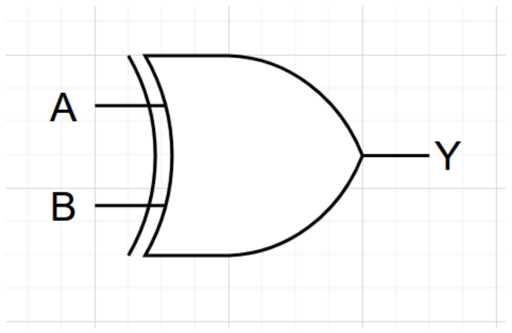
CS4093 - Assignment 1

Alexander Stradnic - 119377263

XOR : An Exclusive OR gate works in a similar manner to OR, except that it only activates on one of the inputs only, not both.

A	B	Y	
0	0	0	
0	1	1	
1	0	1	
1	1	0	

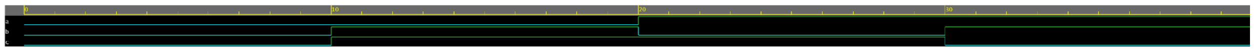
Verilog Code

```
module xor_gate (c,a,b);
  input     a,b;
  output    c;
  assign c = a^b;
endmodule
```

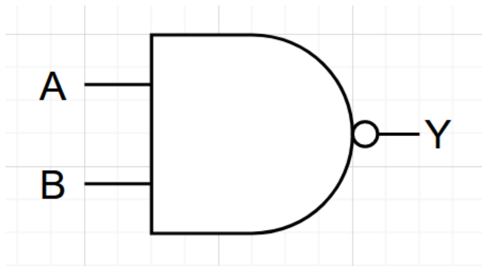
Testbench Code

```
module xorgate;
  reg a;
  reg b;
  wire c;

  xor_gate TEST(.c(c), .a(a), .b(b));

  initial begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
      a=0;
      b=0;
      #10;
      a=0;
      b=1;
      #10;
      a=1;
      b=0;
      #10;
      a=1;
      b=1;
      #10;
  end
endmodule
```

Waveform



NAND : A NAND gate works by inverting the output taken from an AND gate.

A	B	Y	
0	0	1	
0	1	1	
1	0	1	
1	1	0	

Verilog Code

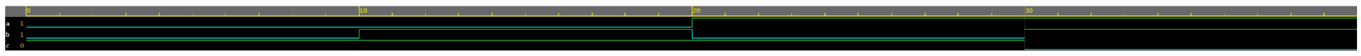
```
module nand_gate (c,a,b);
  input     a,b;
  output    c;
  assign c = !(a&b);
endmodule
```

Testbench Code:

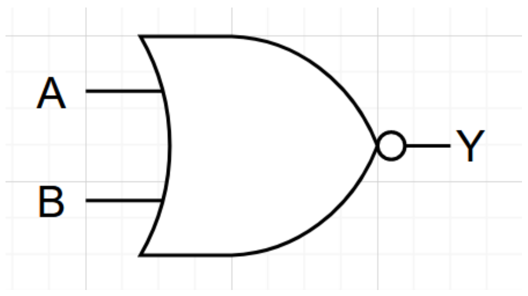
```
module nandgate;
  reg a;
  reg b;
  wire c;

  nand_gate TEST(.c(c), .a(a), .b(b));

  initial begin
```

```
      $dumpfile("dump.vcd");
      $dumpvars(1);
      a=0;
      b=0;
      #10;
      a=0;
      b=1;
      #10;
      a=1;
      b=0;
      #10;
      a=1;
      b=1;
      #10;
  end
endmodule
```

Waveform



NOR : A NOR gate works by inverting the output from an OR gate.

A	B	Y	
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Verilog Code

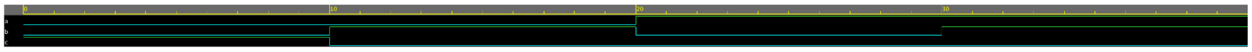
```
module nor_gate (c,a,b);
  input     a,b;
  output    c;
  assign c = !(a|b);
endmodule
```

