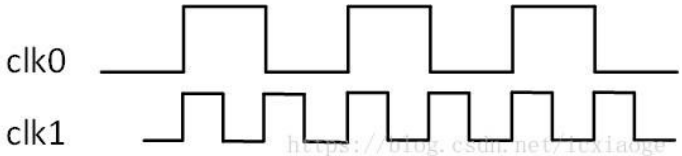
目录

Synchronous Clocks vs. Asynchronous Clocks

- **相位相同**: 时钟跳变沿相同



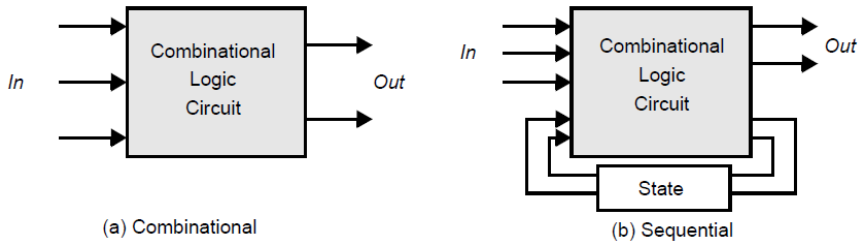
- **相位同步**: 相位差固定，时钟跳变沿保持固定时间差

同步时钟 synchronous clocks: fixed phase relation		
同相位时钟	**同源时钟**	**同时钟域时钟**
相位相同 频率可以不同	来自一个 PLL or DLL， 相位差固定，不要求相同，即相位同步	同源时钟 相位相同 频率相同
异步时钟 asynchronous clocks: from different clock resources (oscillators or quartz crystal)		
来自不同晶振 来自不同 PLL 即非同源时钟 相位不同，频率不同		

- ⇨ **跨时钟域处理 cross-clock domain**: 即异步时钟处理
- ⇨ **同源晶振只能产生同步时钟**

Sequential Logic (Circuit) vs. Combinational Logic (Circuit)

- **Combinational Logic**: Outputs of the circuit depend only on its current input signals.
- **Sequential Logic**: Outputs of the circuit depend on not only its current input data, but also its previous input signals (previous states).

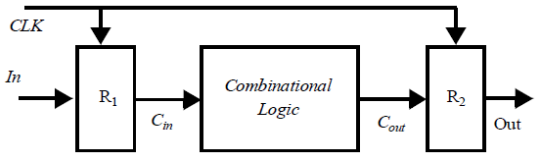


Synchronous Sequential Logic vs. Asynchronous Sequential Logic

- **Sequential Logic** = **Synchronous Sequential Logic** + **Asynchronous Sequential Logic**

Synchronous Sequential Logic 同步时序逻辑（同步设计）

- All memory states (clocked Flip-Flop, in some case gated Latch) will only be changed at the same rising or falling clock edge. That means changes to all signals in the circuit are triggered by the clock signal. This clock signal has same frequency and fixed phase offset with respect to the local clock (system clock).
- It exhibits **worst-case** behavior. (clock frequency is determined by delay of the longest path)



- **举例**: 带同步复位的触发器（clocked FF all signals）

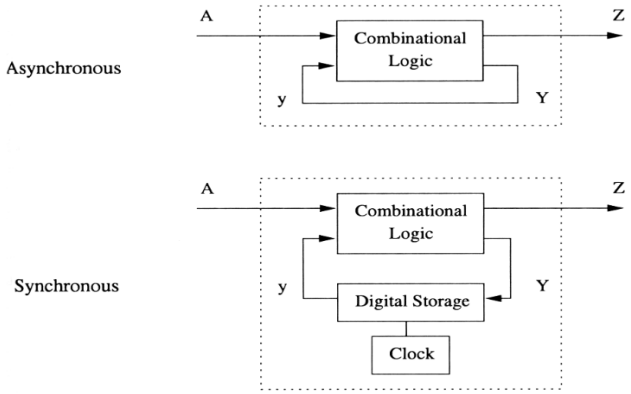
```
always @(posedge clk or posedge RST)
 if(RST)
   Q<=1'b0;
 else
   Q<=A|(B&C&D&E);
```

Properties:

- 只有时钟脉冲同时到达各记忆器件的时钟端时，电路状态改变
- 只有在前一个时钟脉冲引起的电路响应完全结束后，下一时钟脉冲才能到来
- 外部输入信号的变化，应当满足触发器正常工作所需的建立和保持时间（时序约束）

Asynchronous Sequential Logic: a.k.a. self-timed circuit 异步时序逻辑（异步设计）

- It's not synchronized by the local clock signal. Instead it's driven by the pulses of the inputs. The outputs of the circuit change directly in response to changes in inputs. Their memory elements are either un-clocked flip-flops or time-delay elements.
- It exhibits **average-case** behavior. (faster clock does faster computations while slower clock does slower computations).



