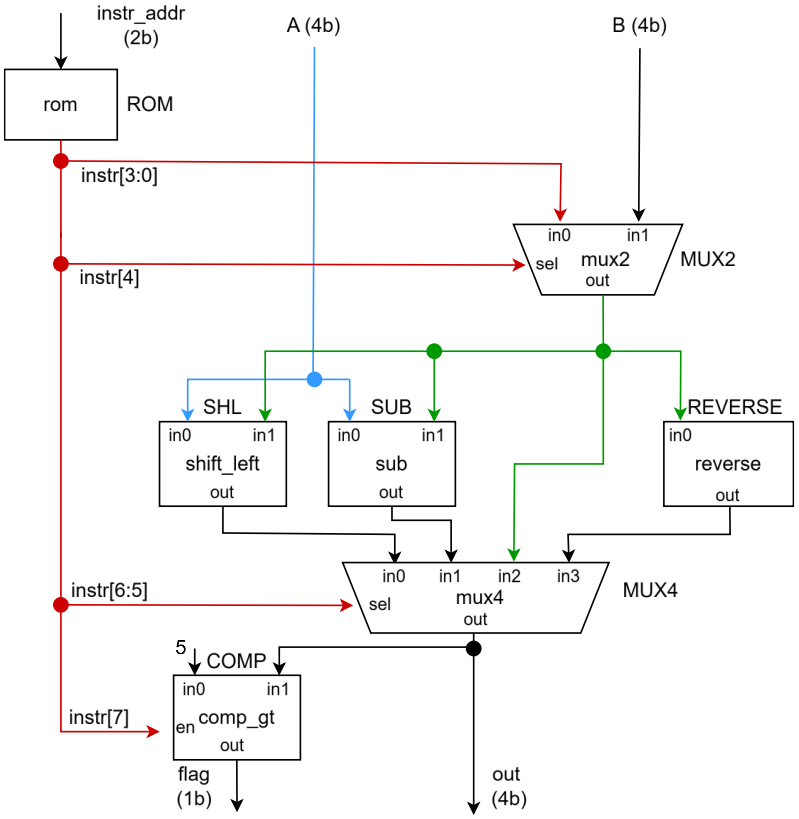
Unitate aritmetico-logică (ALU)

Fie circuitul prezentat în schema de mai jos:



1) **(12p)** Implementați circuitul, păstrând denumirea modulelor (în interiorul chenarelor), a porturilor și a instanțelor (în exteriorul chenarelor). Pentru aceasta, va trebui să implementați în fișiere separate următoarele module:
   - **(1p) mux4**: multiplexor cu patru intrări pe 4 biți.
   - **(1p) mux2**: multiplexor cu două intrări pe 4 biți.
   - **(1p) sub**: circuit ce realizează scăderea *in0 – in1*.
   - **(1p) shift_left**: circuit pentru deplasarea la stânga cu intrări pe 4 biți (*in0* este intrarea ce va fi deplasată, iar *in1* va da numărul de biți cu care se va face deplasare). Deplasarea se va face reținând doar numărul necesar de biți din *in1*.
   - **(1p) reverse**: circuit ce realizează inversarea biților intrării *in0,* astfel: bitul 0 al lui *out* va fi *in0[3]*, bitul 1 al lui *out* va fi *in0[2]*, bitul 2 al lui *out* va fi *in0[1]*, bitul 3 al lui *out* va fi *in0[0]*.
   - **(1p) comp_gt**: circuit de comparație cu enable. Acesta va da la ieșire 1 doar dacă *in1* > *in0* și *en* este 1. Altfel, ieșirea sa va fi 0.
   - **(2p) rom**: memorie ROM cu 4 locații de 8 biți, ce stochează codurile următoarelor instrucțiuni:
     Adresa 0: instrucțiunea 0 ce determină la ieșirea lui MUX4 out = reverse(B)
     Adresa 1: instrucțiunea 1 ce determină la ieșirea lui MUX4 out = A - B
     Adresa 2: instrucțiunea 2 ce determină la ieșirea lui MUX4 out = A << 2
     Adresa 3: instrucțiunea 3 ce determină la ieșirea lui MUX4 out = 14

Obs: Acolo unde apar constante (instrucțiunile 2 și 3), acestea sunt livrate la intrarea modulelor prin *instr[3:0]*. Comparația realizată de **comp_gt** va fi pornită pentru toate cele patru instrucțiuni.

- **(4p) TOP**: modul în care sunt instanțiate toate celelalte module și conectate conform schemei.

2) **(5p)** Implementați un modul de test prin care să se execute instrucțiunile de mai sus în ordine (instrucțiunea 0 -> 1 -> 2 -> 3). Păstrați valorile A = 2 și B = 8 constante pe toată durata simulării (se va varia doar intrarea *instr_addr*). Unitatea de timp va avea 1 ns, iar pasul cu care se va modifica *instr_addr* va fi de 3ns. Faceți un printscreen la o forma de undă care să conțină semnalele: *instr_addr*, *A, B, flag*, *out* și *instr* (ieșirea ROM). Forma de undă trebuie să aibă un nivel potrivit de zoom astfel încât să reiasă din aceasta funcționarea corectă a circuitului.

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

instr_addr[0]	SW0
instr_addr[1]	SW1
A[0]	SW8
A[1]	SW9
A[2]	SW10
A[3]	SW11
B[0]	SW12
B[1]	SW13
B[2]	SW14
B[3]	SW15
out[0]	LED0
out[1]	LED1
out[2]	LED2
out[3]	LED3
flag	LED14

Demonstrați și explicați funcționarea corectă a circuitului implementat pe FPGA