私有的通用数据库 文件（ Native Generic Database,NGD ）中。

2.MAP - 将设计映射到目标器件的资源上，可以选择在此阶段完成资源的布局。

3.Place and Route - 按照时序约束的要求，完成设计的布局布线。

4.Generate Programming File - 生成一个可下载到FPGA器件的bit流文件。

FPGA 速度等级：

转 FPGA的速度等级（speed grade）

2014年12月06日 18:22:39　宇宙379　阅读数：10263

1. 对于Xilinx的 CPLDs来说，值越小，速度越高；
2. 对于Xilinx FPGAs 来说，值越大，速度越高。

Each speed grade increment is ~15% faster than the one before it. So a -5 is 10% faster than a -4 speed grade.

For example, Virtex-4 speed grades are -10 (slowest), -11, and -12 (fatest)；
Virtex-5 spede grades are -1 (slowest), -2, and -3 (fastest)

器件的速度等级

关于器件速度等级的选型，一个基本的原则是：在满足应用需求的情况下，尽量选用速度等级低的器件。该选型原则有如下好处：

(1)由于传输线效应，速度等级高的器件更容易产生信号反射，设计要在信号的完整性上花更多的精力；

(2)速度等级高的器件一般用得比较少，价格经常是成倍增加，而且高速器件的供货渠道一般比较少，器件的订货周期一般都比较长，经常会延误产品的研发周期，降低产品的上市率。

时序约束主要包括周期约束，偏移约束，静态时序路径约束三种。通过附加时序约束可以综合布线工具调整映射和布局布线，使设计达到时序要求。

附加时序约束的一般策略是先附加全局约束，然后对快速和慢速例外路径附加专门约束。附加全局约束时，首先定义设计的所有时钟，对各时钟域内的同步元件进行分组，对分组附加周期约束，然后对 FPGA/CPLD 输入输出 PAD 附加偏移约束、对全组合逻辑的 PAD TO PAD 路径附加约束。附加专门约束时，首先约束分组之间的路径，然后约束快、慢速例外路径和多周期路径，以及其他特殊路径。

前端设计的主要流程

1、规格制定:芯片规格，也就像功能列表一样，是客户向芯片设计公司（称为 Fabless，无晶圆设计公司）提出的设计要求，包括芯片需要达到的具体功能和性能方面的要求。

2、详细设计:Fabless 根据客户提出的规格要求，拿出设计解决方案和具体实现架构，划分模块功能。

3、HDL 编码:使用硬件描述语言（VHDL，Verilog HDL，业界公司一般都是使用后者）将模块功能以代码来描述实现，也就是将实际的硬件电路功能通过 HDL 语言描述出来，形成 RTL（寄存器传输级）代码。

4、仿真验证:仿真验证就是检验编码设计的正确性，检验的标准就是第一步制定的规格。看设计是否精确地满足了规格中的所有要求。规格是设计正确与否的黄金标准，一切违反，不符合规格要求的，就需要重新修改设计和编码。设计和仿真验证是反复迭代的过程，直到验证结果显示完全符合规格标准。仿真验证工具 Mentor 公司的 Modelsim，Synopsys 的 VCS，还有 Cadence 的 NC-Verilog 均可以对 RTL 级的代码进行设计验证，该部分个人一般使用第一个-Modelsim。该部分称为前仿真，接下来逻辑部分综合之后再一次进行的仿真可称为后仿真。

5、逻辑综合－－Design Compiler:仿真验证通过，进行逻辑综合。逻辑综合的结果就是把设计实现的 HDL 代码翻译成门级网表 netlist。综合需要设定约束条件，就是你希望综合出来的电路在面积，时序等目标参数上达到的标准。逻辑综合需要基于特定的综合库，不同的库中，门电路基本标准单元（standard cell）的面积，时序参数是不一样的。所以，选用的综合库不一样，综合出来的电路在时序，面积上是有差异的。一般来说，综合完成后需要再次做仿真验证（这个也称为后仿真，之前的称为前仿真）逻辑综合工具 Synopsys 的 Design Compiler，仿真工具选择上面的三种仿真工具均可。

6、STA:Static Timing Analysis（STA），静态时序分析，这也属于验证范畴，它主要是在时序上对电路进行验证，检查电路是否存在建立时间（setup time）和保持时间（hold time）的违例（violation）。这个是数字电路基础知识，一个寄存器出现这两个时序违例时，是没有办法正确采样数据和输出数据的，所以以寄存器为基础的数字芯片功能肯定会出现问题。STA 工具有 Synopsys 的 Prime Time。

7、形式验证:这也是验证范畴，它是从功能上（STA 是时序上）对综合后的网表进行验证。常用的就是等价性检查方法，以功能验证后的 HDL 设计为参考，对比综合后的网表功能，他们是否在功能上存在等价性。这样做是为了保证在逻辑综合过程中没有改变原先 HDL 描述的电路功能。形式验证工具有 Synopsys 的 Formality。前端设计的流程暂时写到这里。从设计程度上来讲，前端设计的结果就是得到了芯片的门级网表电路。