- **举例**: 带异步复位的触发器（un-clocked FF for RST signal）

```
always @(posedge clk)
 if(RST)
   Q<=1'b0;
 else
   Q<=A|(B&C&D&E);
```

Advantages:

- **Faster**—average-case behavior.
- **Immunity to metastable behavior** (设计可靠性高)—simply delay the computation until stable state.
- **Modularity** (可移植性高)—异步电路只考虑 sequence of events 不考虑 Timing of events，可以直接互连。The designer does not have to worry about the delays incurred in individual modules or the delays inserted by connection wires.

- **Low Power**: In systems with a global clock (synchronous circuit), all of the latches and registers operate and consume dynamic energy during each clock pulse, in spite of the fact that many of those latches and registers might not have new data to store. Another power consumption comes from Hazards in synchronous circuits. Asynchronous circuits are Hazard-free and un-clocked.
- Freedom from Clock Skew (无时钟偏移问题)—Asynchronous circuit don't have clocks.

Synchronous Circuit vs. Asynchronous Circuit

Synchronous Circuit: 同步电路

- All signals are coordinated by a central clock (system clock).

Asynchronous Circuit: 异步电路

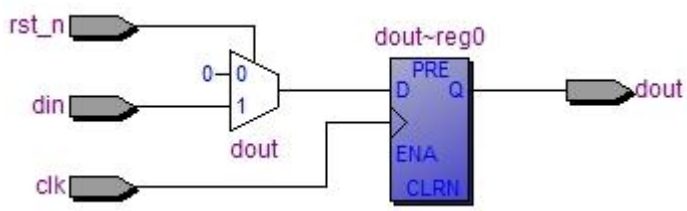
- Signals are coordinated by different clocks.

Synchronous Reset vs. Asynchronous Reset

	同步复位 sync	**异步复位 async**
特点	复位信号只有在时钟上升沿到来时才能有效。	无论时钟沿是否到来，只要复位信号有效，就进行复位。
Verilog 描述	always@(posedge CLK)	always@(posedge CLK or negedge Rst_n)
优点	1) 利于仿真器仿真。 2) 因为只有在时钟有效电平到来时才有效，所以可以滤除高于时钟频率的毛刺。 3) 可以使所设计的系统成为 100%的同步时序电路，有利于时序分析。	1) 设计相对简单。 2) 因为大多数目标器件库的 D-FF 都有异步复位端口，因此采用异步复位可以节省资源。 3) 异步复位信号识别方便，而且可以很方便的使用 FPGA 的全局复位端口 GSR。
缺点	1) 复位信号的有效时长必须大于时钟周期，才能真正被系统识别并完成复位任务。同时还要考虑，诸如：clk skew, 组合逻辑路径延时, 复位延时等因素。 2) 由于大多数的逻辑器件的目标库内的 D-FF 都只有异步复位端口，所以，倘若采用同步复位的话，综合器就会在寄存器的数据输入端口插入组合逻辑，这样就会耗费较多的逻辑资源，增加面积。	1) 复位信号容易受到毛刺的影响。 2) 在复位信号释放(release)的时候容易出现问题。具体就是说：若复位释放刚好在时钟有效沿附近时，很容易使寄存器输出出现亚稳态 (在时钟上升沿附近 rst 置 1，这时候建立时间还不够长，数据可能还未打入寄存器，导致输出不确定），从而导致亚稳态。
总结	复位信号使用前进行滤毛刺和异步复位/同步释放处理，而且复位信号低电平有效。	

