**Assumptions :**

Total vector length = 256

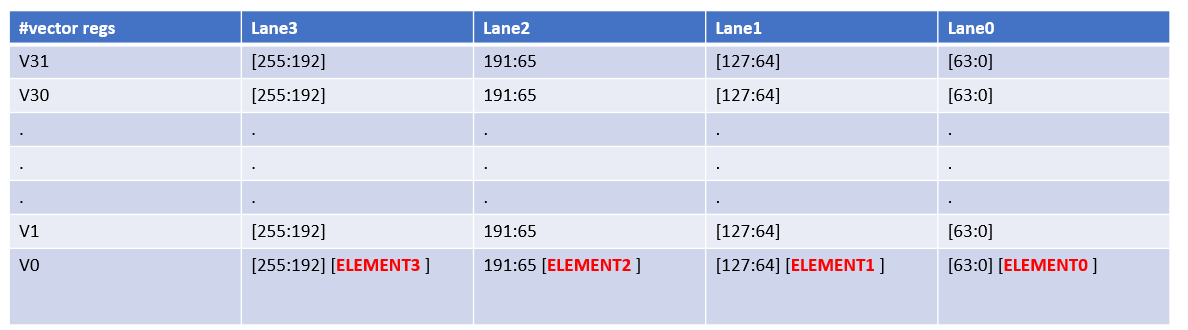
Total lanes = 4

Num banks per lane = 4

Max SEW = 64

**How is data distributed across banks ?**

Lets take an example , if the SEW is set as maximum 64, then there will be 4 element to operate on distributed in 4 different lanes. The distribution will be as follows.



Lane0 will hold element 0 i.e LSB of [63:0] for every vector registers out of 32. If we have 4 banks then we can divide these element0 into 4 equal parts :

Lane 1 : :

Bank1 : V0-V7 [63:0]

Bank2: V8- V15 [63:0]

Bank3: V16-V23 [63:0]

Bank4: V23-V31 [63:0]

Lane 2:: Bank1 : V0-V7 [127:64]

Bank2: V8- V15 [127:64]

Bank3: V16-V23 [127:64]

Bank4: V23-V31 [127:64]

Lane 3:: Bank1 : V0-V7 [191:65]

Bank2 : V8- V15 [191:65]

Bank3 : V16-V23 [191:65]

Bank4 : V23-V31 [191:65]

Lane 4:: Bank1 : V0-V7 [255:192]

Bank2 : V8- V15 [255:192]

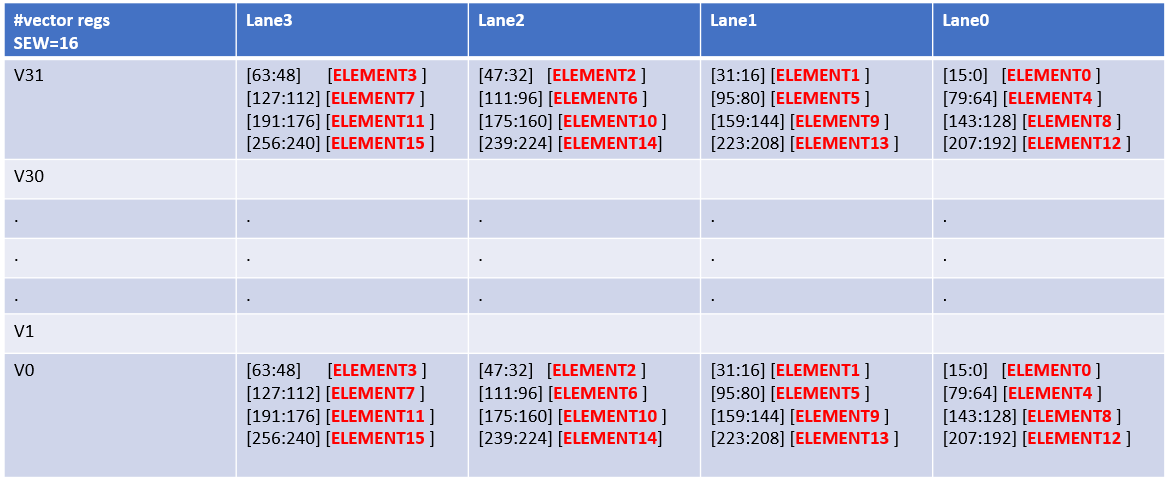
Bank3 : V16-V23 [255:192]

Bank4 : V23-V31 [255:192]

For lower SEW, we again divide the vector regs in similar way. Hence number of elements per vector operation = VLEN/SEW.