Backend design flow 后端设计流程：

1、DFT:Design ForTest，可测性设计。芯片内部往往都自带测试电路，DFT 的目的就是在设计的时候就考虑将来的测试。DFT 的常见方法就是，在设计中插入扫描链，将非扫描单元（如寄存器）变为扫描单元。关于 DFT，有些书上有详细介绍，对照图片就好理解一点。DFT 工具 Synopsys 的 DFT Compiler

2、布局规划(FloorPlan)：布局规划就是放置芯片的宏单元模块，在总体上确定各种功能电路的摆放位置，如 IP 模块，RAM，I/O 引脚等等。布局规划能直接影响芯片最终的面积。工具为 Synopsys 的 Astro

3、CTS：Clock Tree Synthesis，时钟树综合，简单点说就是时钟的布线。由于时钟信号在数字芯片的全局指挥作用，它的分布应该是对称式的连到各个寄存器单元，从而使时钟从同一个时钟源到达各个寄存器时，时钟延迟差异最小。这也是为什么时钟信号需要单独布线的原因。CTS 工具，Synopsys 的 Physical Compiler

4、布线(Place & Route):这里的布线就是普通信号布线了，包括各种标准单元（基本逻辑门电路）之间的走线。比如我们平常听到的 0.13um 工艺，或者说 90nm 工艺，实际上就是这里金属布线可以达到的最小宽度，从微观上看就是 MOS 管的沟道长度。工具 Synopsys 的 Astro

5、寄生参数提取:由于导线本身存在的电阻，相邻导线之间的互感,耦合电容在芯片内部会产生信号噪声，串扰和反射。这些效应会产生信号完整性问题，导致信号电压波动和变化，如果严重就会导致信号失真错误。提取寄生参数进行再次的分析验证，分析信号完整性问题是非常重要的。工具 Synopsys 的 Star-RCXT

6、版图物理验证:对完成布线的物理版图进行功能和时序上的验证，验证项目很多，如 LVS（Layout Vs Schematic）验证，简单说，就是版图与逻辑综合后的门级电路图的对比验证；DRC（Design Rule Checking）：设计规则检查，检查连线间距，连线宽度等是否满足工艺要求，ERC（Electrical Rule Checking）：电气规则检查，检查短路和开路等电气规则违例；等等。工具为 Synopsys 的 Hercules 实际的后端流程还包括电路功耗分析，以及随着制造工艺不断进步产生的 DFM（可制造性设计）问题，在此不说了。物理版图验证完成也就是整个芯片设计阶段完成，下面的就是芯片制造了。物理版图以 GDSII 的文件格式交给芯片代工厂（称为 Foundry）在晶圆硅片上做出实际的电路，再进行封装和测试，就得到了我们实际看见的芯片