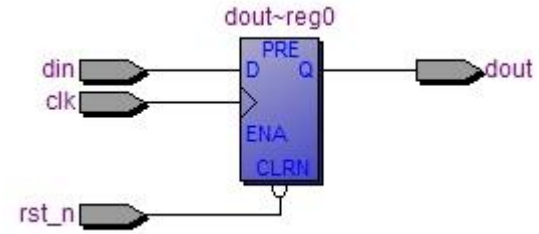
- Synthesized circuit of synchronous reset

```verilog
//Synchronous Reset
module sync_rst(clk, rst_n, din, dout);
    input clk;
    input rst_n;
    input din;
    output dout;

    reg dout;

    always@(posedge clk) begin
        if(!rst_n)
            dout <= 1'b0;
        else
            dout <= din;
    end
endmodule
```



- Synthesized circuit of asynchronous reset

```verilog
//Asynchronous Reset
module async_rst(clk, rst_n, din, dout);
    input clk;
    input rst_n;
    input din;
    output dout;

    reg dout;

    always@(posedge clk or negedge rst_n) begin
        if(!rst_n)
            dout <= 1'b0;
        else
            dout <= din;
    end
endmodule
```

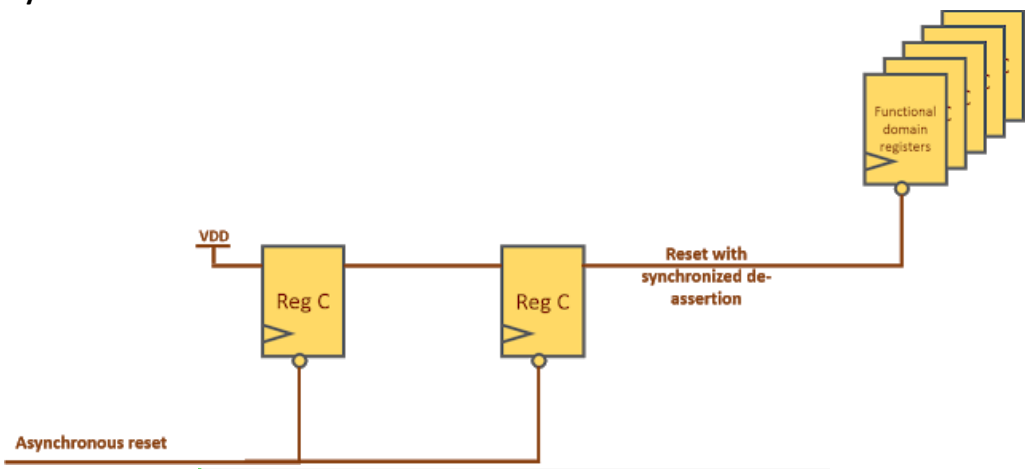


Synchronized Asynchronous Reset a.k.a. Asynchronous Reset and Synchronous Release

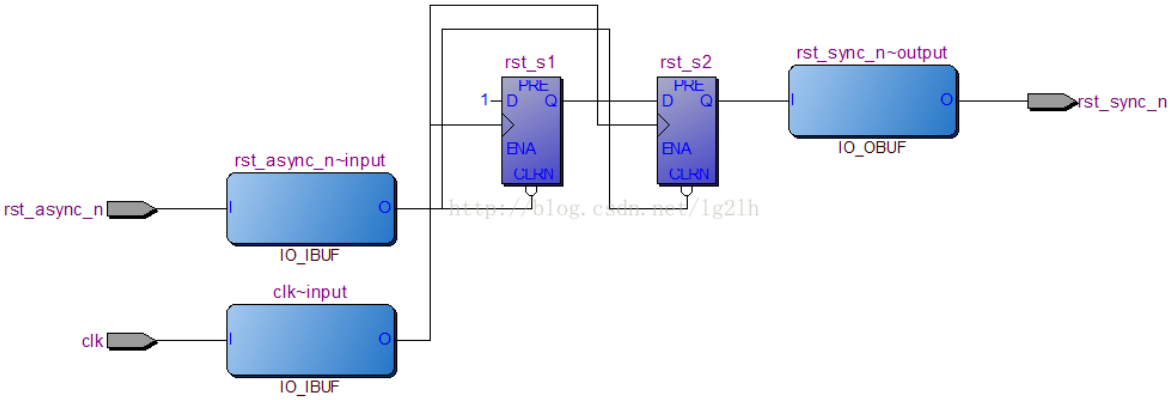
- **异步复位/同步释放电路:** aka Reset Synchronizer，就是在复位信号到来时不受时钟信号的同步，而是在复位信号释放时受到时钟信号的同步。
- **Need for reset synchronizer**: The way most of the designs have been modelled needs **asynchronous reset assertion** and **synchronous de-assertion**.