Testbench Code

```
module norgate;
```

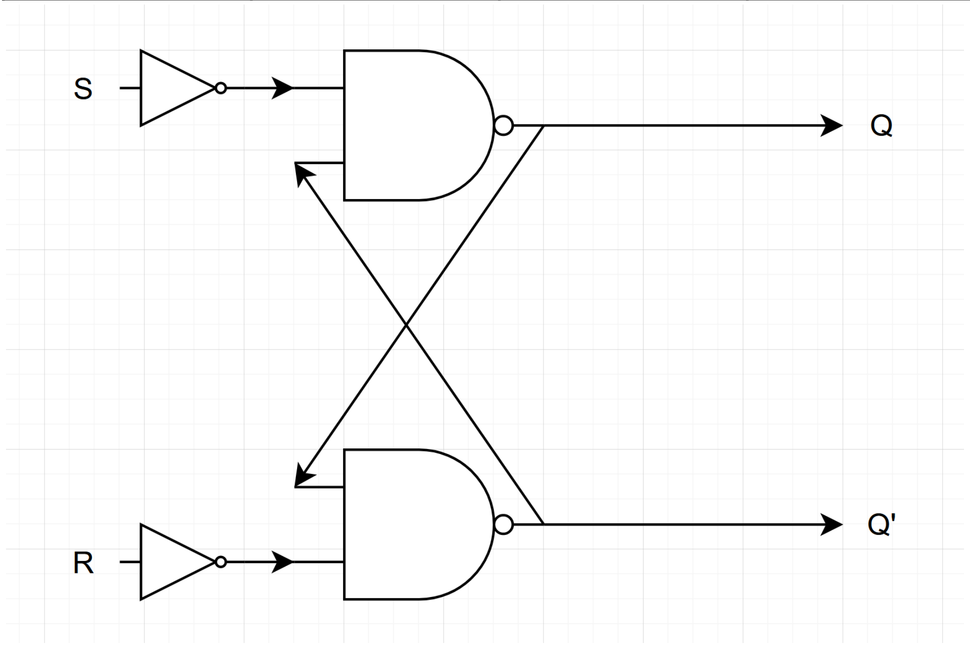
```
  reg a;
  reg b;
  wire c;

  nor_gate TEST(.c(c), .a(a), .b(b));

  initial begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
      a=0;
      b=0;
      #10;
      a=0;
      b=1;
      #10;
      a=1;
      b=0;
      #10;
      a=1;
      b=1;
      #10;
  end
endmodule
```

Waveform



SR FLIP FLOP : An SR flip-flop can store a 1-bit value, and operates using 2 NAND gates. An input is designated as the setter (S) with the other input being the resetter (R).

S	R	Q	Q’	State
0	0	Q	Q’	No change
1	0	1	0	Set
0	1	0	1	Clear
1	1	X	X	Invalid



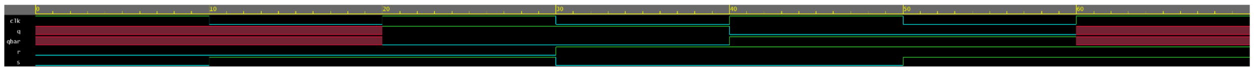
Verilog Code

```
module sr_flipflop(q,qbar,clk,s,r);
input clk;
input s;
input r;
output q,qbar;
reg q,qbar;
 always @(posedge clk) begin
 if (r == 0 && s==0) begin
 q <= q;
 qbar <= qbar;
 end else if (s == 1 && r == 0) begin
 q <= 1;
 qbar <= 0;
 end else if (s == 1 && r == 1) begin
 q <= 1'bx;
 qbar <= 1'bx;
 end else if (s == 0 && r == 1) begin
 q <= 0;
 qbar <= 1;
 end
 end
endmodule
```