1、数字电路设计的核心是逻辑设计。数字电路的逻辑值只有'1'和'0'，表征的是模拟 电压或电流的离散值，一般'1'代表高电平，'0'代表低电平。

2、当前的数字电路的电平标准常见的有：TTL、CMOS、LVTTL、LVCMOS、 ECL、PECL、

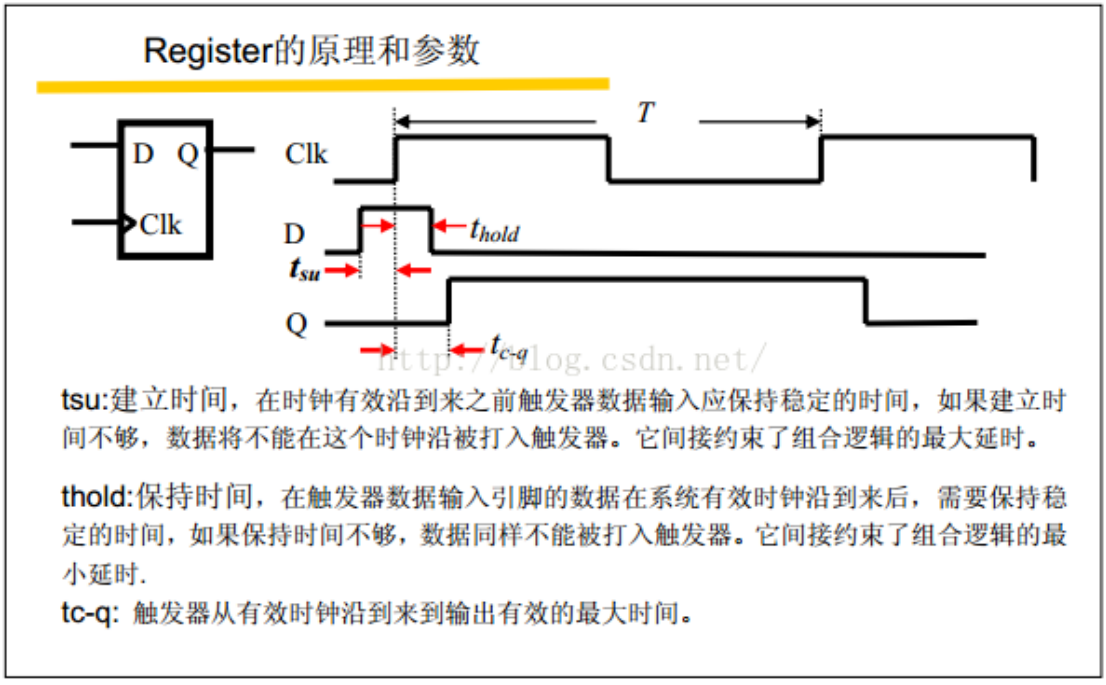
LVDS、HSTL、SSTL 等。

3、数字电路设计大致可分为组合逻辑电路和时序逻辑电路。

组合逻辑电 路的输出仅与当前的输入有关，而时序逻辑电路的输出不但与输入有关，还和系统上一个 状态有关。

组合逻辑电路由任意数目的逻辑门电路组成，一般包括与门、或门、异或门、与非门、或非门等。

4、时序逻辑电路由时钟的上升沿或下降沿驱动工作，其实真正被时钟沿驱动的是电路中的触发器（Register），也称为寄存器。触发器的工作原理和参数如下图：



5、竞争与冒险

竞争：当一个逻辑门的输入有两个或两个以上的变量发生改变时，由于这些变量时经过不同路径产生的，是它们状态改变的时刻有先有后，这种时差引起的现象称为竞争（Race），

竞争的结果将很可能导致冒险（Hazard）发生（例如产生毛刺）。

组合逻辑电路的冒险仅在信号状态改变的时刻出现毛刺，这种冒险是过渡性的，它不会是状态值偏离正常值。但在时序电路中，冒险是本质的，可导致电路输出值永远偏离正常值或发生震荡。

避免冒险的最简单的方法是同一时刻只允许单个输入变量发生变化，或使用寄存器采样的办法。

6、毛刺的产生

信号在器件中传输的时候，所需要的时间是不能精确估计的，当多路信号同时发生跳变的瞬间，就产生了“竞争冒险”。

这是，往往会出现一些不确定的尖峰信号，即“毛刺”。

7、毛刺的危害：

毛刺是数字电路设计中的棘手问题，它的出现会影响电路工作的稳定性、可靠性，严重时会导致整个数字系统的误动作和逻辑紊乱。

8、毛刺的消除

（1）输出加 D 触发器

原理是用一个 D 触发器去读带毛刺的信号，利用 D 触发器对输入信号的毛刺不敏感的

特点，去除信号中的毛刺。

（2）信号同步法

设计数字电路的时候采用同步电路可以大大减少毛刺。做到真正的“同步”就是去除毛刺信号的关键问题。所以同步的关键就是保证在时钟的跳变沿读取的数据时稳定的数据胃不是毛刺的数据。以下为两种具体的信号同步方法：

1）信号延时同步法

原理就是在两级信号传递的过程中加一个延时环节，从而保证在下一 个模块中读取到的数据是稳定后的数据，即不包含毛刺信号

2）状态机控制

由状态机在特定的时刻分别发出控制特定模块的时钟信号或者模块使能信号，状态机的循环控制就可以使得整个系统协调运作，同时减少毛刺信号。

（3）格雷码计数器

格雷码计数器的输出每次只有一位跳变

9、同步电路设计

同步电路设计是指所有电路在同一个公共时钟的上升沿或下降沿的触发下同步的工作。

同步电路的设计准则：

（1）尽可能在设计中使用同一时钟，时钟走全局时钟网络。走全局时钟网络的时钟是最简单、最可预测的时钟，并保证 Clocke skew 可以小到忽略的地步

（2）避免使用混合时钟沿采样数据。

（3）尽量少在模块内部使用计数器分频所产生的时钟。计数器分频时钟需完成的逻辑功能完全可由 PLL 锁相环或时钟使能电路替代。计数器分频时钟的缺点是使得系统内时钟不可控，并产生较大的 Clock skew，还使静态时序分析变得复杂。

（4）避免使用门控时钟。因为经组合逻辑产生的门控时钟极可能产生毛刺，且组合逻辑电路的 Jitter 和 Skew 比较大。

（5）当整个电路需要多个时钟来实现，则可以将整个电路分成若干局部同步电路（尽可能以同一个时钟为一个模块），局部同步电路之间接口当作异步接口考虑，而且每个时钟信 号的时钟偏差（△T）要严格控制。

