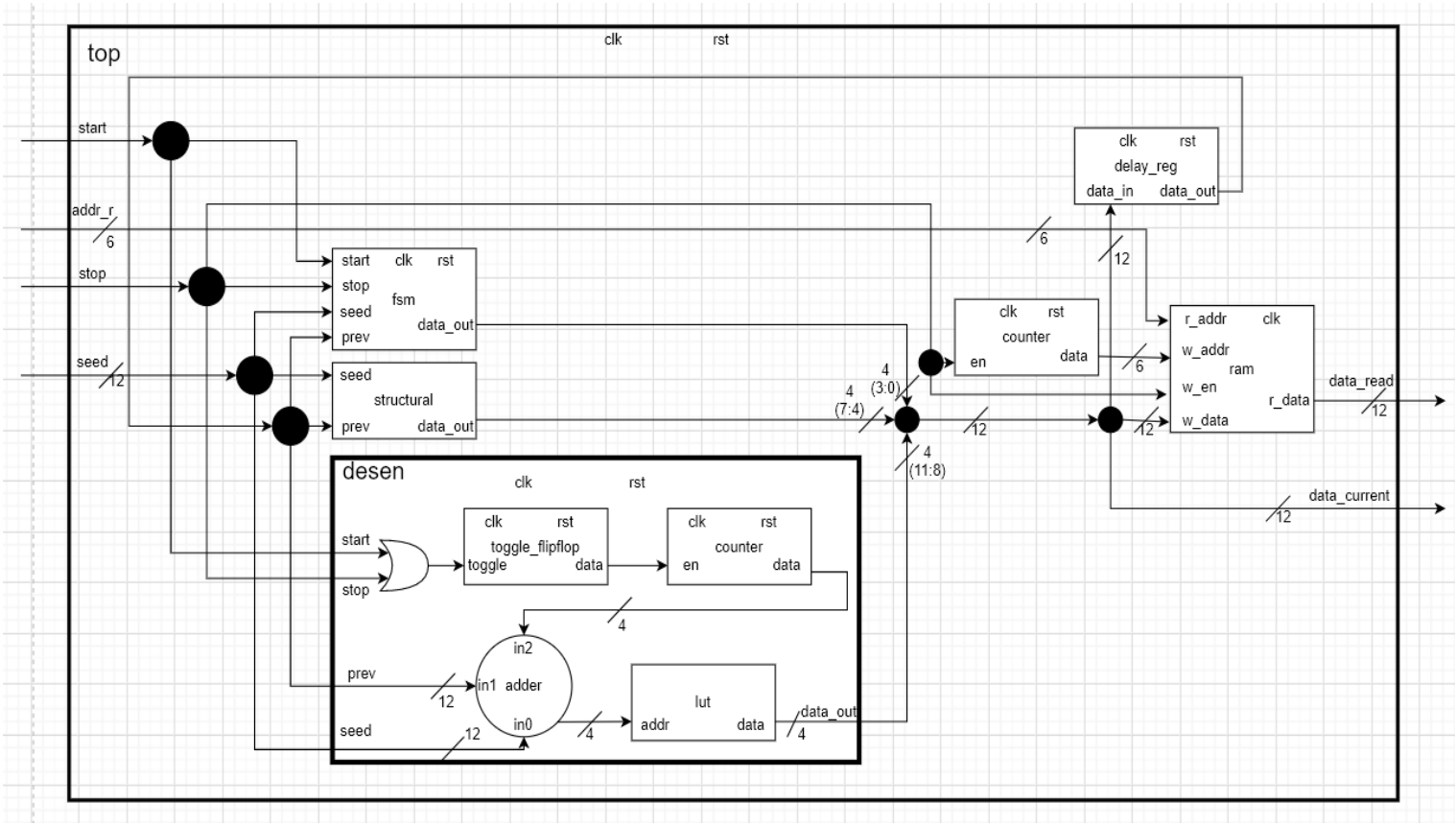
## Subiect:

## Timp de lucru: 1.40h

## Cerinta:

Implementati schema de mai jos in verilog si efectuati simulari pentru verificarea functionalitati.

## Schema bloc:




## Descrierea schemei + cerinte speciale:

Circuitul de mai sus este un generator de numere pseudoaleatoare pe 12 biti. Numerele sunt alcatuita din 3 seturi de 4biti generate independent si apoi concatenate (firul data current).

