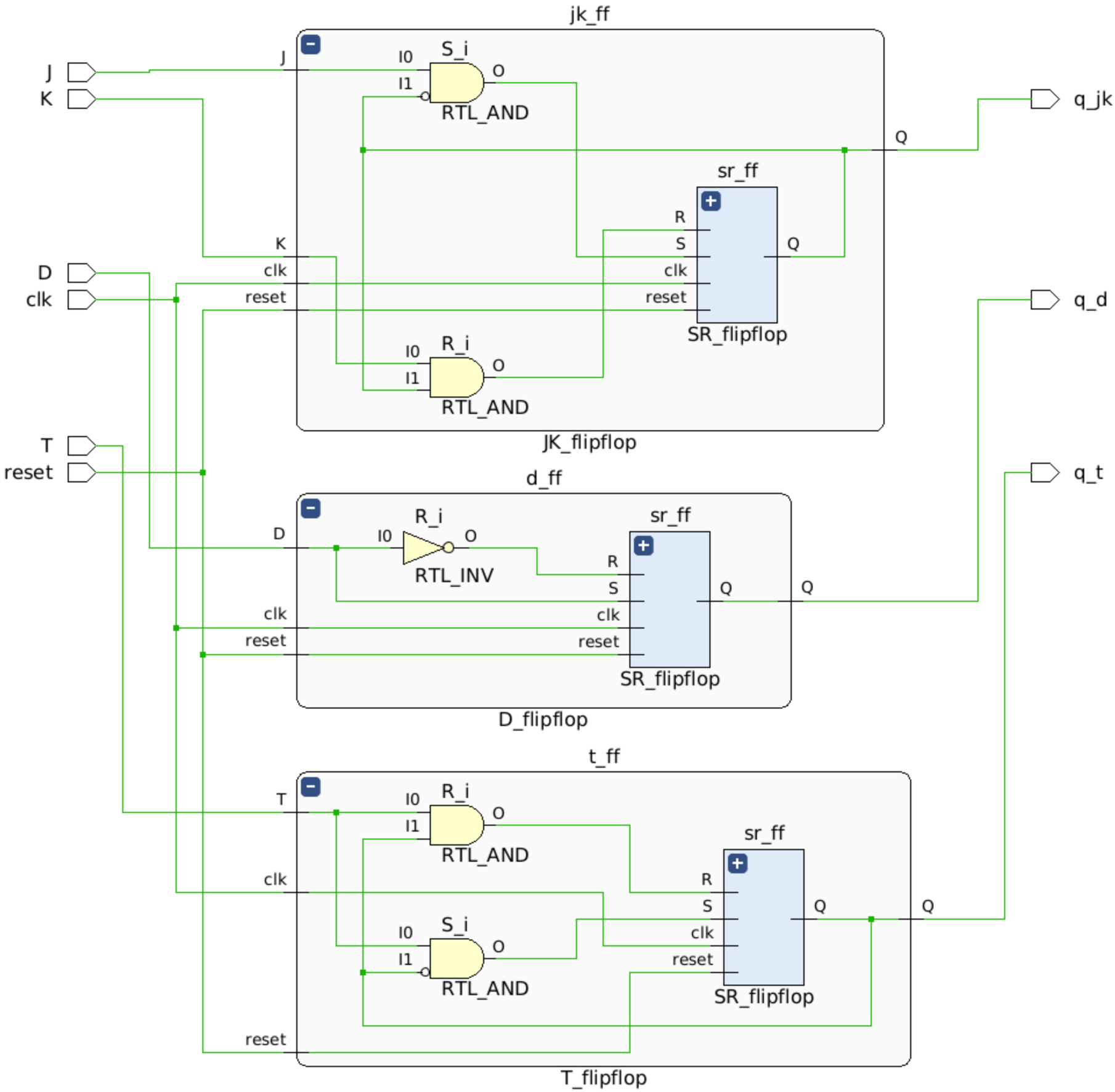
11 Cells 9 I/O Ports 27 Nets



SR_conversion.v | SR_flipflop.v | SR_conversion_tb.v | JK_flipflop.v | Untitled 7

Tcl Console | Messages | Log

```
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start wit
#   }
# }
# run 1000ns
Testing D Flip-Flop
Testing T Flip-Flop
Testing JK Flip-Flop
q_t: 1
Final States - q_jk: 0, q_d: 0, q_t: 1
$stop called at time : 50 ns : File "/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/pro
INFO: [USF-XSim-96] XSim completed. Design snapshot 'SR_conversion_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 9077
```