（6）电路的实际最高工作频率不应大于理论最高工作频率，留有设计余量，保证芯片可靠工作。

（7）电路中所有寄存器、状态机在系统被 reset 复位时应处在一个已知的状态。

10、时钟的设计讨论

时钟类型包括: 全局时钟、内部逻辑时钟和门控时钟。

（1）全局时钟

全局时钟即同步时钟，它通过 FPGA 芯片内的全局时钟布线网络或区域时钟网络来驱动。

全局时钟的设计方法：

1）由 PLL 锁相环来产生全局时钟。

2）将 FPGA 芯片内部逻辑产生的时钟分配至全局时钟布线逻辑

3）将外部时钟通过专用的全局时钟输入引脚引入 FPGA。

（2）内部逻辑时钟（严格禁用）

内部逻辑时钟即指由芯片每部的组合逻辑或计数器分频产生的时钟。

（3）门控时钟（避免使用）

门控时钟最好只在顶层模块中出现，并将其分离到一个在顶层的独立模块中。

推荐的门控时钟设计如下，该设计一般不会产生毛刺和亚稳态的问题。

11、亚稳态

在同步电路或异步电路中，如果触发器的 setup 时间或 hold 时间不能得到满足，就可能产生亚稳态。此时触发器输出端 Q 在有效时钟沿之后比较长的一段时间处于不确定的状态，在这段时间里 Q 端将会产生毛刺并不断振荡、最终固定在某一电压值上，此电压值并不一定等于原来数据输入端 D 的值。这段时间称为决断时间 （resolution time）。经过决断时间之后，Q 端将稳定到 0 或 1 上，但是究竟是 0 还是 1， 这是随机的，与输入没有必然的关系。

亚稳态的危害：

（1）导致逻辑误判

（2）导致亚稳态的传播（严重情况下输出 0～1 之间的中间电压值还会使下一级 产生亚稳态）

12、对跨时域数据的处理

核心就是要保证下级时钟对上级数据采样的 setup 时间或 hold 时间满足要求，即尽量避免亚稳态的发生和传播。但是，只要系统中有异步元件，亚稳态就无法避免。推荐下面的方法来解决异步时钟域数据同步问题：

（1）用触发器打两拍

（2）异步 FIFO 或 DPRAM

因为异步 FIFO 或 DPRAM 使用格雷码计数器设计读写地址的指针，所以它可以很 好地避免亚稳态的发生。

（3）调整时钟相位（难度大，适用面有限）

ISERDES

读数据通路由读数据采集和再次采集阶段组成。这两个阶段都是在每个 Virtex-4 I/O 均有的内置 ISERDES 中实现。ISERDES 有三个时钟输入：CLK、OCLK 和 CLKDIV。读数据在 CLK (DQS) 域中采集，随后在 OCLK （FPGA 快速时钟）域中被再次采集，最后传输至 CLKDIV （FPGA 分频时钟）域以提供并行数据。

- CLK：使用 BUFIO 布线的读 DQS 提供 ISERDES 的 CLK 输入，如图 7 所示。
- OCLK：在硬件上，ISERDES 的 OCLK 输入连接到 OSERDES 的 CLK 输入。在本设计中，CLKfast_90 时钟为 ISERDES OCLK 输入和 OSERDES CLK 输入提供时钟。OCLK 使用的时钟相位由写数据所要求的相位所决定。
- CLKDIV：为了实现正确的功能，OCLK 和 CLKDIV 时钟输入必须相位对齐。因此，CLKDIV 输入使用与 CLKfast_90 相位对齐的 CLKdiv_90 或 CLKdiv_90 的反转信号提供。利用 CLKdiv_90 或其反转信号作为 ISERDES 最后一级的时钟，有助于限制将 DQS 和 DQ 与 CLKfast_90 域对齐所需的 tap 数量。

