1． 异步清零的 RS 触发器；

S	R	Q	Q^{n+1}		说明
0	0	0	0	Q	输出状态不变
0	0	1	1		
0	1	0	0	0	输出状态与 S的状态相同
0	1	1	0		
1	0	0	1	1	输出状态与 S的状态相同
1	0	1	1		
1	1	0	-		输出状态不定
1	1	1	-		

（1）RSFF 代码

```
module rschufaqi(clkin,rst,r,s,qout);

input clkin,rst,r,s;

output reg qout;

always @ (posedge clkin or negedge rst)

 if (rst==0)

   qout <= 0;

 else

   begin

    case({r,s})

      2'b00: qout<=qout;

      2'b01:qout <= 0;

      2'b10:qout <= 1;

      2'b11: qout <= 1'bx;

      default:qout <= 0;

    endcase

   end
```

```
endmodule
```

（2）测试文件代码

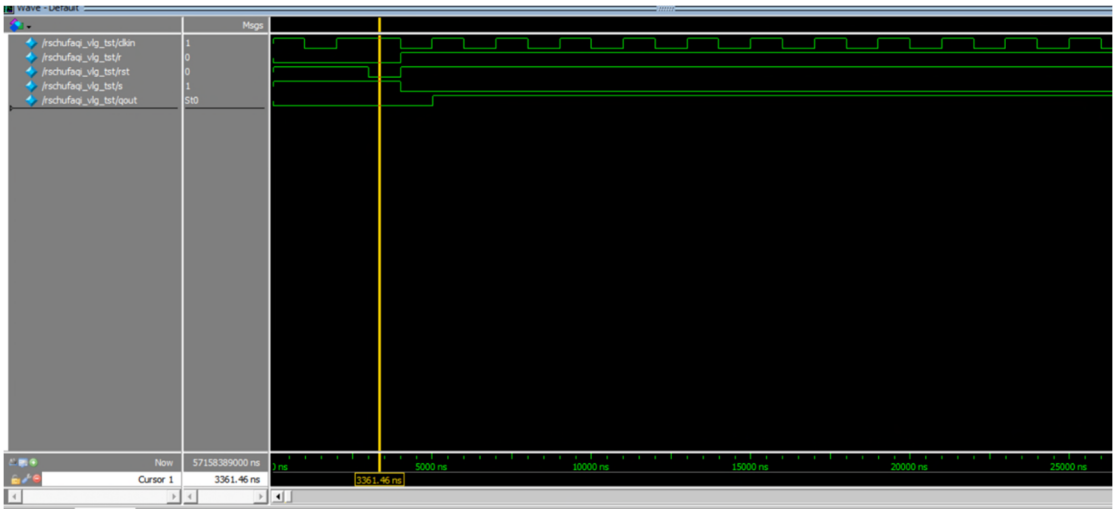
```
`timescale 100 ns/ 1 ps
module rschufaqi_vlg_tst();
// constants
// general purpose registers
//reg eachvec;
// test vector input registers
reg clkin;
reg r;
reg rst;
reg s;
// wires
wire qout;

// assign statements (if any)
rschufaqi i1 (
// port map - connection between master ports and signals/registers
    .clkin(clkin),
    .qout(qout),
    .r(r),
    .rst(rst),
```

```
      .s(s)
);
initial
begin
// code that executes only once
// insert code here --> begin
  rst=1;
clkin=1;
r=0;
s=1;
#30
rst=0;
clkin=0;
#10
rst=1;
clkin=1;
r=1;
s=0;
// --> end
$display("Running testbench");
end
always
```

```
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #10 clkin=~clkin;
//@eachvec;
// --> end
end
endmodule
```

（3）仿真波形

r=0,s=1,rst=0;　　qout=0.



2．　异步清零同步置位的 D 触发器；

输入				输出	
clk	set	clr	d	q	nq
X	X	0	X	0	1
↑	0	1	X	1	0
↑	1	1	0	0	1
↑	1	1	1	1	0
0,1, ↓	1	1	X	保持	保持

（1）DFF 代码

```
module dffchufaqi(clkin,rst,D,set,qout);
input clkin,rst,D,set;
output reg qout;
always @ (negedge clkin or negedge rst)
if(!rst)
qout<=0;
else if(set==1)
qout<=1;
else
qout<=D;
endmodule
```

