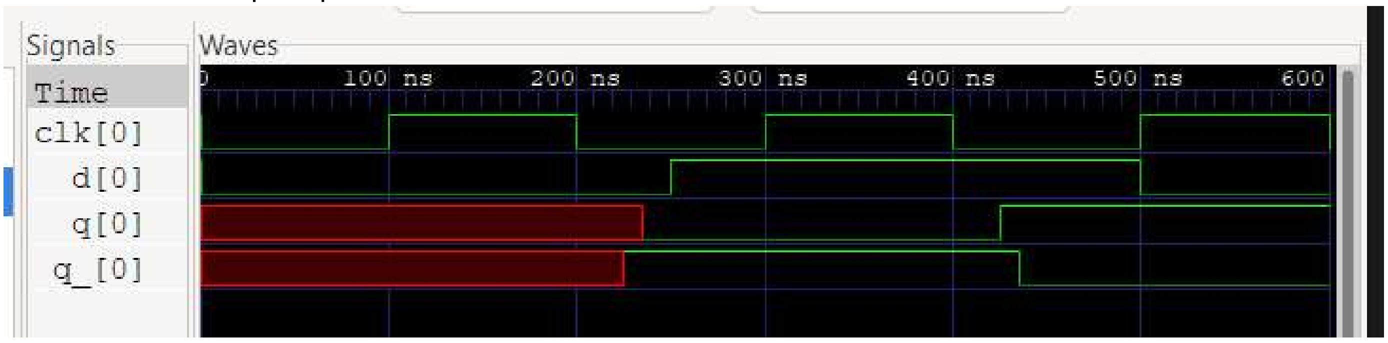
DDLab project 5

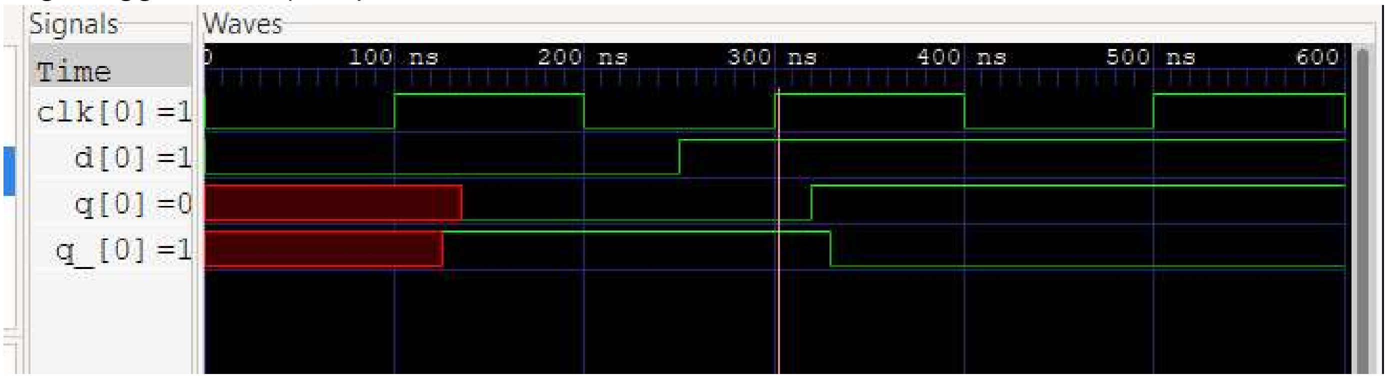
```verilog
`timescale 1ns/100ps

module SRFF (
    s,r,q,q_
);
    output q, q_;
    input s,r;

    nand #10 a(q,s,q_);
    nand #10 b(q_,r,q);
endmodule

module t_SRFF (

);
    wire q,q_;
    reg s,r;

    SRFF sr(s,r,q,q_);

    initial begin
        s=1'b0;r=1'b1;
        #100 s=1'b1;r=1'b0;
        #100 s=1'b1;r=1'b1;
    end
    // initial $dumpvars;
    // initial #300 $finish;
endmodule

module D_latch (
    d,en,q,q_
);
    input d,en;
    output q,q_;

    wire d_,s,r;
    not #5 n(d_,d);
    nand #10 a(s,d,en);
    nand #10 b(r,d_,en);
    SRFF sr(s,r,q,q_);
endmodule

module t_D_latch (

);
    wire q,q_;
    reg d,en;

    D_latch d_latch(d,en,q,q_);

    initial begin
        en=1'b1;d=1'b0;
        #100 d=1'b1;
        #100 d=1'b0;
        #100 en=1'b0;
        #100 d=1'b1;
        #100 d=1'b0;
    end
    initial $dumpvars;
    // initial #600 $finish;
endmodule

module msDFF (
    d,clk,q,q_
);
    input d,clk;
    output q,q_;

    wire d_,y;
    not #5 (clk_,clk);
    D_latch m(.d(d),.en(clk),.q(y));
    D_latch s(y,clk_,q,q_);

endmodule

module t_msDFF (

);
    wire q,q_;
    reg d,clk;

    msDFF msdff(d,clk,q,q_);

    initial begin
        clk = 1'b0;
        repeat(7)
            #100 clk = clk + 1'b1;
    end
    initial begin
        d = 1'b0;
        #250 d=1'b1;
        #250 d=1'b0;
    end
    initial $dumpvars;
    // initial #600 $finish;
endmodule

module etDFF (
    d,clk,q,q_
);
    input d,clk;
    output q,q_;

    wire w1,ws,wr;

    SRFF sr1(.q_(ws),.s(w1),.r(clk));
    SRFF sr2(ws,wr,q,q_);

    nand #15 (wr,ws,clk,w1);
    nand #10 (w1,d,wr);
endmodule

module t_etDFF (
);
    wire q,q_;
    reg d,clk;

    etDFF etdff(d,clk,q,q_);

    initial begin
        clk = 1'b0;
        repeat(7)
            #100 clk = clk + 1'b1;
    end
    initial begin
        d = 1'b0;
        #250 d=1'b1;
        // #250 d=1'b0;
    end
    initial $dumpvars;
    // initial #600 $finish;
endmodule

module bhDFF (
    d,clk,q,q_
);
    input d,clk;
    output reg q,q_;

    always @(posedge clk ) begin
        q <= d;
        q_ <= ~d;
    end
endmodule

module t_bhDFF (

);
    wire q,q_;
    reg d,clk;

    bhDFF bhdff(d,clk,q,q_);

    initial begin
        clk = 1'b0;
        repeat(7)
            #100 clk = clk + 1'b1;
    end
    initial begin
        d = 1'b0;
        #250 d=1'b1;
        // #250 d=1'b0;
    end
    initial $dumpvars;
    // initial #600 $finish;
endmodule

module bhDFF_negedge (
    d,clk,q,q_
```

```
);
    input d,clk;
    output reg q,q_;

    always @(negedge clk ) begin
        q <= d;
        q_ <= ~d;
    end
endmodule

module set_up_and_hold_time (

);
    wire ms_q,ms_q_,et_q,et_q_,bh_q,bh_q_,bhn_q,bhn_q_;
    reg d,clk;

    msDFF msdff(d,clk,ms_q,ms_q_);
    etDFF etdff(d,clk,et_q,et_q_);
    bhDFF bhdff(d,clk,bh_q,bh_q_);
    bhDFF_negedge bhdff_n(d,clk,bhn_q,bhn_q_);

    initial begin
        clk = 1'b0;
        repeat(20)
            #100 clk = clk + 1'b1;
    end
    initial begin
        d = 1'b0;
        #398 d=1'b1; //ms setup violation
        #352 d=1'b0;
        #51 d=1'b1; //ms hold violation
        #298 d=1'b0; //et setup violation
        #402 d=1'b1; //et hold violation
    end
    initial $dumpvars;
    initial #2100 $finish;

endmodule
```

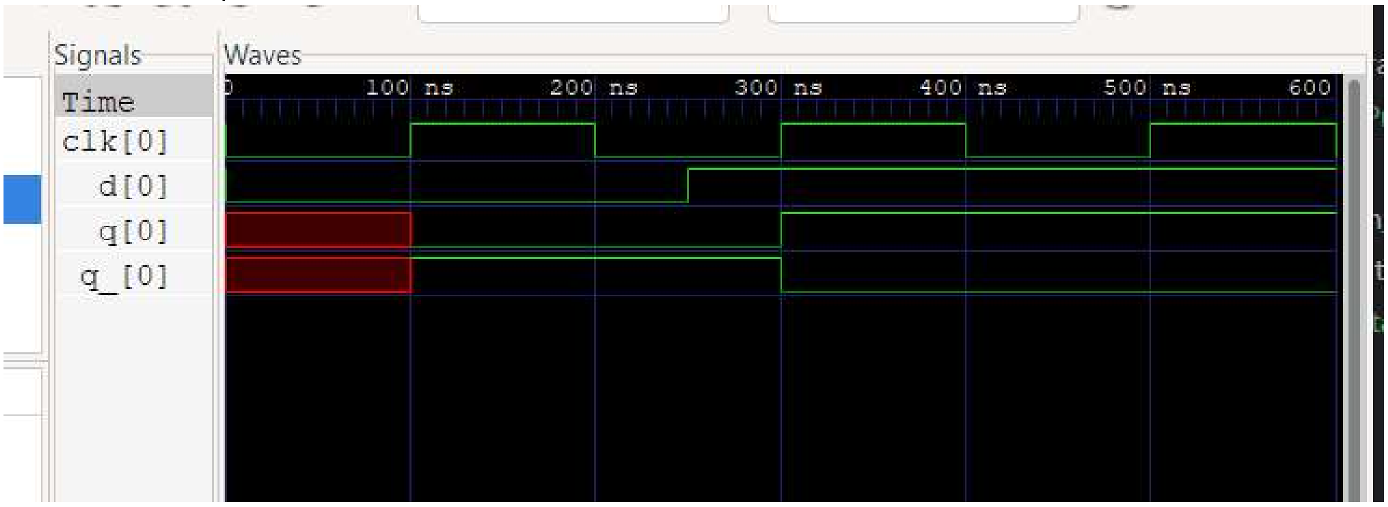
master-slave D flip-flop



edge-triggered D flip-flop

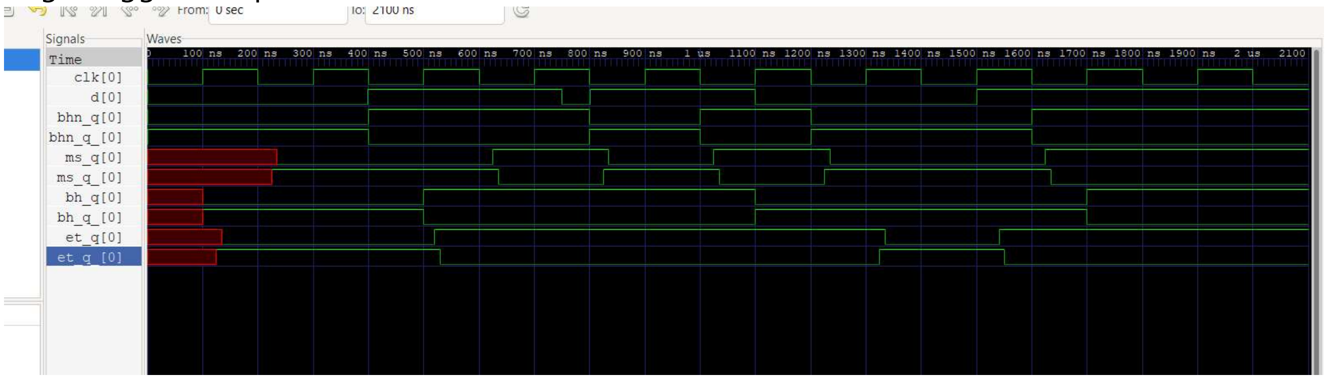


behavior description



setup time and hold time violations

- master-slave setup time violation: ~400ns
- master-slave hold time violation: ~800ns (no hold time)
- edge-trigger setup time violation: ~1100ns
- edge-trigger setup time violation: ~1500ns



類型	**setup time (ns)**	**原因**	**hold time (ns)**	**原因**
master-slave	>30ns	master D 端需要 35 ns 才能輸出至 Y 端，而 slave En 在 clk 從 1 -> 0 5ns 後便會開始運作， 因此為了確保 slave 收到正確的值，setup time 應大於 35 - 5 = 30 (ns)	>0ns	clk 從 1 -> 0 後，master 第一級的 NAND 輸出就會固定在 1， 此時不論如何更動 D 的值都不會影響後續處理及輸出結果
edge-trigger	>20ns	D 變動後，要經過 20 ns clk 所連接的兩個 NAND 的全部輸入才會都穩定	>25ns	D 連接的 NAND 的輸出要等 clk 變化後 25 ns 才會穩定，在這之前 D 的改變都有可能影響輸出