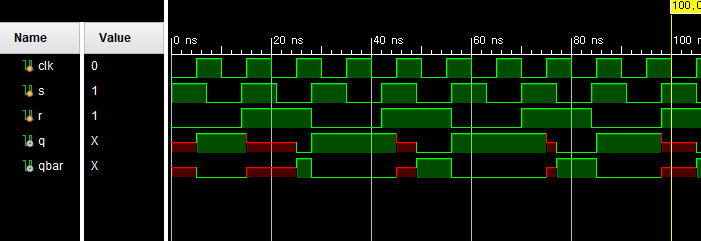
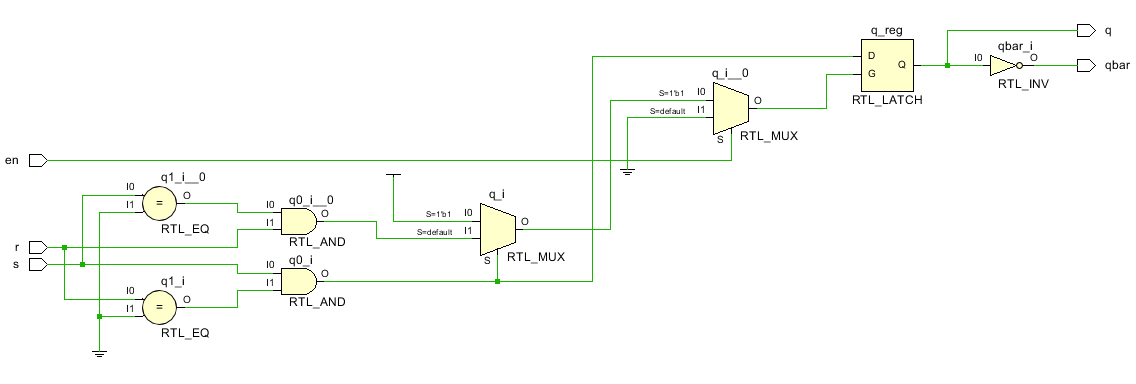
Sim/code/tt/fpga

SR

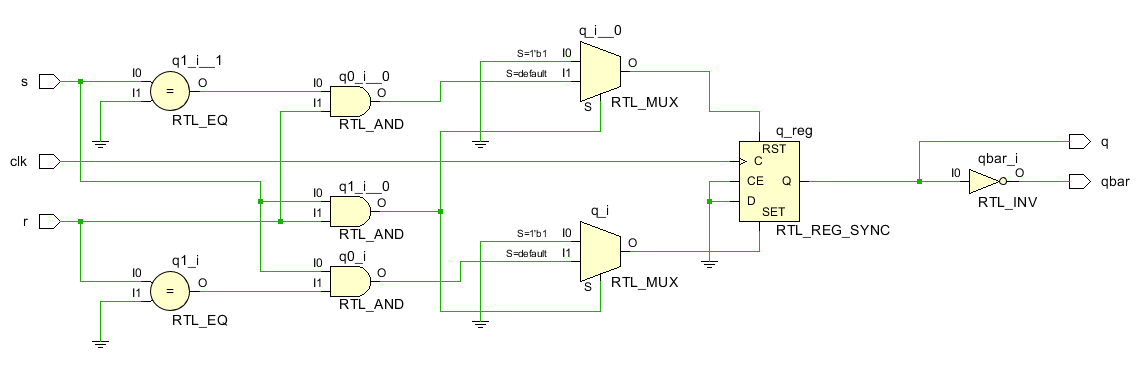
Latch

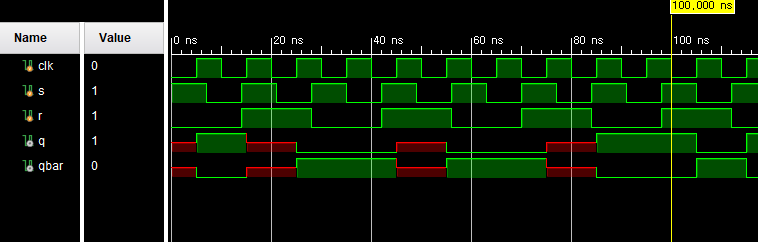




|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| 입력 순서 | R | S | Q | ~Q |
| (1) | 0 | 1 | 1 | 0 |
| (2) | 0 | 0 | 1 | 0 |
| (3) | 1 | 0 | 0 | 1 |
| (4) | 0 | 0 | 0 | 1 |
| (5) | 1 | 0 | 0 | 1 |
| (6) | 1 | 1 | x | x |

FlipFlop

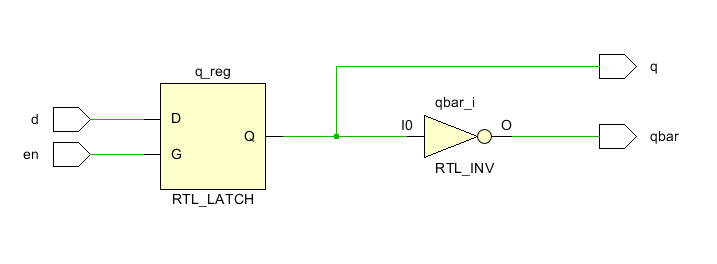


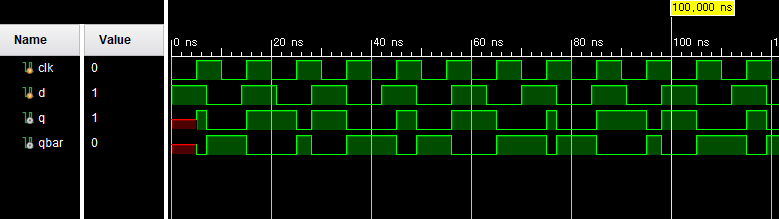


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| 입력 순서 | R | S | Q | ~Q |
| (1) | 0 | 1 | 1 | 0 |
| (2) | 0 | 0 | 1 | 0 |
| (3) | 1 | 0 | 0 | 1 |
| (4) | 0 | 0 | 0 | 1 |
| (5) | 1 | 0 | 0 | 1 |
| (6) | 1 | 1 | x | x |

D

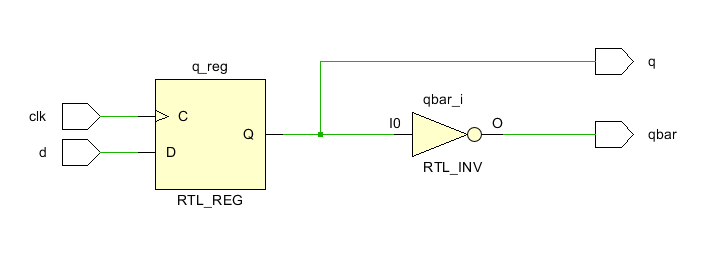
Latch

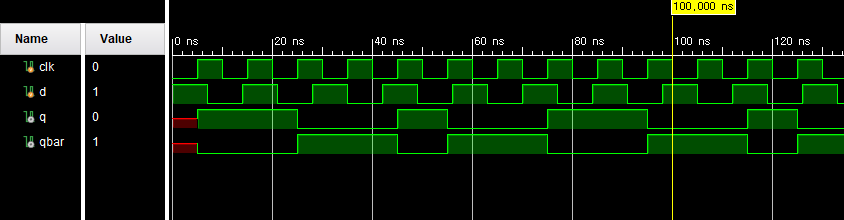




|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| D | E | Q | ~Q |
| 0 | 0 | N/A | N/A |
| 0 | 1 | 0 | 1 |
| 1 | 0 | N/A | N/A |
| 1 | 1 | 1 | 0 |

Flipflop





`timescale 1ns / 1ps

module SRLatch(

en, s, r, q, qbar

);

input en, s, r;

output q, qbar;

reg q;

assign qbar = ~q;

always @(en or s or r) begin

if(en) begin

if((s==1)&&(r==1))

q <= 1'bx;

else if((s==1)&&(r==0))

q <= 1'b1;

else if((s==0)&&(r==1))

q <= 1'b0;

else

q <= q;

end

end

endmodule

module srlatch (

input S,

input R,

input En,

output reg Q,

output reg Qc

);

always @(\*) begin

if (En) begin

Q = ~(R | Qc);

Qc = ~(S | Q);

end

end

endmodule : srlatch

module SRFlipFlop(

clk, s, r, q, qbar

);

input clk, s, r;

output q, qbar;

reg q;

assign qbar = ~q;

always @(posedge clk) begin

if((s==1)&&(r==1))

q <= 1'bx;

else if((s==1)&&(r==0))

q <= 1'b1;

else if((s==0)&&(r==1))

q <= 1'b0;

else

q <= q;

end

endmodule

module DLatch(

en, d, q, qbar

);

input en, d;

output q, qbar;

reg q;

assign qbar = ~q;

always @(en or d) begin

if(en)

q <= d;

end

endmodule

module DFlipFlop(

clk, d, q, qbar

);

input clk, d;

output q, qbar;

reg q;

assign qbar = ~q;

always @(posedge clk) begin

q <= d;

end

endmodule

`timescale 1ns / 1ps

module srlatchsim;

reg clk, s, r;

wire q, qbar;

SRLatch srlatch(clk, s, r, q, qbar);

initial clk=1'b0;

initial s=1'b1;

initial r=1'b0;

always clk = #5 ~clk;

always s = #7 ~s;

always r = #14 ~r;

initial begin

#1600

$finish;

end

endmodule

module srffsim;

reg clk, s, r;

wire q, qbar;

SRFlipFlop srflipflop(clk, s, r, q, qbar);

initial clk=1'b0;

initial s=1'b1;

initial r=1'b0;

always clk = #5 ~clk;

always s = #7 ~s;

always r = #14 ~r;

initial begin

#1600

$finish;

end

endmodule

module dlatchsim;

reg clk, d;

wire q, qbar;

DLatch dlatch(clk, d, q, qbar);

initial clk=1'b0;

initial d=1'b1;

always clk = #5 ~clk;

always d = #7 ~d;

initial begin

#1600

$finish;

end

endmodule

module dffsim;

reg clk, d;

wire q, qbar;

DFlipFlop dff(clk, d, q, qbar);

initial clk=1'b0;

initial d=1'b1;

always clk = #5 ~clk;

always d = #7 ~d;

initial begin

#1600

$finish;

end

endmodule