a) When reset is asserted, it propagates to all designs; brings them to reset state whether or not clock is toggling; i.e. assertion should be asynchronous
b) When reset is deasserted, wait for a clock edge, and then, move the system to next state as per the FSM (Finite State Machine); i.e. deassertion should be synchronous

- **Synthesized circuit**

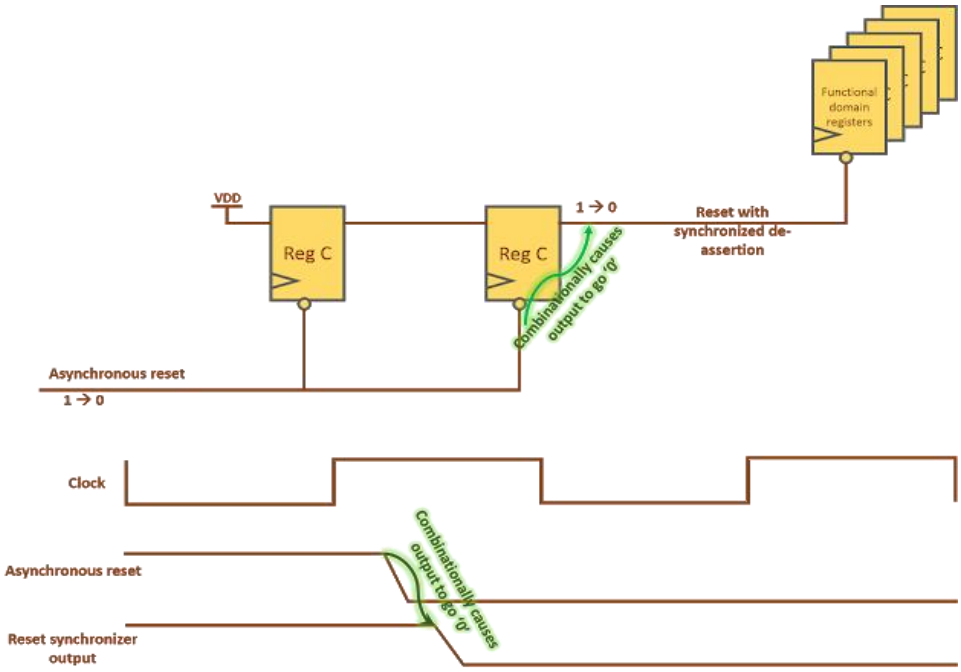



```
module ex1 (
output rst_sync_n,
input clk, rst_async_n);


reg rst_s1, rst_s2;
always @ (posedge clk, posedge rst_async_n)
if (rst_async_n) begin
rst_s1 <= 1'b0;
rst_s2 <= 1'b0;
end
else begin
rst_s1 <= 1'b1;
rst_s2 <= rst_s1;
end

assign rst_sync_n = rst_s2;
endmodule
```