For eg , for SEW=16 , there will be 256/16 = 16 vector elements, and each lane distribution is

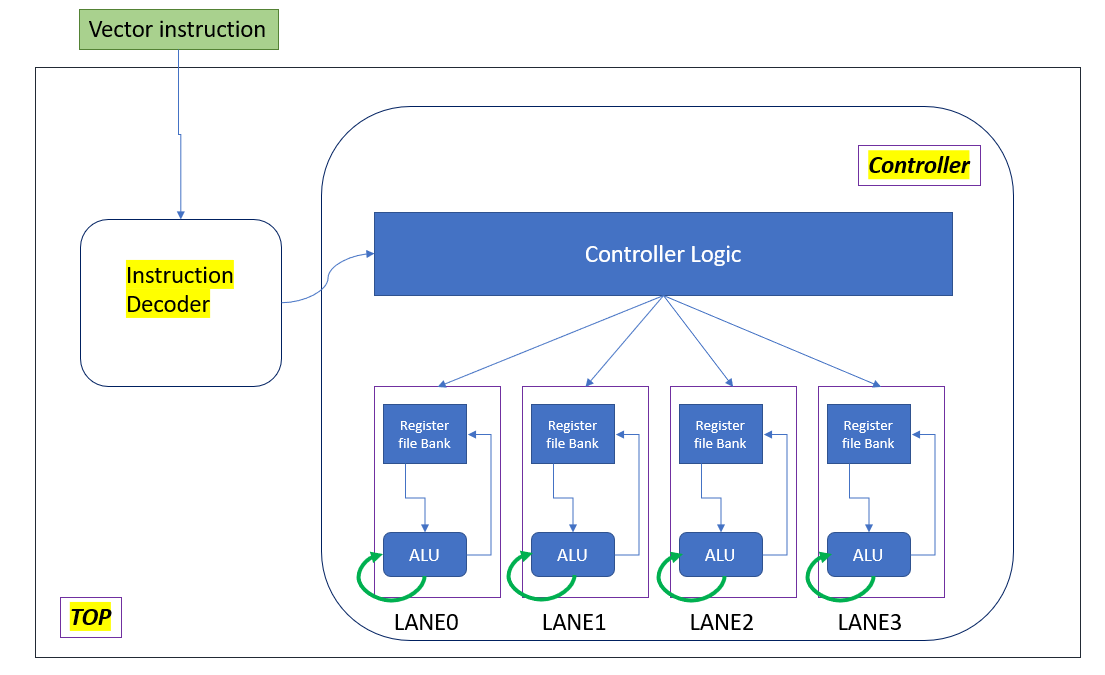
as below: with lane0 containing every 4th element starting from 0th element and same for other lanes.



The databus width of the bank is the max SEW which is 64 bits. Since there are 4 banks , it is assumed that each address of the bank holds a single element , although irrespective of the element width , i.e if the SEW= 8 , then the rest 64-8=56 bits of the bank addresses are ignored/assumed to be zero. Althought this assumption as taken from reference ARA architecture where there no part select , this assumption might need to be revisited.

The number of addresses per bank required will be determined by the min SEW i.e 8. Hence for VLEN 256 and SEW=8 , there will be 64 element per vector register (2 56/8) . Hence for 8 vector registers (each bank is mapped to 8 vector regs) : will be 64 x 8 = 256 element location of memory ,i.e address bus required is = 8 bits.

**BLOCK DIAGRAM**



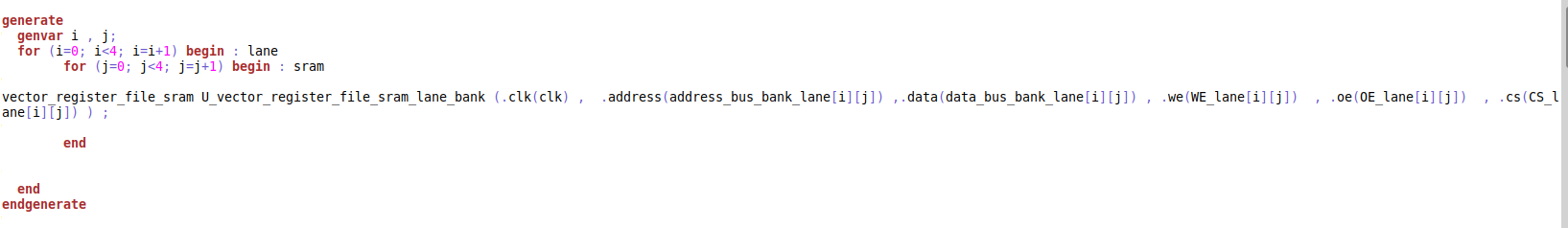
**INSTRUCTION DECODER :**

The decoder takes 32 bit vector instruction, to determine the vector operation to perform and which vector registers are involved. Currently., vector-vector operations for add , sub , multiplication and MAC operation is coded. The decoder signals – vs1 , vs2 , vd , and type of operation , whether it’s a load/store or alu operation: are output to the controller.

**CONTROLLER :**

The controller consists of :

1. Vector register files – 4 per lane coded with genvar.



1. Combinational ALU - Since this is combinational , hence output is immediately available

To the output. The SEW Is also fed to the ALU , so that the output is restricted to the SEW width and rest of the output 64-SEW bits are turned to 0. There are 3 inputs for MAC operation support. Only integer ALU operations supported now.

1. 4x1 MUX – per input of lane , to selecy which of databus of bank goes to which input of lane A,B or C



**TBD :**

1. *A re-circulation mux is missing now – it should be added for MAC operation.*
2. *Also assumption is that data is already stored in banks , we are just using it. Load/store operations needs to be updated*
3. *Bank conflict cases needs to be taken care. Right now the TB is testing vs1 , vs2 and vd for different banks.*