verilog

1. 异步复位

异步复位，同步释放，复位释放时需和clk满足时序关系。

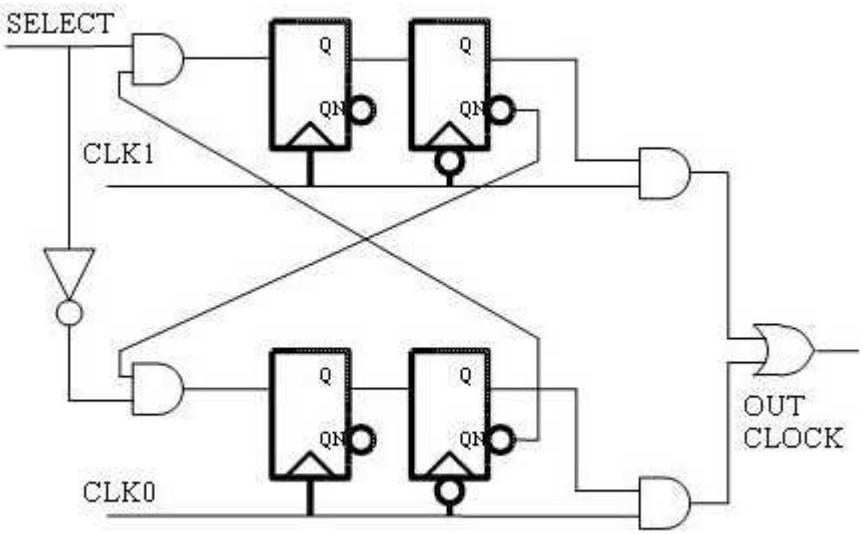
```
always@(posedge clk or negedge rst_n) begin
  if (!rst_n) begin
  end else begin
  end
end
```

2. 同步复位

同步复位时rst_n拉低， 时钟需要一直有。

```
always@(posedge clk) begin
  if (!rst_n) begin
  end else begin
  end
end
```

3. 无毛刺切换电路



4. 操作符优先级

Table 12—Precedence rules for operators

+ - ! ~ (unary)	Highest precedence
**	
* / %	
+ - (binary)	
<< >> <<< >>>	
< <= > >=	
== != === !==	
& ~&	
^ ^~ ~^	
| ~|	
&&	
||	
?: (conditional operator)	Lowest precedence

5. 驱动强度

Drive strength
four driving strengths:

```
supply strong pull weak
```

```
assign (strong1, pull0) mynet = enable ;
```

The charge strength specification shall be used only with trireg nets.
three charge storage strengths:

```
large medium small
```

```
# Strength name Strength level
# supply0 7
# strong0 6
# pull0 5
# large0 4
# weak0 3
# medium0 2
# small0 1
```

```
# highz0 0
# highz1 0
# small1 1
# medium1 2
# weak1 3
# large1 4
# pull1 5
# strong1 6
# supply1 7
```

6. define使用

```
#``拼接

`define  NAME   yb
`define  INST   `NAME``_inst    等价于yb_inst
```

```
# args方式
#比如有如下define定义
`define append(f) f``_master

#当使用时
`append(clock)等价于 clock_master
```

```
`define H(x) "Hello, x"

$display(`H(world));
#打印内容
Hello, world
```

7. fwrite

```
module file_test(

    );
reg [3:0]data[0:15];
reg [3:0]data2[0:15];
integer handle1;
integer i=0;
initial
begin
    $readmemb("num.txt",data);//默认文件在工程所在路径目录

    handle1 = $fopen("num2.txt","w");
    repeat(16)
    begin
        $fwrite(handle1,"%d\n",data[15-i]);
```

```
        i = i+1;
    end
    $fclose(handle1);
end
endmodule
```

8. module parameter

如果port width是参数

```
module a #(parameter WIDTH=8)(
    input [WIDTH-1:0] datai,
    output [WIDTH-1:0] datao
);
assign datao = datai;
endmodule
```