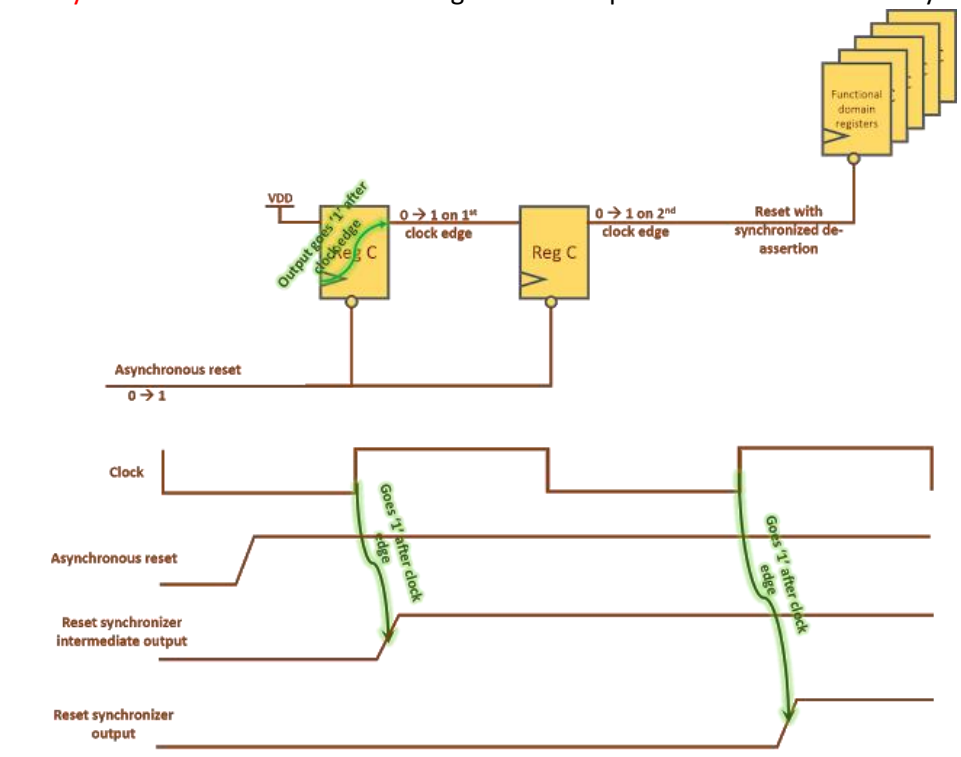


- **<u>How reset synchronizer works</u>**:

a) **Asynchronous Reset Assertion**: When the reset is asserted, it first propagates to reset synchronizer flops. It resets both the flops of reset synchronizer asynchronously (without waiting for clock edge) thereby generating reset assertion for fanout registers.

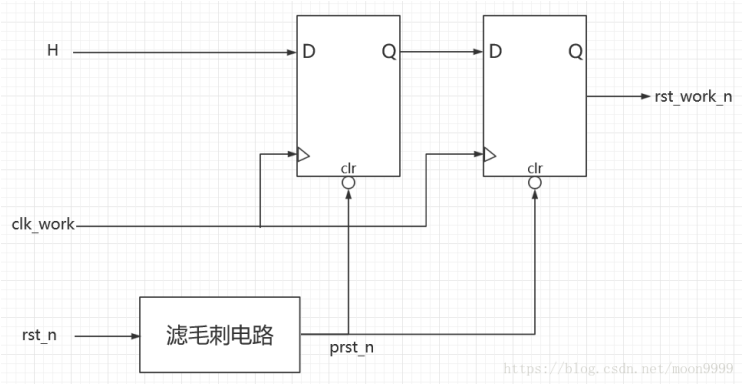


b) **Synchronous De-assertion**: Similarly, the de-assertion of reset first reaches the two flops of reset synchronizer. Now, the first flop in chain propagates 1 to intermediate output upon arrival of a clock edge. Upon next clock edge, this signal propagates to the output thereby reaching the fanout registers. The reset de-assertion timing (recovery and removal checks timing) should be met from second stage of reset synchronizer to all the domain registers' reset pins as the deassertion is synchronous.