（2）测试文件代码

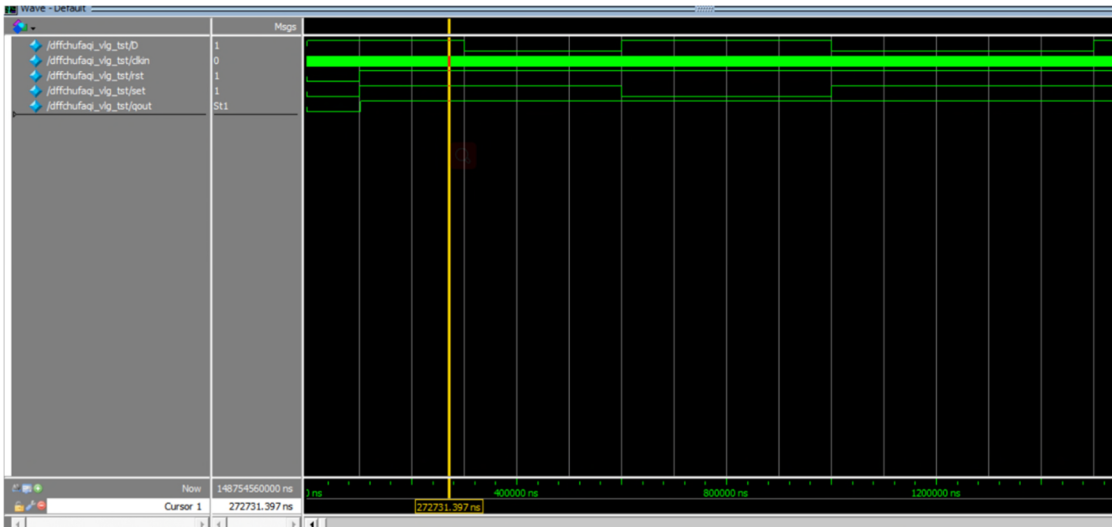
```
`timescale 100 ns/ 1 ps
module dffchufaqi_vlg_tst();
// constants
```

```
// general purpose registers
//reg eachvec;
// test vector input registers
reg D;
reg clkin;
reg rst;
reg set;
// wires
wire qout;

// assign statements (if any)
dffchufaqi i1 (
// port map - connection between master ports and signals/registers
    .D(D),
    .clkin(clkin),
    .qout(qout),
    .rst(rst),
    .set(set)
);
initial
begin
// code that executes only once
```

```
// insert code here --> begin
        clkin=0;
rst=0;
set=0;
D=1;
#1000
clkin=1;
rst=1;
set=1;
D=1;
#2000
set=1;
D=0;
#3000
set=0;
D=1;
#4000
set=1;
D=0;
#5000
set=1;
D=1;
```

```
// --> end
$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #10 clkin=~clkin;
//@eachvec;
// --> end
end
endmodule
```

（3）仿真波形 D=1,st=1,set=1;　　qout=1.



3． 异步清零同步置位同步使能的 JK 触发器；

输入					输出	
$\overline{R_D}$	$\overline{S_D}$	CP	J	K	Q	$\overline{Q}$
0	1	×	×	×	1	0
1	0	×	×	×	0	1
0	0	×	×	×	1	1
1	1	↓	0	0	Q	$\overline{Q}$
1	1	↓	1	0	1	0
1	1	↓	0	1	0	1
1	1	↓	1	1	触 发	
1	1	1	1	1	Q	Q

（1）JKFF 代码

```
module jkchufaqi(clkin,rst,j,k,ena,set,qout);
input clkin,rst,j,k,ena,set;
output reg qout;
always @ (posedge clkin or posedge rst )
 if (rst==1)
   qout <= 0;
 else if(set==1)
 qout<=1;
 else    if(ena==1)
   begin
     case({j,k})
```

```
        2'b00: qout<=qout;
        2'b01:qout <= 0;
        2'b10:qout <= 1;
        2'b11: qout <=~qout;
        default:qout <= 0;
      endcase
    end
    else qout <= qout;
endmodule
```

（2）测试文件代码

```
`timescale 100 ns/ 1 ps
module jkchufaqi_vlg_tst();
// constants
// general purpose registers
//reg eachvec;
// test vector input registers
reg clkin;
reg ena;
reg j;
reg k;
reg rst;
reg set;
```

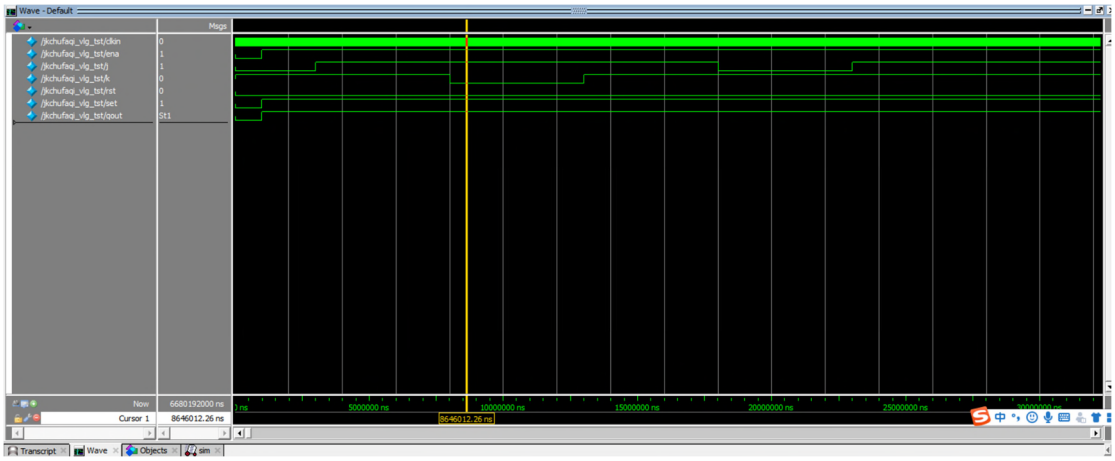
```
// wires
wire qout;


// assign statements (if any)
jkchufaqi i1 (
// port map - connection between master ports and signals/registers
      .clkin(clkin),
      .ena(ena),
      .j(j),
      .k(k),
      .qout(qout),
      .rst(rst),
      .set(set)
);
initial
begin
// code that executes only once
// insert code here --> begin
       clkin=0;
rst=0;
set=0;
ena=0;
```

```
j=0;
k=1;
#10000
set=1;
ena=1;
j=0;
k=1;
#20000
j=1;
k=1;
#50000
j=1;
k=0;
#50000
j=1;
k=1;
#50000
j=0;
k=1;
#50000
j=1;
k=1;
```

```
// --> end
$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #10 clkin=~clkin;
//@eachvec;
// --> end
end
endmodule
```

