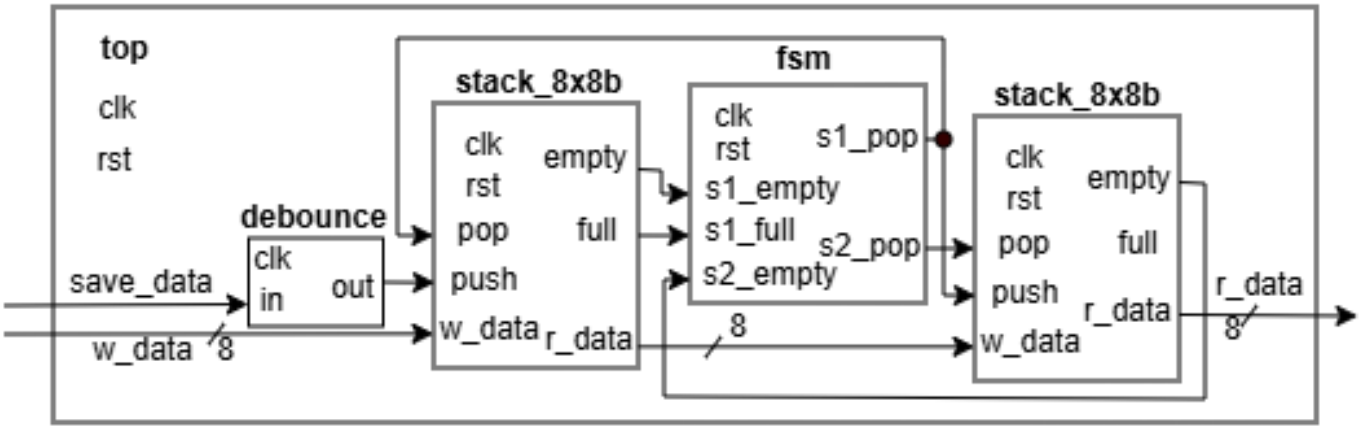
Subiect:

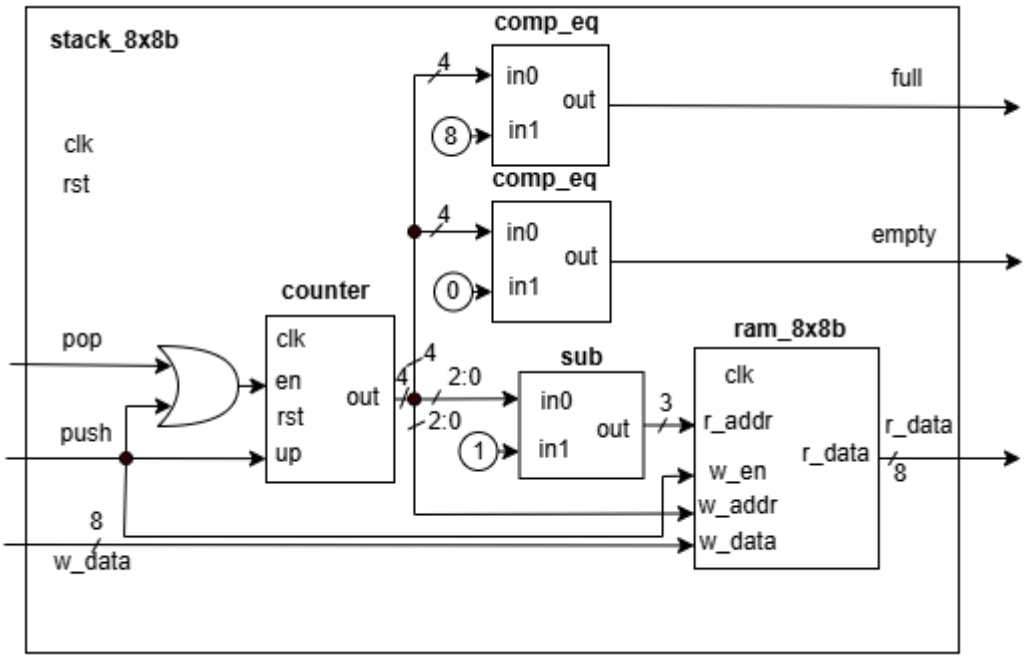
Timp de lucru: 1.20h

Cerinta:

Implementati schema de mai jos in System Verilog, efectuati simulari si testare fizica pentru verificarea functionalitati.

Schema bloc:





Descrierea schemei + cerinte speciale:

Circuitul de mai sus este o coada obtinuta prin 2 stive.

Datele sunt salvate prin activarea intrarii “save_data”. Golirea cozii se face dupa ce toate cele 8 locatii ale acesteia au fost scrise.

Circuitul cuprinde:

comp_eq - comparator de egalitate.

counter – numarator pe 4b

sub – scazator

ram_8x8b – memorie ram cu citire asincrona si dimensiunea 8 locatii de cate 8b fiecare

debounce – modul de debounce (se da)

fsm – automat ce are rolul de a coordona intreg sistemul.

Acesta va fi implementat ca un automat Mealy fara intarziere.

Acesta va avea 3 stari: incarcarea primei stive, mutarea datelor din stiva 1 in stiva 2, golirea stivei 2.

Cele 2 iesiri (ce comanda operatia de pop a stivei) se activeaza doar in starea corespunzatoare si doar daca stiva pe care o comanda nu este goala.

Pentru simplificare desenului, firul de clock si firul de reset nu a fost desenat efectiv. Sa nu uitati sa il puneti.

Pentru simulare:

Pentru testarea functionalitati se vor introduce datele 1,2,3,4,5,6,7,8 si apoi se va observa iesirea acestora in ordinea corecta.

Atentie: modulul de debounce cere 650_000 ciclii de ceas pentru a inregistra o activare a semnalului “save_data”. Cand faceti operatiile de write fie modificati limita sa fie mai mica, fie lasati suficient timp in tb cu semnalul activ.

Pentru sinteza:

clk – Clock 100MHz

rst – BTN[0]

save_data – BTN[1]

w_data – SW[7:0]

r_data – LED[7:0]

Barem:

Total - 25p

design -18p

- top – 3p
- stack_8x8b – 3p
- fsm – 5p
- debounce – 0p (se da)
- counter – 3p
- comp_eq- 1p
- sub – 1p
- ram_8x8b – 2p

simulare - 5p

- testbench - 3p
- demonstratie simulare - 2p.

- 2p - poza forme de unda cu toate modulele puse in simulare ca grupuri si semnalele din tb vizibile clar, iesirea finala.

placa – 2p

- 1 constrangeri
- 1 demonstratie si intrebari

Module date

```
module debounce(
        input clock,
        output out,
        input in
);
parameter limit = 20'd650000;
reg [19:0] counter;
reg hit;
assign out = (counter == limit);
always@(posedge clock) begin
    if(!in) begin
            counter <= 0;
            hit <= 0;
    end else if(counter == limit) begin
            hit <= 1;
            counter <= counter + 1;
        end else if(in & !hit) begin
            counter <= counter + 1;
        end
end

endmodule
```