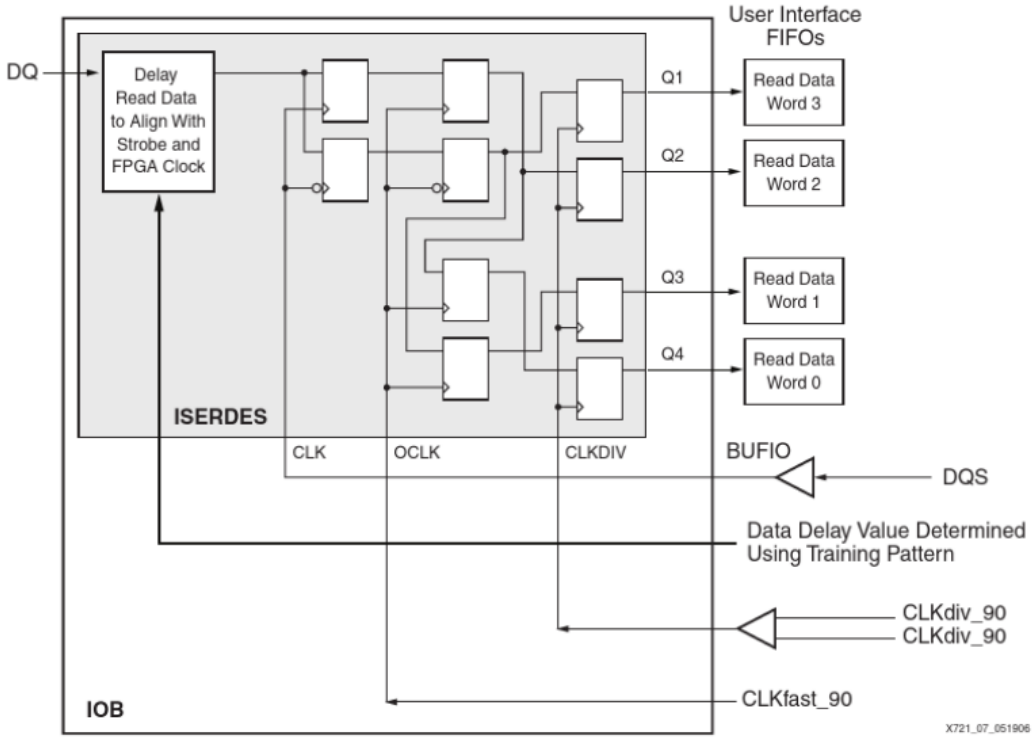
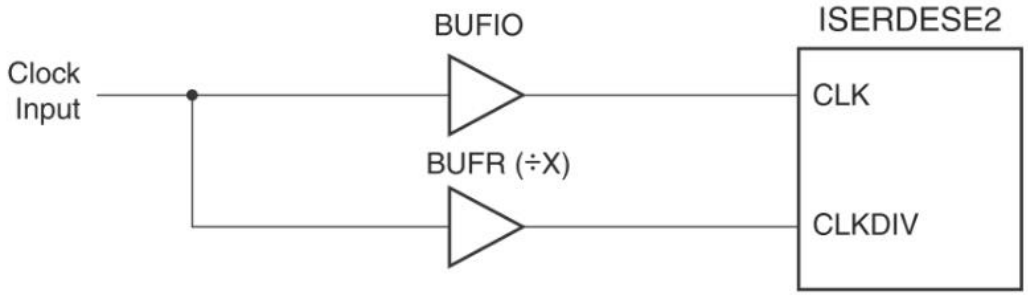


图 7: **使用 ISERDES 采集读数据**

https://blog.csdn.net/XiaoQingCaiGeGe/article/details/85255386



低功耗:

FPGA 主要的功耗是由静态功耗和动态功耗组成 ,降低 FPGA 的功耗就是降低静态功耗和动态功耗 。静态功耗除了与工艺有关外 , 与温度也有很大的关系 。一方面需要半导体公司采用先进的低功耗工艺来设计芯片 ,降低泄漏电流(即选择低功耗的器件);另一方面可以通过降低温度 、结构化的设计来降低静态功耗 。FPGA 动态功耗主要体现为存储器 、内部逻辑 、时钟 、I/O 消耗的功耗 。

①选择适当的 I/O 标准可以节省功耗 。I/O 功耗主要来自器件输出引脚连接的外部负

载电容 、阻抗模式输出

驱动电路以及外部匹配网络的充放电电流 。可选择较低的驱动强度或较低的电压标准 。当系统速度要求使用高功率 I/O 标准时 ,可设置缺省状态以降低功耗 。有的 I/O 标准需要使用上拉电阻才能正常工作 ,因此如果该 I/O 的缺省状态为高电平而不是低电平 ,就可以节省通过该终结电阻的直流功耗 。

②当总线上的数据与寄存器相关时 ,经常使用片选或时钟使能逻辑来控制寄存器的使能 ,尽早对该逻辑进行“数据使能” ,以阻止数据总线与时钟使能寄存器组合逻辑之间不必要的转换 。另一种选择是在电路板上 ,而不是芯片上 ,进行这种“数据使能” , 以尽可能减小处理器时钟周期 。也就是使用 CPLD 从处理器卸载简单任务 , 以便使其更长时间地处于待机模式 。

③设计中所有吸收功耗的信号当中 ,时钟是罪魁祸首 。虽然时钟可能运行在 100 M Hz , 但从该时钟派生出的信号却通常运行在主时钟频率 的较小分量(通常为 12 %～ 15%)。此外 , 时钟的扇出一般也比较高 。这两个因素显示 ,为了降低功耗 ,应当认真研究时钟 。首先 , 如果设计的某个部分可以处于非活动状态 ,则可以考虑禁止时钟树翻转 ,而不是使用时钟使能 。时钟使能将阻止寄存器不必要的翻转 ,但时钟树仍然会翻转 ,消耗功率。其次 , 隔离时钟以使用最少数量的信号区 。不使用的时钟树信号区不会翻转 ,从而减轻该时钟网络的负载 。合理的布局可以在不影响实际设计的情况下达到此目标 。