Valorile din acest sir se vor salva intr-o memorie ram ca sa poata fi apoi citite din sistem.

Sistemul este alcauit din urmatoarele blocuri/module:

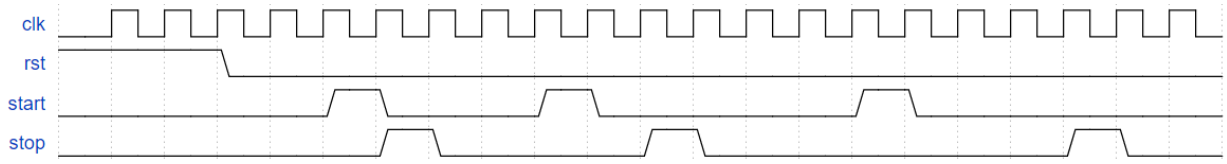
- delay_reg - registru de intarziere cu un ciclu de ceas. folosit pentru sincronizarea semnalelor
- counter - numarator
- ram - memorie de tip ram, asincrona
- toggle_flipflop - registru de toggle, isi schimba valoarea cand la intrare semnalul de toggle este activ
- lut - look up table - implementat ca o memorie de tip rom. Datele de la iesire sunt inversul pe biti ai datelor de la intrare (ex: la addr 0011 se va afla data 1100, la addr 0100 se va afla data 0010)
- desen - modul de tip wrapper, folosit pentru instantiere mai usoara in top. Acesta va calcula bitii 11:8 din numarul pseudorandom generat.
- adder - sumator
- structural - modul ce calculeaza bitii 7:4 din numarul pseudorandom generat, **implementat folosind o descriere structurala** in Verilog. Implementarea trebuie sa realizeze urmatoarea functionalitate: in functie de ultimii 2 biti din semnalul "seed", iesirea va consta in bitii [3:0] (pt val 0), [7:4] (pt val 1), [11:8] (pt val 2) ai numarului precedent generat sau in ultimii 4 biti ai numarului 2*x + x + 4 (unde x e numarul precedent generat) (pt val 3).
- fsm - modul ce calculeaza bitii 3:0 din numarul pseudorandom generat**, implementat ca un automat**. El genereaza numarul dupa urmatoarele reguli:
    - La reset, iesirea are valoarea ultimilor 4b din seed
    - La aparitia semnalului de start el incepe pe fiecare ciclu de ceas sa adune valoarea curenta (inclusiv ciclu cand apare start) peste valoarea curenta.
    - La aparitia semnalului de stop nu se mai efectueaza adunari, iesirea ramanand stabila (inclusiv pe ciclul de ceas cand e vazut stop nu se mai efectueaza adunari)
- top - modulul in care se instantiaza restul modulelor si implementeaza functionalitatea finala

Pentru simplificare desenului, firele de clock si reset nu au fost desenate efectiv. Sa nu uitati sa le puneti.

Poarta or, sumatoarele, registrul de intarziere, nu trebuie obligatoriu declarate ca modul separat. Restul da.

## Pentru simulare:

Pentru testarea functionalitati se va lasa circuitul sa functioneze pentru 400 de cicli de ceas, primii, ciclii respectand figura:




Liniile punctate reprezinta 10 unitati de timp.

Semnalul addr_r se genereaza astfel incat sa puteti citi din memoria ram o valoare salvata anterior.

Seed-ul se alege aleator (trebuie pus).

Pentru generarea altor valori (daca se doreste):

Start si stop nu pot fi active amandoua deodata pe acelasi ciclu de ceas.

Intre perechi diferite de start/stop vor fi minim 2 cicli de ceas cu ambele inactive.

## Barem:

Total - 30p

design - 20p

- top - 3p
- fsm - 4p
- structural - 3p
- desen - 2p
- toggle_flipflop - 1p
- counter_desen - 1p
- lut - 2p
- counter_top - 1p
- delay_reg - 1p
- ram - 2p

simulare - 10p

- testbench - 5p
- demonstratie simulare - 5p.
    - 2p - poza forme de unda cu toate modulele puse in simulare ca grupuri si vizibile semnalele din tb, intrari si iesiri.

- 3p memoria ram cu valorie corecte salvate in aceasta vizibile (primele cate incap intrun screenshot) + valorile din tb deasupra ca sa le pot vedea usor.