Type a Tcl command here

SR_flipflop.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/project_1/project_1.srcs/sources_1/new/SR_flipflop.v

```
2  //////////////////////////////////////////////////////////////////////////////////
3  // Engineer: Anjan Prasad
4  // Create Date: 11/11/2024 06:17:07 AM
5  // Module Name: SR_flipflop
6  //////////////////////////////////////////////////////////////////////////////////
7
8
9  module SR_flipflop (
10     input S, R, clk, reset,
11     output reg Q
12 );
13     always @(posedge clk or posedge reset) begin
14         if (reset)
15             Q <= 1'b0;
16         else if (S && !R)
17             Q <= 1'b1;
18         else if (!S && R)
19             Q <= 1'b0;
20     end
21 endmodule
22
```

JK_flipflop.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/project_1/project_1.srcs/sources_1/new/JK_flipflop.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Engineer: Anjan Prasad
4  // Create Date: 11/11/2024 06:21:51 AM
5  // Module Name: JK_flipflop
6  //////////////////////////////////////////////////////////////////////////////////
7
8
9  module JK_flipflop (
10     input J, K, clk, reset,
11     output Q
12 );
13     wire S, R;
14
15     assign S = J & ~Q; // Set when J=1 and Q=0
16     assign R = K & Q;  // Reset when K=1 and Q=1
17
18     SR_flipflop sr_ff (.S(S), .R(R), .clk(clk), .reset(reset), .Q(Q));
19 endmodule
20
```

D_flipflop.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/project_1/project_1.srcs/sources_1/new/D_flipflop.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Engineer: Anjan Prasad
4  // Create Date: 11/11/2024 06:21:51 AM
5  // Module Name: D_flipflop
6  //////////////////////////////////////////////////////////////////////////////////
7
8
9  module D_flipflop (
10     input D, clk, reset,
11     output Q
12 );
13     wire S, R;
14
15     assign S = D;
16     assign R = ~D;
17
18     SR_flipflop sr_ff (.S(S), .R(R), .clk(clk), .reset(reset), .Q(Q));
19 endmodule
20
```

T_flipflop.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/project_1/project_1.srcs/sources_1/new/T_flipflop.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Engineer: Anjan Prasad
4  // Create Date: 11/11/2024 06:21:51 AM
5  // Module Name: T_flipflop
6  //////////////////////////////////////////////////////////////////////////////////
7
8
9  module T_flipflop (
10     input T, clk, reset,
11     output Q
12 );
13     wire S, R;
14
15     assign S = T & ~Q; // Toggle logic: set when T=1 and Q=0
16     assign R = T & Q;  // Toggle logic: reset when T=1 and Q=1
17
18     SR_flipflop sr_ff (.S(S), .R(R), .clk(clk), .reset(reset), .Q(Q));
19 endmodule
20
```

SR_conversion_tb.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_051/project_1/project_1.srcs/sim_1/new/SR_conversion_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Anjan Prasad
// Create Date: 11/11/2024 06:29:36 AM
// Module Name: SR_conversion_tb
//////////////////////////////////////////////////////////////////////////////////
module SR_conversion_tb;
    reg J, K, D, T, clk, reset;
    wire q_jk, q_d, q_t;

    SR_conversion DUT (.J(J),.K(K),.D(D),.T(T),.clk(clk),.reset(reset),.q_jk(q_jk),.q_d(q_d),.q_t(q_t));
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
    initial begin
        reset = 1; J = 0; K = 0; D = 0; T = 0;
        #10 reset = 1;
        #10 reset = 0;
        // Test JK Flip-Flop
        $display("Testing JK Flip-Flop");
        J = 0; K = 0; #10; // Hold
        J = 1; K = 0; #10; // Set
        J = 0; K = 1; #10; // Reset
        J = 1; K = 1; #10; // Toggle
        $display("q_jk: %b", q_jk);
        #20 $stop;
    end
    initial begin
        // Test D Flip-Flop
        $display("Testing D Flip-Flop");
        D = 0; #30; // Set D to 0
        D = 1; #10; // Set D to 1
        D = 0; #10; // Set D back to 0
        $display("q_d: %b", q_d);
        #20 $stop;
    end
    initial begin
        // Test T Flip-Flop
        $display("Testing T Flip-Flop");
        T = 0; #10; // Hold
        T = 1; #10; // Toggle
        T = 1; #10; // Toggle again
        $display("q_t: %b", q_t);

        // Final hold state
        $display("Final States - q_jk: %b, q_d: %b, q_t: %b", q_jk, q_d, q_t);
        #20 $stop;
    end
endmodule
```