Assignment 1

# XOR Gate

XOR gate will only output a 1 if the 2 inputs are different. For example if A is a 1 and B is 0, then C will be a 1. If A and B are the same, then C will be 0.

XOR Table

|  |  |  |
| --- | --- | --- |
| A | B | C |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

XOR diagram

Chart, line chart

Description automatically generated

design.sv

module xor\_gate(input a,b, output c);

  assign c = a^b;

endmodule

testbench.sv

module test;

    reg a,b;

    wire c;

  xor\_gate TEST(.a(a), .b(b), .c(c));

  initial begin

    $dumpfile("dump.vcd");

    $dumpvars(1);

    a=0;

    b=0;

    #10

    a=1;

    b=0;

    #10

    a=0;

    b=1;

    #10

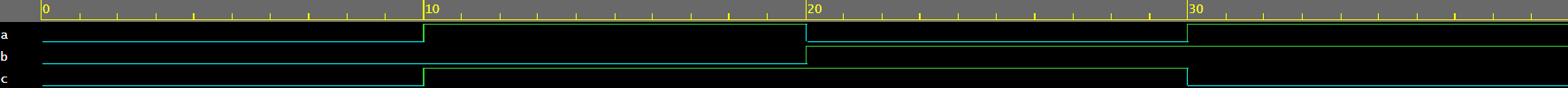
    a=1;

    b=1;

    #10;

  end

endmodule

waveform

# NAND Gate

NAND is the inverse of the AND gate. The AND gate output is 1 if all inputs are 1. NAND is the opposite, it’s output is 1 unless all inputs are 1.

NAND Table

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | C’ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

NAND Diagram

Diagram

Description automatically generated

design.sv

module nand\_gate(input a,b, output c);

  assign c = ~(a&b);

endmodule

testbench.sv

module test;

    reg a,b;

    wire c;

  nand\_gate TEST(.a(a), .b(b), .c(c));

  initial begin

    $dumpfile("dump.vcd");

    $dumpvars(1);

    a=0;

    b=0;

    #10

    a=1;

    b=0;

    #10

    a=0;

    b=1;

    #10

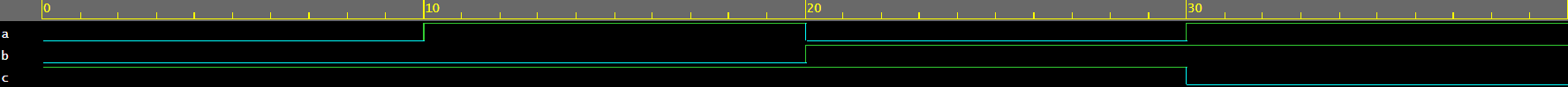
    a=1;

    b=1;

    #10;

  end

endmodule

waveform

# NOR

NOR gate is the opposite of the OR gate. For OR, the output is 1 when any of it’s inputs are 1. For NOR, the output is 0 if any of it’s inputs are 1.

NOR Table

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | C’ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

NOR diagram

Diagram

Description automatically generated

design.sv

module nor\_gate(input a,b, output c);

  assign c = ~(a|b);

endmodule

testbench.sv

module test;

    reg a,b;

    wire c;

  nor\_gate TEST(.a(a), .b(b), .c(c));

  initial begin

    $dumpfile("dump.vcd");

    $dumpvars(1);

    a=0;

    b=0;

    #10

    a=1;

    b=0;

    #10

    a=0;

    b=1;

    #10

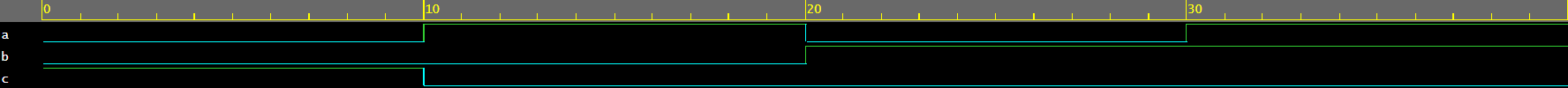
    a=1;

    b=1;

    #10;