④根据预测的下一状态条件列举状态机 ,并选择常态之间转换位较少的状态值 ,这样就能尽可能减少状态机网络的转换量(频率)。确定常态转换和选择适当的状态值 ,是降低功耗且对设计影响较小的一种简单方法 。编码形式越简单(如 1 位有效编码或格雷码),使用的解码逻辑也会越少。

⑤要计算覆盖整个产品生命周期或预期电池工作时间内所有状态下的功耗 ,要考虑上电 、

待机 、空闲 、动态和断电等多种状态 ,要计算最坏情况下的静态功耗 。在所有降低功耗的措施中 ,选择合适的低功耗器件起决定性的作用 ,带来的效果是立竿见影的 ,而且无需花费大量的时间 、精力和成本采取额外的措施 。所以 ,选择一款低功耗的 FPGA 器件有助于提高产品性能 , 降低产品成本 ,提高产品的可靠性 。

SPI

CPHA=0，表示第一个边沿：

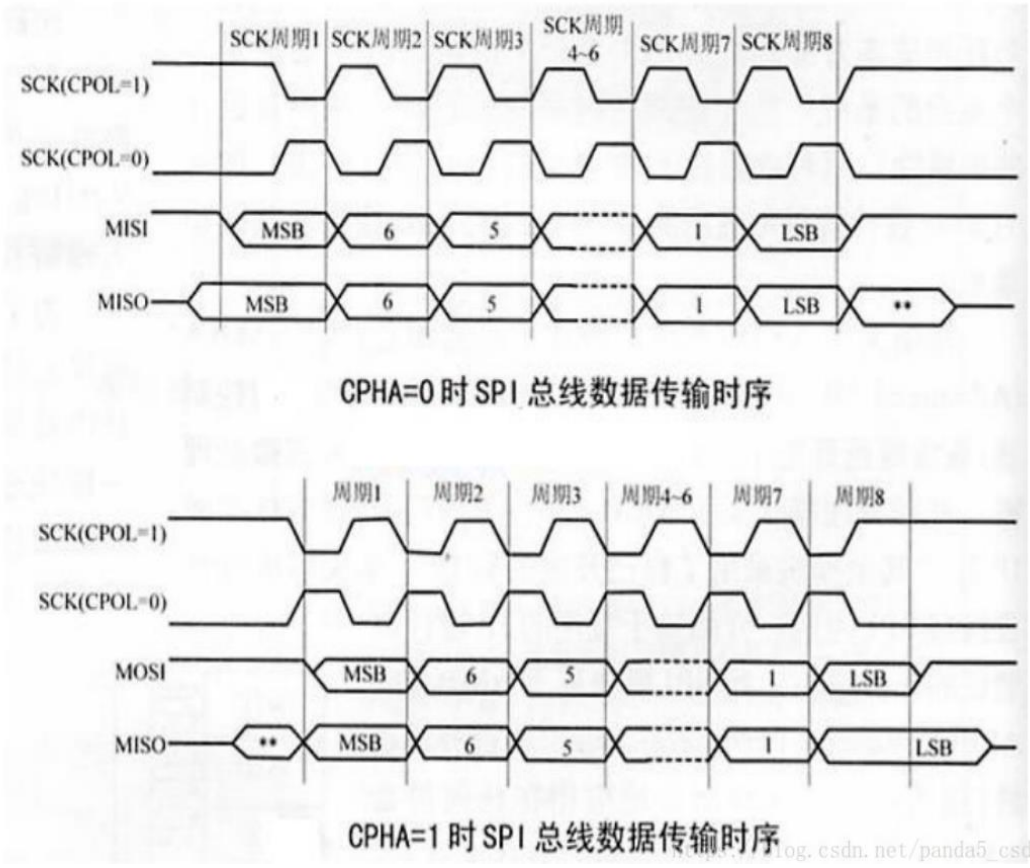
对于CPOL=0，idle时候的是低电平，第一个边沿就是从低变到高，所以是上升沿；