Testbench Code

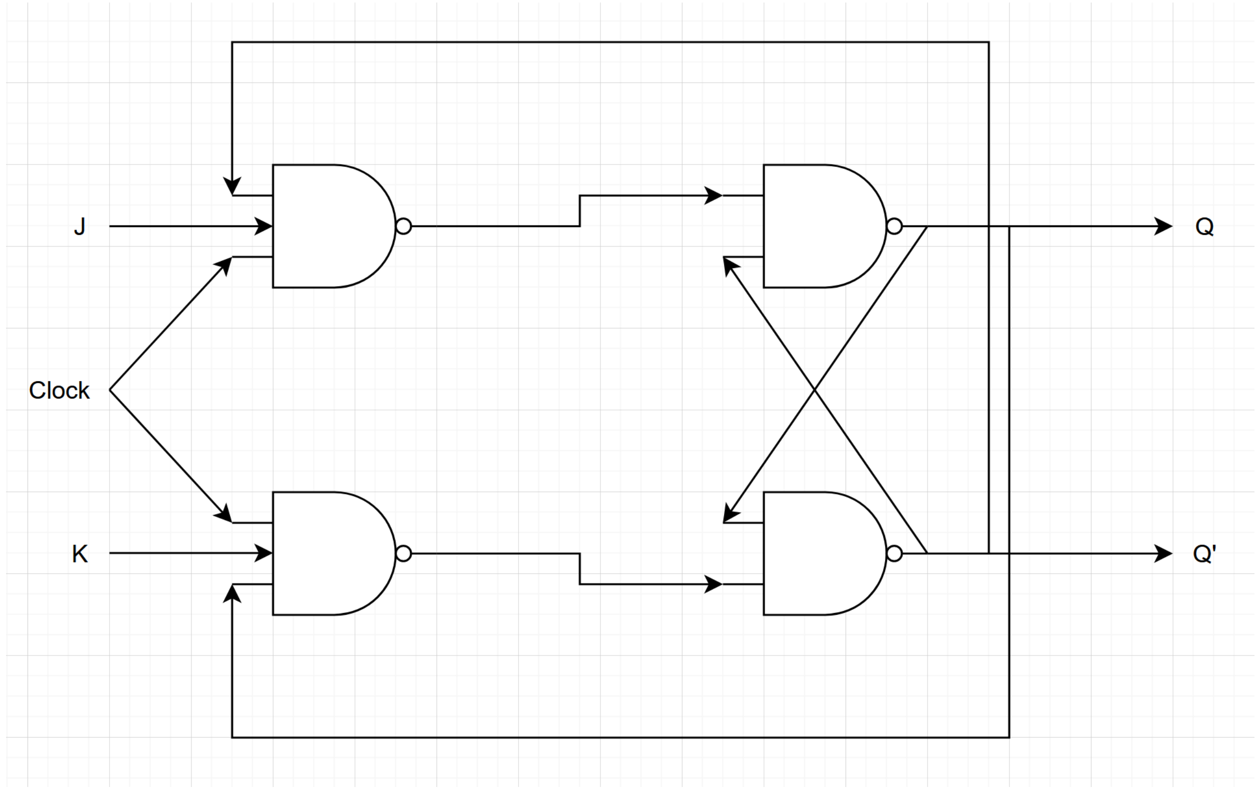
```
module TEST;
reg clk,s,r;
wire q,qbar;
 sr_flipflop TEST (.q(q), .qbar(qbar), .clk(clk), .s(s),.r(r));
initial begin
$dumpfile("dump.vcd");
$dumpvars(1);
clk=1; // initial
r = 0;
s = 0;
#10;
clk=0; // set but clock inactive
r = 0;
s = 1;
#10;
clk=1; // clock active, thus set works
#10;
clk=0; // reset but clock inactive
r = 1;
s = 0;
#10;
clk = 1; // reset activates
#10;
clk = 0;
s = 1; // bad state but not active
#10;
clk = 1; // bad state activated
#10;
end
endmodule
```

Waveform



JK FLIP FLOP : A JK flip-flop can store a 1-bit value, and operates using 2 NAND gates, in a similar manner to the SR flip-flop. The JK Flip-flop overcomes the issue with the SR flip-flop where the state is invalid when both inputs are set to 1 by flipping the state of the outputs in that instance.

J	K	Q	Q’	State
0	0	Q	Q’	No change
1	0	1	0	Set
0	1	0	1	Clear
1	1	~Q	~Q’	Toggle State



Verilog Code

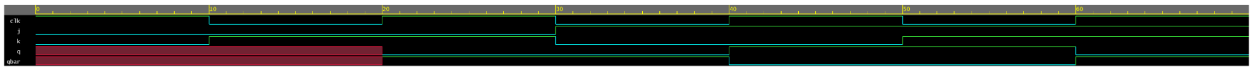
```
module jk_flipflop(q,qbar,clk,j,k);
input clk;
input j;
input k;
output q,qbar;
reg q,qbar;
 always @(posedge clk) begin
   if (j==0 && k==0) begin
 q <= ~q;
 qbar <= ~qbar;
   end else if (j == 1 && k == 0) begin
 q <= 1;
 qbar <= 0;
   end else if (j == 1 && k == 1) begin
 q <= ~q;
 qbar <= ~qbar;
   end else if (j == 0 && k == 1) begin
 q <= 0;
 qbar <= 1;
 end
 end
endmodule
```