  end

endmodule

Waveform

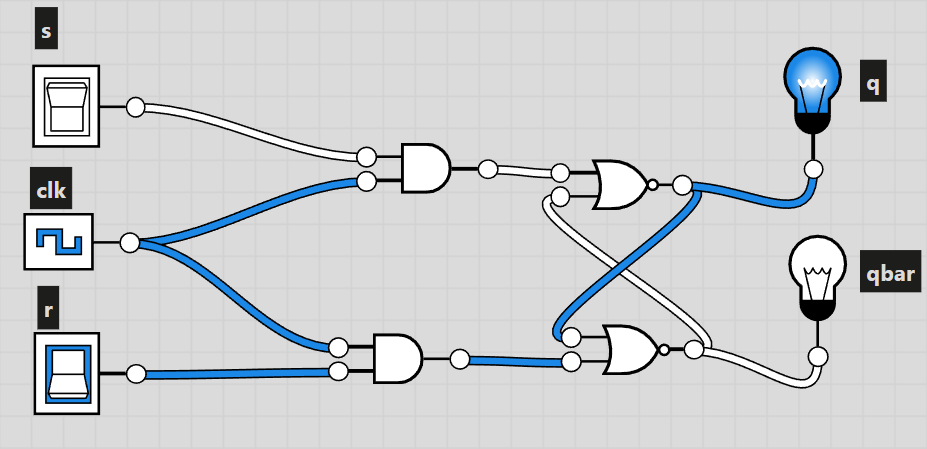
# SR Flip Flop

SR flip flop is a synchronous circuit that has inputs S, R and clk. When S is 1, then Q is 1. When R is 1 then Qbar is 1.

Table

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | Qbar |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | X | X |

Diagram



design.sv

module srff(input s,r,clk, output q,qb);

 reg q,qb;

 initial

   begin

     q = 1'b1;

     qb = 1'b0;

   end

  always @(posedge clk or negedge clk)

    begin

      case ({s,r})

        {1'b0,1'b0}:

          begin

            q=q;

            qb=qb;

          end

        {1'b0,1'b1}:

          begin

            q=0;

            qb=1;

          end

        {1'b1,1'b0}:

          begin

            q=1;

            qb=0;

          end

        {1'b1,1'b1}:

          begin

            q=1'bx;

            qb=1'bx;

          end

      endcase

    end

endmodule

testbench.sv

module Test;

 reg s;

 reg r;

 reg clk;

 wire q;

 wire qb;

 srff Test(.s(s), .r(r), .clk(clk), .q(q), .qb(qb));

 initial begin

   $dumpfile("dump.vcd");

   $dumpvars(1);

   clk=0;

 end

  always #5 clk = ~clk;

  initial

    begin

      s=1;

      r=0;

      #10

      s=0;

      r=1;

      #10

      s=0;

      r=0;

      #10;

      s=1;

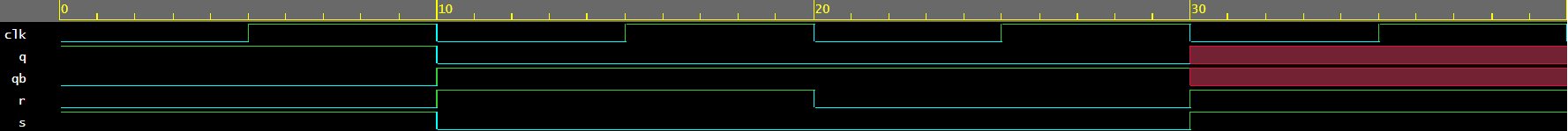
      r=1;

      #10;

      $finish;

  end

endmodule

Waveform

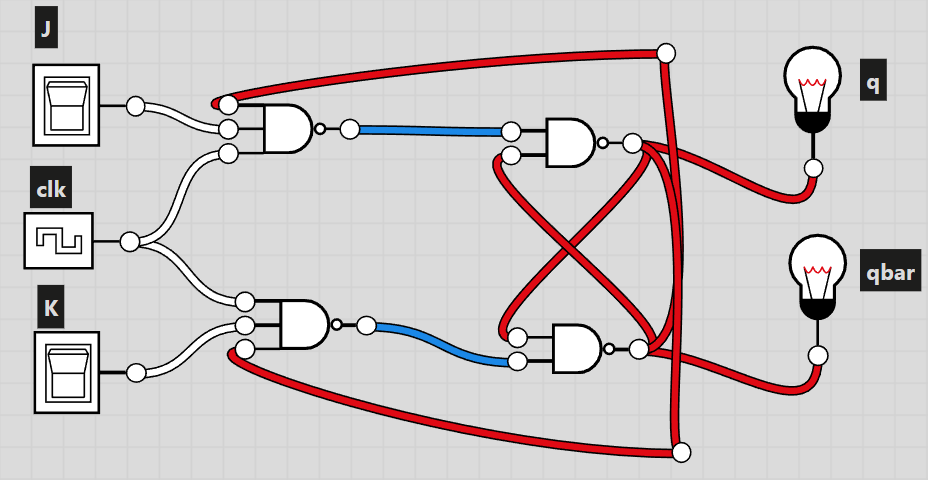
# JK Flip Flop

If J and K are different, then when the clock is 1, it takes the value of J. If J and K are both 1 and the clock is 1, then the value of K is taken. Otherwise no change happens.