（3）仿真波形

ena=1,j=1,k=0,rst=0,set=1;　　qout=1;



4． 异步清零异步置位同步使能的 T 触发器

Tn	Qn+1
0	Qn
1	/Qn

(1)TFF 代码

```
module tchufaqi(clkin,rst,T,ena,set,qout);
input clkin,rst,T,ena,set;
output reg qout;
always @ (posedge clkin or posedge rst or posedge set or
posedge ena )
 if (rst==1)
   qout <= 0;
 else if(set==1)
 qout<=1;
 else if(ena==1)
   begin
    case({T})
    1'b0: qout<=qout;
```

```
      1'b1: qout <=~qout;
      default:qout <= 0;
    endcase
  end
  else qout <= qout;
endmodule
```

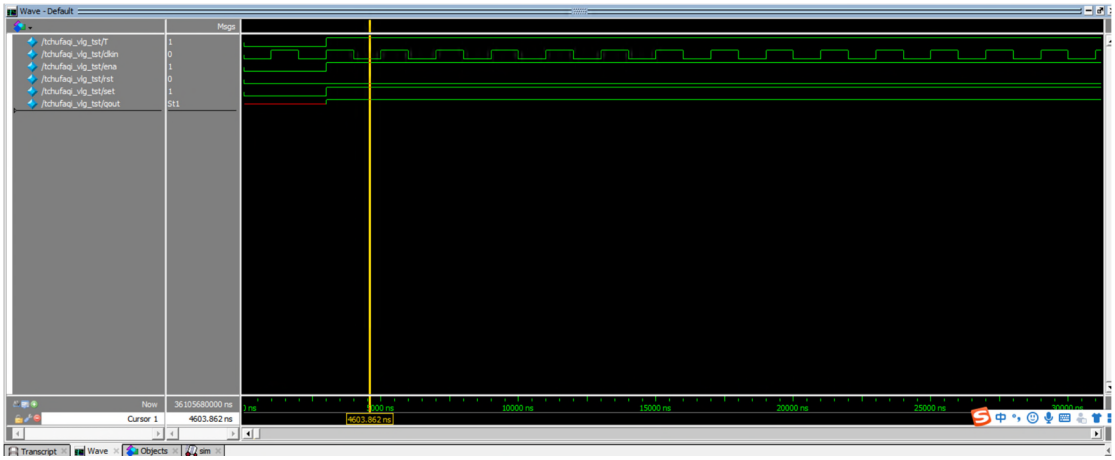
(2)测试文件代码

```
 `timescale 100 ns/ 1 ps
module tchufaqi_vlg_tst();
// constants
// general purpose registers
//reg eachvec;
// test vector input registers
reg T;
reg clkin;
reg ena;
reg rst;
reg set;
// wires
wire qout;

// assign statements (if any)
```

```
tchufaqi i1 (
// port map - connection between master ports and signals/registers
      .T(T),
      .clkin(clkin),
      .ena(ena),
      .qout(qout),
      .rst(rst),
      .set(set)
);
initial
begin
// code that executes only once
// insert code here --> begin
       clkin=0;
rst=0;
set=0;
ena=0;
T=0;
#30
set=1;
ena=1;
T=1;
```

```
#500000
T=0;
#500000
T=1;
// --> end
$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #10 clkin=~clkin;
//@eachvec;
// --> end
end
endmodule
```

（4）仿真波形 T=1,ena=1,rst=0,set=1;　　qout=1.


Wave - Default
Msgs
/tchufaqi_vlg_tst/T
/tchufaqi_vlg_tst/clkn
/tchufaqi_vlg_tst/ena
/tchufaqi_vlg_tst/rst
/tchufaqi_vlg_tst/set
/tchufaqi_vlg_tst/qout
1
0
1
0
1
St1
Now
36105680000 ns
Cursor 1
4603.862 ns
1 ns
5000 ns
10000 ns
15000 ns
20000 ns
25000 ns
30000 ns
4603.862 ns
Transcript
Wave
Objects
sim