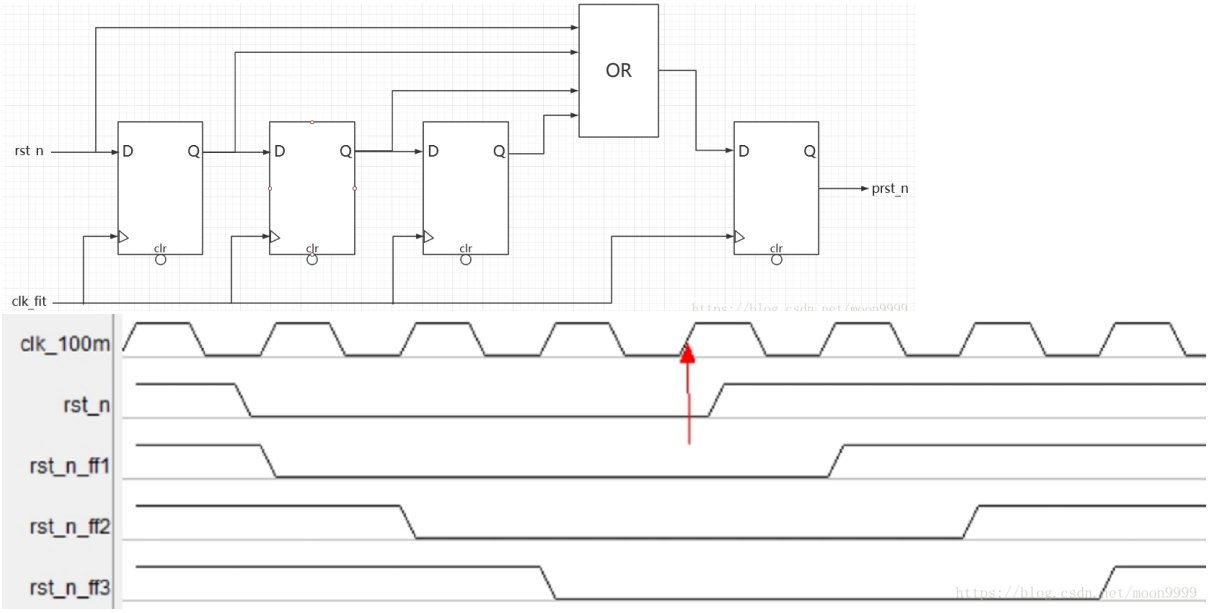


Glitch Filtering (assume negative reset, low voltage level)

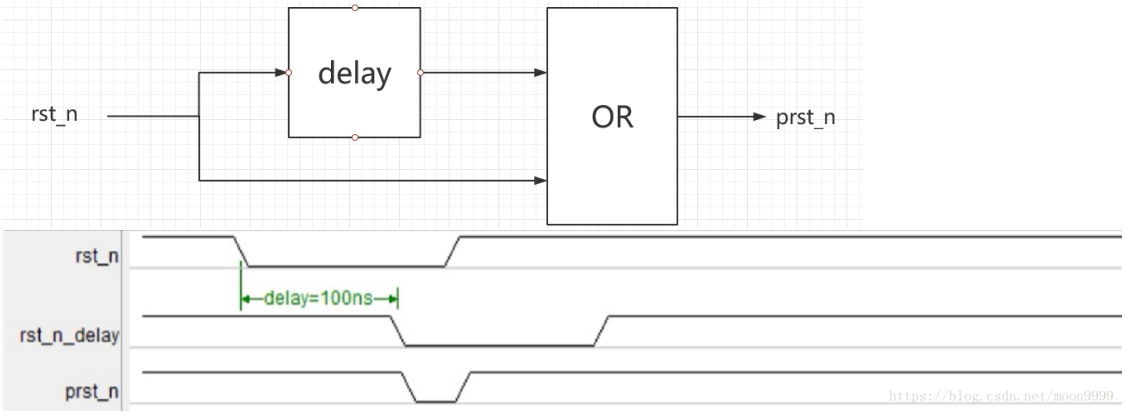
- 滤除复位信号毛刺（体现在异步复位|同步释放电路中）



a) 滤毛刺时钟+触发器完成: assume 滤毛刺时钟 clk_fit=100M，即 T=10ns。在 OR 前放置了 N 个延时 D 触发器，就可以滤除 N*T 时长以下的复位毛刺信号。复位信号必须维持 L 电平在 N 个 clk_fit 时钟周期以上，OR（或门）才能在时钟沿出打出一个 L 信号出来。



b) 通过延时器件完成: 如果我要滤去 100ns 以下毛刺，那么就引入 100ns 延时器件



Reference

[1] Verilog 中同步复位和异步复位比较

https://blog.csdn.net/chriscb/article/details/77097548

[2] 在 ASIC 中异步复位信号的处理——滤毛刺和异步复位/同步撤离
https://blog.csdn.net/moon9999/article/details/81916578

[3] Reset Synchronizer

https://vlsiuniverse.blogspot.com/2016/09/reset-synchronizer.html

[4] 同源时钟、同相位时钟、同时钟域时钟

https://blog.csdn.net/icxiaoge/article/details/80962041

[5] Digital Integrated Circuits (2nd Edition) by Jan M. Rabaey

https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxyYWplZXZ2Y2UyMDA3fGd4OjNkOTk3MmQ0OTkyYmJiZjU

[6] Hardware ----同步电路和异步电路时序设计及流水线思想

https://blog.csdn.net/weixin_42683993/article/details/90692945

[7] Asynchronous Circuits by Maitham Shams

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.1788&rep=rep1&type=pdf

[8] 同步电路和异步电路的区别

https://blog.csdn.net/sad_123_happy/article/details/16926641

[9] 复位最佳方式：异步复位，同步释放

https://blog.csdn.net/frank_wff/article/details/43226507