Table

|  |  |  |  |
| --- | --- | --- | --- |
| J | K | Q | Qbar |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

Diagram



design.sv

module jkff(input j,k,clk, output q,qb);

 reg q,qb;

 initial

   begin

     q = 1'b1;

     qb = 1'b0;

   end

  always @(posedge clk or negedge clk)

    begin

      case ({j,k})

        {1'b0,1'b0}:

          begin

            q=q;

            qb=qb;

          end

        {1'b0,1'b1}:

          begin

            q=0;

            qb=1;

          end

        {1'b1,1'b0}:

          begin

            q=1;

            qb=0;

          end

        {1'b1,1'b1}:

          begin

            q=q;

            qb=qb;

          end

      endcase

    end

endmodule

testbench.sv

module Test;

 reg j;

 reg k;

 reg clk;

 wire q;

 wire qb;

  jkff Test(.j(j), .k(k), .clk(clk), .q(q), .qb(qb));

 initial begin

   $dumpfile("dump.vcd");

   $dumpvars(1);

   clk=0;

 end

  always #5 clk = ~clk;

  initial

    begin

      j=1;

      k=0;

      #10

      j=0;

      k=1;

      #10

      j=0;

      k=0;

      #10;

      j=1;

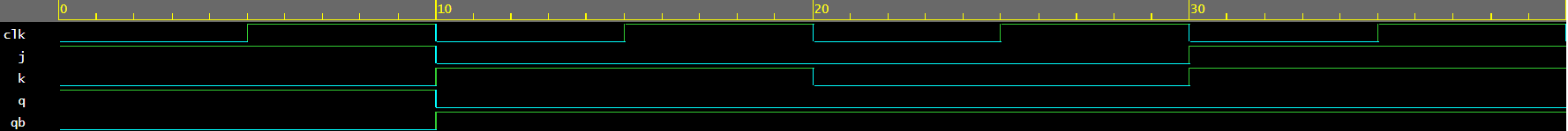
      k=1;

      #10;

      $finish;

  end

endmodule

waveform

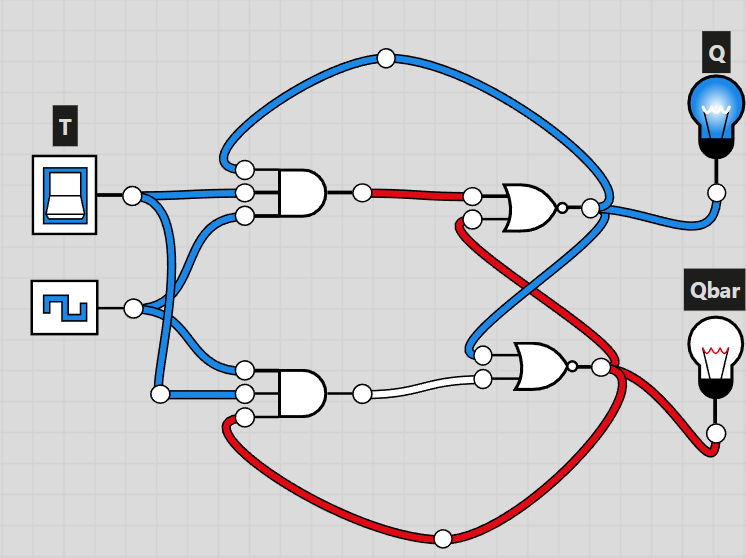
# T Flip Flop

The state only changes when T is 1. When the clock is 0, q is 1 and qbar is 0, when the clock is 1, q is 0 and qbar is 1.

Table

|  |  |  |
| --- | --- | --- |
| T | Q | Qbar |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

Diagram



design.sv

module tff(input t,clk, output q,qb);

 reg q,qb;

 initial

   begin

     q = 1'b0;

     qb = 1'b1;

   end

  always @(posedge clk or negedge clk)

    begin

      case (t)

        {1'b0}:

          begin

            q=q;

            qb=qb;

          end

        {1'b1}:

          begin

            q=~q;

            qb=~qb;

          end

      endcase

    end

endmodule

testbench.sv

module Test;

 reg t;

 reg clk;

 wire q;

 wire qb;

  tff Test(.t(t), .clk(clk), .q(q), .qb(qb));

 initial begin

   $dumpfile("dump.vcd");

   $dumpvars(1);

   clk=0;

 end

  always #5 clk = ~clk;

  initial

    begin

      t=0;

      #10

      t=1;

      #10

      t=0;

      #10;

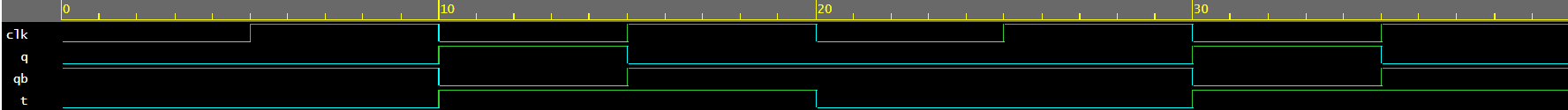
      t=1;

      #10;

      $finish;

  end

endmodule

waveform