对于CPOL=1，idle时候的是高电平，第一个边沿就是从高变到低，所以是下降沿；

CPHA=1，表示第二个边沿：

对于CPOL=0，idle时候的是低电平，第二个边沿就是从高变到低，所以是下降沿；

对于CPOL=1，idle时候的是高电平，第一个边沿就是从低变到高，所以是上升沿；

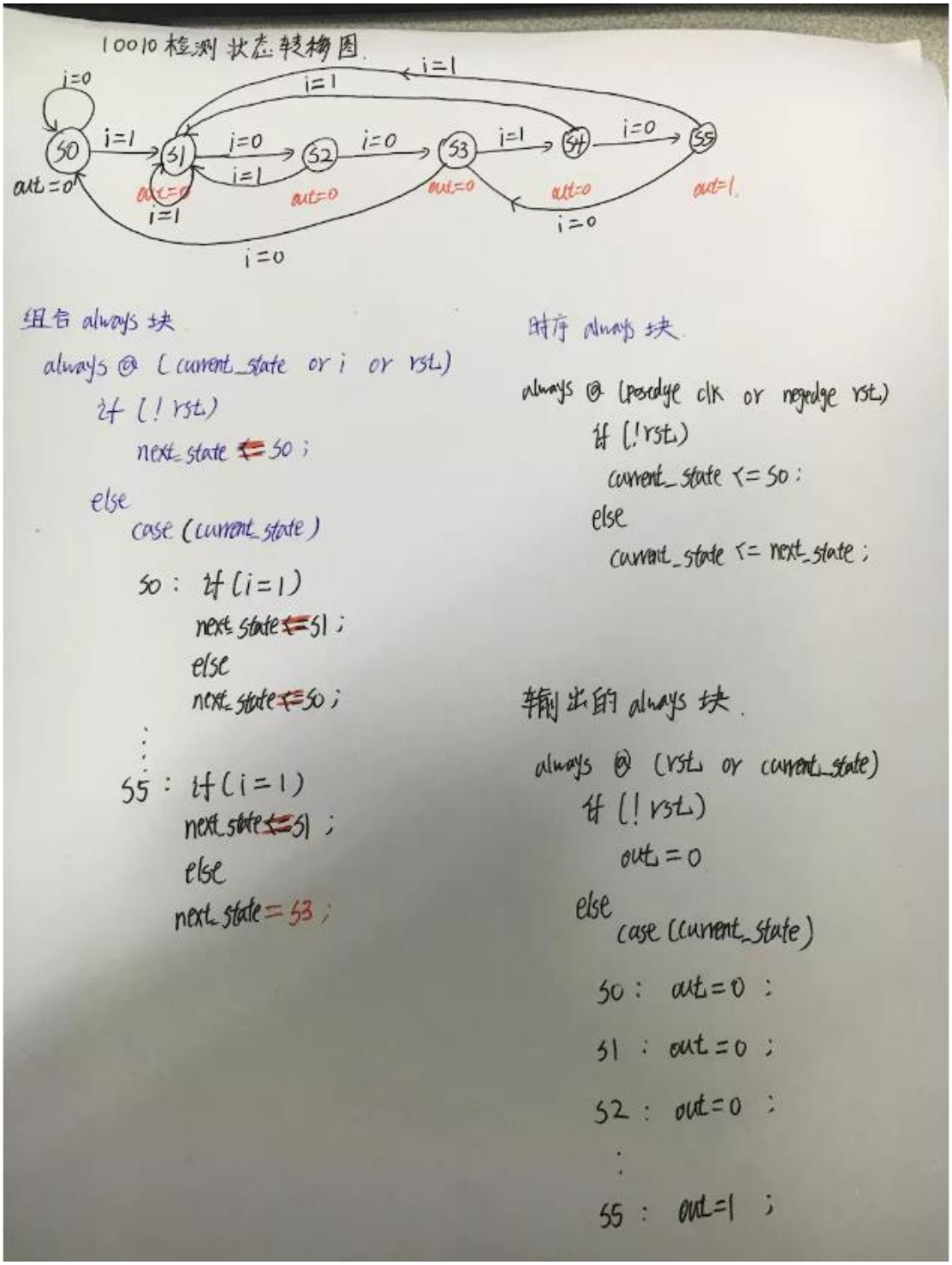


https://blog.csdn.net/panda5_csdn/article/details/81117970

异步 fifo

https://blog.csdn.net/wyj_2016/article/details/78469272

序列检测：


10010 检测 状态转移图
S0
S1
S2
S3
S4
S5
i=0
i=1
out=0
out=1
组合 always 块
always @ (current_state or i or rst_)
if (!rst_)
next_state <= S0;
else
case (current_state)
S0: if (i=1)
next_state <= S1;
else
next_state <= S0;
S5: if (i=1)
next_state <= S1;
else
next_state = S3;
时序 always 块
always @ (posedge clk or negedge rst_)
if (!rst_)
current_state <= S0;
else
current_state <= next_state;
输出的 always 块
always @ (rst_ or current_state)
if (!rst_)
out_ = 0
else
case (current_state)
S0: out_ = 0;
S1: out_ = 0;
S2: out_ = 0;
S5: out_ = 1;

ADC 复位电路:

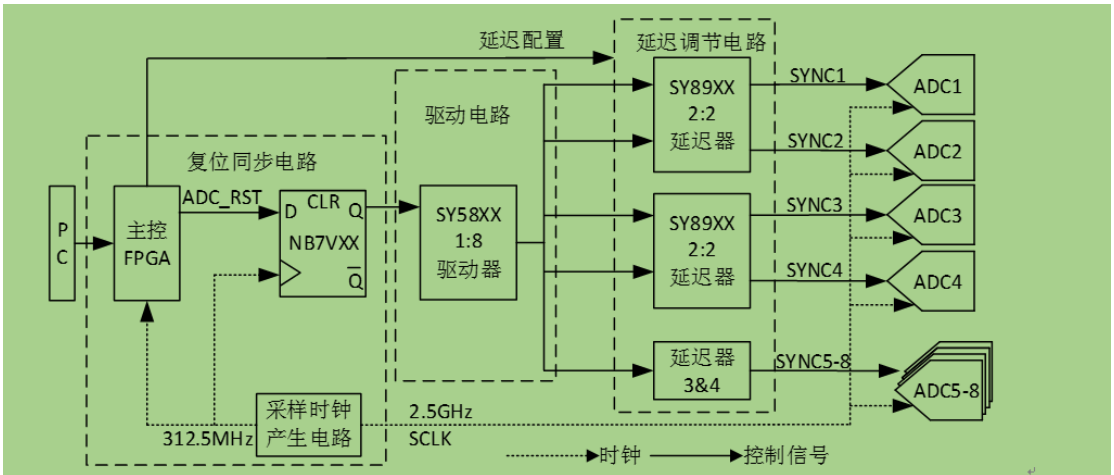


图 4-5 ADC 复位硬件原理图

复位电路硬件原理图如图 4-5 所示。由时钟产生电路产生频率为 312.5MHz 的时钟接入 D 触发器的 CLK 端口，FPGA 的复位差分信号接入数据输入端口(D)，数据输出端(Q)接到驱动芯片的输入端。当上位机发出复位指令，FPGA 内部利用 312.5MHz 的时钟产生一个脉宽为 5 个时钟周期的高脉冲信号传给 D 触发器，经过 D 触发器实现与采样时钟的源同步处理。经 D 触发器同步后的复位信号连接到驱动芯片的输入端，经过 1：8 驱动后驱动为 8 路差分复位信号，每两路复位信号为一组，传给延迟调节芯片的输入端，芯片延迟通过软件配置每一组复位信号的延迟后传给 ADC 的同步复位信号端口。

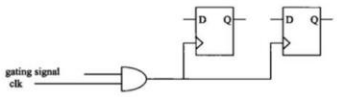
门控时钟：

5、门控时钟

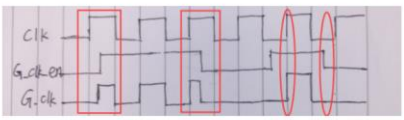
门控时钟可是低功耗设计的宠儿，关于门控时钟的资料也有一大堆，下面就来写写门控时钟吧。

门控时钟也就是在使能信号有效的时候，把时钟打开；使能信号无效的时候，时钟关闭。时钟关闭之后，它所驱动的寄存器就不会反正，因此也就降低了动态功耗。

门控时钟最开始的电路是：

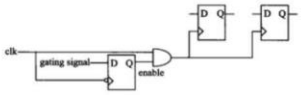


这种门控时钟bug多，我们先来看看这种电路的bug波形是怎么样的，也就是知道问题所在，好让我们改进：

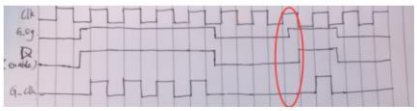


从波形图中可以看到，门控使能信号如果在时钟的高电平的时候开启或者关闭，就会导致产生的门控时钟高电平被截断，变成毛刺；门控使能信号对在时钟低电平时跳变对产生的门控时钟没有影响。因此我们的针对点就是高电平时的翻转。

因此我们就可以通过设置一种电路，让门控使能信号在通过这个逻辑电路之后，仅仅在时钟低电平的时候进行翻转，而在时钟高电平的时候，不能翻转也就是保持。从而我们就想到了低电平触发的锁存器，使能信号通过低电平的锁存器之后，如果使能信号在高电平跳变，锁存器的输出信号是不会改变的，电路图如下所示：



波形如下所示：



这里需要注意的是：

当门控使能信号是高电平有效的时候，也就是高电平打开门控时钟，低电平关闭门控时钟，那么就使用上面的电路，也就是：**低电平触发**的锁存器+**与**门。

当门控使能信号是低电平有效的时候，那么就要换成：**高电平触发**的锁存器+**或**门。

PS：当涉及毛刺的问题的时候，特别是由于使能信号与时钟而产生的毛刺，锁存器起很大的作用。

一般情况，在进行芯片设计的时候，我们不必自己设计门控时钟，大多是ASIC/SoC生产商都有对应的门控时钟单元。

FPGA 静态时序分析——IO 口时序

https://www.cnblogs.com/raymon-tec/p/5307557.html

Verilog 缩位与

&ain = ain[4] & ain[3] & ain[2] & ain[1] & ain[0] = 1 & 0 & 1 & 0 & 1 = 0.

部分逻辑电路设计：

https://www.cnblogs.com/ylsm-kb/p/9141295.html