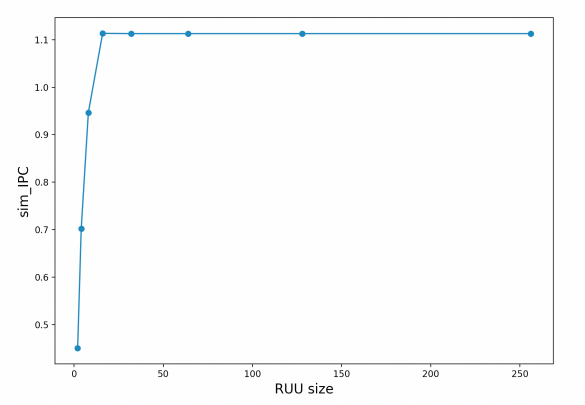
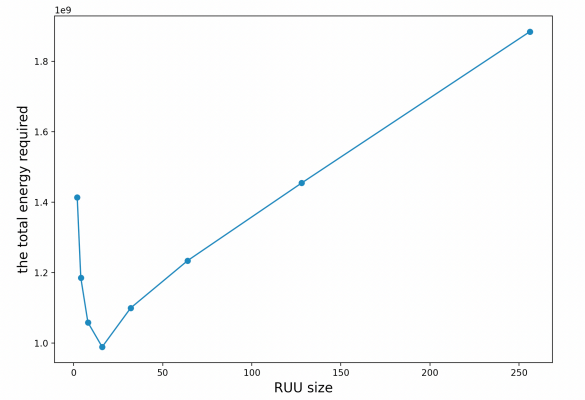
Vary the RUU size between 2 and 256 (only powers of two are valid):



sim_IPC: increases at first, the stay the same-> when the RUU is too small, making it bigger makes the program run much faster. But at some point, increasing RUU size stops helping.

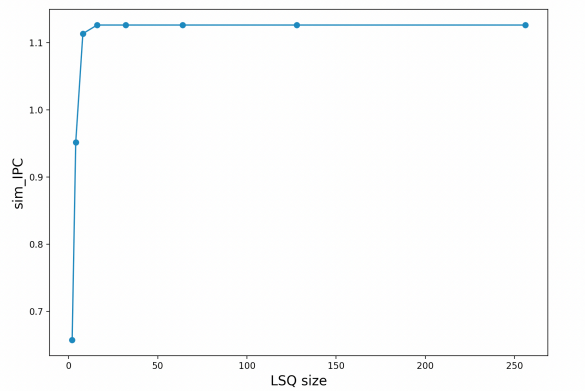


total energy consumed: decrease then increase -> increasing RUU size does cost energy. So making the RUU bigger than necessary leads to a fast execution time, but an unnecessarily high energy consumption

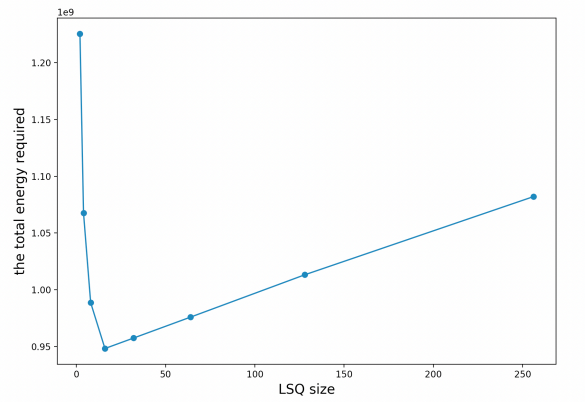
//the energy modeling mode we use by default assumes that all the available units are always on and running

//unravel the two factors: how much energy does it cost, and what effect does it have on execution time.

Vary the LSQ size between 2 and 256 (only powers of two are valid):



Sim_IPC: same as ruu



total energy consumed:same as ruu

What microarchitectural structure is limiting the simulated execution speed when running this application in the default simulated architecture? Under what circumstances is it the RUU size? Under what circumstanes is it the LSQ size?

-- RUU handles register synchronization/communication

– unifies reorder buffer and reservation stations • managed as a circular queue • entries allocated at Dispatch, deallocated at Commit

– out-of-order issue, when register and memory deps satisfied • memory dependencies resolved by load/store queue (LSQ)

LSQ handles memory synchronization/communication

– contains all loads and stores in program order • load/store primitives really, address calculation is separate op • effective address calculations reside in RUU (as ADD insts)

– loads issue out-of-order, when memory deps known satisfied • load addr known, source data identified, no unknown store address

when ruu <= lsq size, as ruu size increases, the speed increases upto when lsq size = ruu size.

when ruu>= lsq size, as lsq size increases, the speed increases upto when lsq size = ruu size

when ruu==lsq size, as ruu&lsq size increase, the speed increases

What combination of RUU size and LSQ size leads to the lowest total energy comsumption to complete the computation?

-- the lowest total energy comsumption: RUU size = LSQ size = 16

-> minimising execution time(min when ruu = 256, lsq = 128, the sim_IPC(indicate the exec time) here = 2.0637, but the energy comsuption here is extremely large(= 1499931696.4137)) does not minimise energy consumption(min when ruu = lsq = 16, the sim_IPC here = 1.1261).

find the simple-scalar configuration which finishes the computation in the minimum total energy:

-- from above, we know the lowest total energy comsumption: RUU size = LSQ size = 16

-- branch prediction: results are similar, not make too much diff -> keep it as perfect

-- # of integer ALUs: energy consumed decreases first then increase(choose 2)

--# of floating point ALUs: # increases, energy consumed increases(choose 1)

--# of floating point multipliers/ dividers: not much diff(default)

--# of integer multipliers/ dividers: no diff(default)

-- fetch/decode/issue width: