7.

Version 1

design

```
module Hw7_7(DA,DB,A,B,x);
output DA,DB;
input A,B,x;

assign DA = (A & (~x)) | ((~B) & x);
assign DB = (A & x) | (B & (~x));

endmodule
```

Testbench

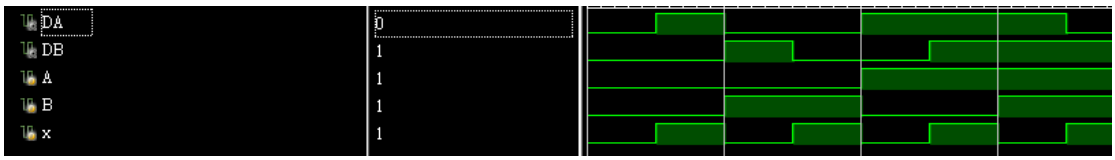
```
module test;
wire DA,DB;
reg A,B,x;

Hw7_7 UO(.DA(DA),.DB(DB),.A(A),.B(B),.x(x));

initial begin
A=0; B=0; x=0;
#10 A=0; B=0; x=1;
#10 A=0; B=1; x=0;
#10 A=0; B=1; x=1;
#10 A=1; B=0; x=0;
#10 A=1; B=0; x=1;
#10 A=1; B=1; x=0;
#10 A=1; B=1; x=1;

end
endmodule
```

Waveform



Version2

design

```
module Hw7_7_v2(next_state,state,x,clk,rst_n);

output reg [1:0] next_state;
output reg [1:0] state;
input x;
input clk,rst_n;

always@(posedge clk or negedge rst_n)
    if(~rst_n)
        state <= 2'b00;
    else
        state <= next_state;

always@*
    case(state)
    2'b00: next_state = x?2'b10:state;
    2'b01: next_state = x?2'b00:state;
    2'b10: next_state = x?2'b11:state;
    2'b11: next_state = x?2'b01:state;
    default: next_state = state;
    endcase

endmodule
```

testbench

```
module test;

reg clk,rst_n;
reg x;
wire [1:0] next_state;
wire [1:0] state;

Hw7_7_v2 U1(.next_state(next_state),.state(state),.x(x),.clk(clk),.rst_n(rst_n));

always
 #5 clk = ~clk;

initial begin
clk=0; rst_n=1; x=0;
#4 rst_n=0;
#6 rst_n=1;
#10 x=1;
#10 x=0;
#10 x=1;
#10 x=0;
#10 x=1;
#10 x=0;
#10 x=1;
#10 x=0;
#10 x=1;
end

endmodule
```

waveform