Testbench Code

```
module TEST;
reg clk,j,k;
wire q,qbar;
  jk_flipflop TEST (.q(q), .qbar(qbar), .clk(clk), .j(j),.k(k));
initial begin
$dumpfile("dump.vcd");
$dumpvars(1);
clk=1; // initial
j = 0;
k = 0;
#10;
clk=0; // set but clock inactive
j = 0;
k = 1;
#10;
clk=1; // clock active, thus set works
#10;
clk=0; // reset but clock inactive
j = 1;
k = 0;
#10;
clk = 1; // reset activates
#10;
clk = 0;
k = 1; // bad state but not active
```

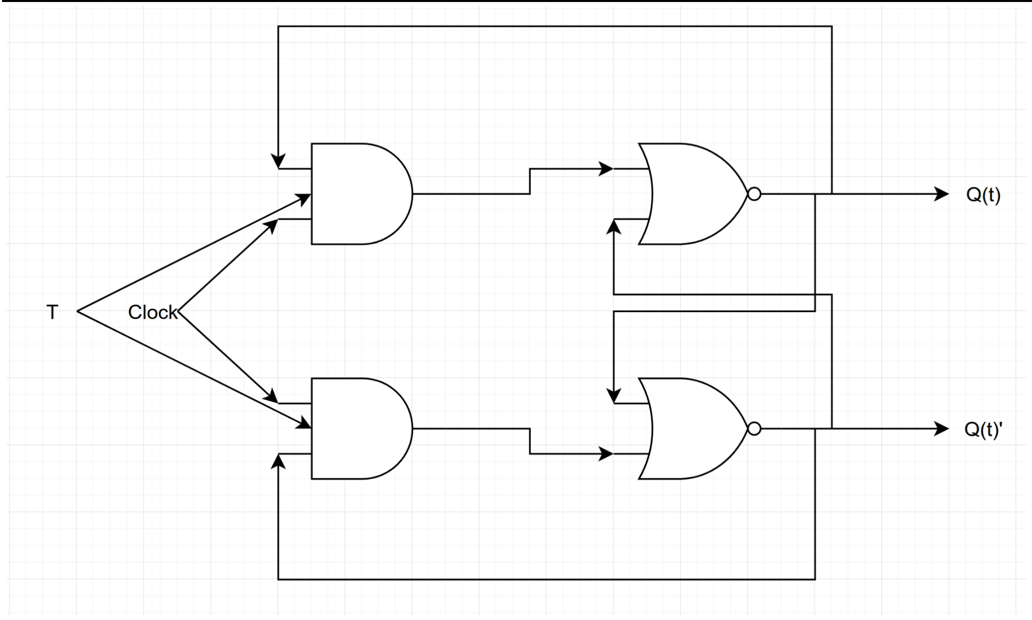
```
#10;
clk = 1; // bad state activated
#10;
end
endmodule
```

Waveform



T FLIP FLOP : An T flip-flop can store a 1-bit value, and flips with the input (AND clock if relevant).

T	Q	Q’	State
0	Q	Q’	Hold State
1	~Q	~Q’	Toggle State



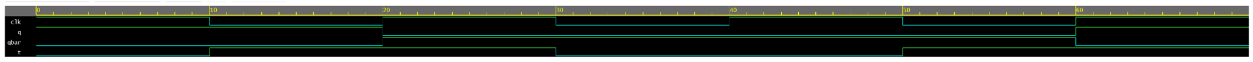
Verilog Code

```
module t_flipflop(q,qbar,clk,t);
input clk;
input t;
output q,qbar;
reg q,qbar;

 always @(posedge clk) begin
   if(q === 1'bx) begin
   q <= 1;
   qbar <= 0;
   end else if (t == 0) begin
 q <= q;
 qbar <= qbar;
   end else if (t == 1) begin
 q <= ~q;
 qbar <= ~qbar;
 end
 end
endmodule
```

Testbench Code

```
module TEST;
reg clk,t;
wire q,qbar;
  t_flipflop TEST (.q(q), .qbar(qbar), .clk(clk), .t(t));
initial begin
$dumpfile("dump.vcd");
$dumpvars(1);
clk=1; // initial
t = 0;
#10;
clk=0; // toggle but clock inactive
t = 1;
#10;
clk=1; // clock active, thus toggle works
#10;
clk=0; // hold but clock inactive
t = 0;
#10;
clk = 1; // hold 'activates'
#10;
clk = 0;
t = 1; // toggle but not active
#10;
clk = 1; // toggle state activated
#10;
end
endmodule
```

Waveform


clk
q
qbar
t