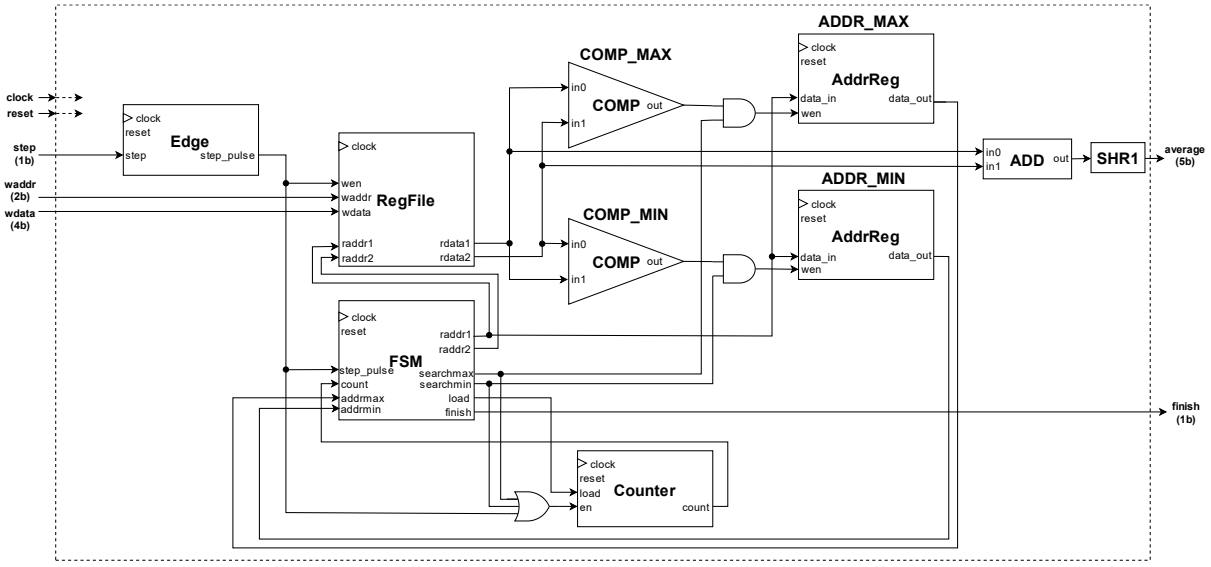
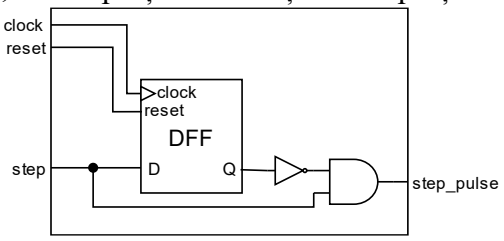
Media dintre cel mai mic și cel mai mare număr

În schema de mai jos este prezentat un circuit ce parcurge numerele stocate într-un RegFile și calculează media dintre cel mai mic și cel mai mare element:



Obs: Semnalele *clock* și *reset* ale submodulelor se leagă la *clock* și *reset* ale modulului TOP. Reset este sincron, activ in 1.

1) **(20 p)** Implementați circuitul, păstrând denumirea modulelor (în interiorul chenarelor) și a semnalelor de intrarea și ieșire. Pentru aceasta, va trebui să implementați în fișiere separate următoarele module:
   - **(2p) Edge**: modul ce generează la ieșirea sa (*step_pulse*) un puls de lungime maxim o perioadă de ceas pentru fiecare front crescător al intrării *step*. Implementarea se va face cu ajutorul unui bistabil, a unei porți inversor și a unei porți AND, astfel:



   - **(2p) RegFile**: register file cu 4 locații de 4 biți și două căi de citire (*rdata1*, corespunzătoare *raddr1* și *rdata2*, corespunzătoare *raddr2*), ambele asincrone. Scrierea se face doar dacă *wen* este 1.
   - **(1p) ADD**: sumator pe 4 biți, cu ieșirea pe 5 biți (ieșirea include carry).
   - **(1p) SHR1**: circuit de deplasare la dreapta cu o poziție a unui șir de 5 biți.
   - **(1p) AddrReg**: registru pe 2 biți, cu enable (scrierea se va face doar dacă *en* este 1). Reset este sincron, activ în 1, iar valoarea de reset este 0. Acest modul va avea două instanțe în TOP, ADDR_MAX și ADDR_MIN.
   - **(1p) COMP**: modul ce realizează compararea a două numere pe 4 biți. Acesta va genera 1 pe ieșire dacă in0 > in1. Acest modul va avea două instanțe în TOP, COMP_MAX și COMP_MIN.

- **(2p) Counter**: numărător pe 2 biți, cu enable (valoarea sa se **decrementează** doar dacă *en* este 1) și load. La reset și la load, numărătorul se va încărca cu valoarea 3. Semnalul load are prioritate peste enable.
- **(4p) FSM**: automat finit cu 4 stări ce controlează funcționarea circuitului:
    - Starea **LD_RAM**: stare de reset în care se realizează scrierea memoriei RAM. Automatul va trece din această stare în **SEARCHMAX**, dacă valoarea *count* este 0 și *step_pulse* este 1.
    - Starea **SEARCHMAX**: stare în care se realizează căutarea maximului. Automatul va trece din această stare în **SEARCHMIN**, dacă valoarea *count* este 0.
    - Starea **SEARCHMIN**: stare în care se realizează căutarea minimului. Automatul va trece din această stare în **FINISH**, dacă valoarea *count* este 0.
    - Starea **FINISH**: stare în care se semnalizează terminarea căutării maximului și a minimului și calculul mediei. Automatul va rămâne blocat în această stare, indiferent de starea intrărilor.

  Ieșirile automatului sunt:
    - ***raddr1***: copiază valoarea *addrmax* dacă starea automatului este FINISH și *count* în rest.
    - ***raddr2***: copiază valoarea *addrmax* dacă starea automatului este SEARCHMAX și valoarea *addrmin* în rest.
    - ***searchmax***: este 1 doar în starea SEARCHMAX și 0 în rest.
    - ***searchmin***: este 1 doar în starea SEARCHMIN și 0 în rest.
    - ***load***: este 0 pe tot parcursul funcționării automatului.
    - ***finish***: este 1 doar în starea FINISH și 0 în rest.

- **(6p) TOP**: modul în care sunt instanțiate toate celelalte module și conectate conform schemei.

2) **(6p)** Implementați un modul de test prin care să se simuleze următorul comportament:
- Unitatea de timp va avea 1 ns.
- Resetați inițial toate componentele.
- Scrieți cele 4 locații de memorie ale RAM, astfel:
    - La adresa 0, valoarea 7.
    - La adresa 1, valoarea 15.
    - La adresa 2, valoarea 1.
    - La adresa 3, valoarea 2.

  Pentru fiecare scriere, datele de intrare și adresa RAM vor trebui menținute constante 12ns. În timpul acestor 12ns se va genera un puls pe *step* de 3ns, între nanosecunda 2 și nanosecunda 5.
- După scrierea completă a memoriei, lăsați simularea să ruleze încă 100ns.

Faceți un printscreen la o forma de undă care să conțină următoarele componente: *clock*, *reset, step*, *waddr, wdata, average, finish,* ieșirile registrelor ADDR_MAX si ADDR_MIN, registrul de stare al FSM și memoria RegFile. Forma de undă trebuie să aibă un nivel potrivit de zoom astfel încât să reiasă din aceasta funcționarea corectă a circuitului.

3) **(4p)** Sintetizați circuitul, astfel încât porturile să fie conectate la următoarele dispozitive:

clock	CLOCK_100MHz
reset	BTN0
step	BTN3
waddr[0]	SW0
waddr[1]	SW1
wdata[0]	SW12
wdata[1]	SW13
wdata[2]	SW14
wdata[3]	SW15
average [0]	LED0
average [1]	LED1
average [2]	LED2
average [3]	LED3
average [4]	LED4
finish	LED15

Demonstrați funcționarea corectă a circuitului implementat pe FPGA