CEVA指令

## LSU指令

|  |  |
| --- | --- |
| **LS接口** | **含义** |
| vbcpld([nonalign],M,N) | 加载地址库(M+N)中8个**地址对**相应的内容到VRF中，标量M指向基地址，为const类型，以字节为单位；向量N为偏移量，其每个元素值取决于元素(char/short/int)类型；nonalign为可选参数，若有则偏移量N以字节为单位，与元素类型无关。NOTE: vbcpld for vector bank couple load |
| vbcpst(A,M,N，[condition]) | 存储向量A的元素到地址库(M+N)中8个**地址对**中，M指向基地址，以字节为单位；N为偏移量，取决于元素类型，condition为可选参数，其值可为nonalign/satu/sat。 |
| vldchk(N,Z0,Z1) | 加载地址库N(绝对地址)对应的内容到目的操作量中，其中奇元素写入Z0,偶元素写入Z1 |
| vldov(N,Z0,Z1,Z2,Z3) | 加载地址库N对应内容到4个相同类型元素的向量中。NOTE: ov for overlap |
| vpld([rel],M,N) | 默认为绝对地址模式，加载以M为基地址N为变址的不同元素到VRF中；rel相对地址模式，M为基地址，N为相对于各自bank下的相对元素数。NOTE: M所指类型必须与输出类型一致。仿真测试，无论绝对地址模式还是相对地址模式下，M不要求64字节对齐。 |
| vpst([rel],A,M,N,[condition]) | 存储向量A的元素到地址库(M+N)16个地址中，rel含义同上，condition:satu/sat。当M指向[u]char/[u]short类型时，N为short16类型；当M指向int/float类型时，N为int8类型。 |
| vst(A,N,condition) | 存储向量A的元素到地址库N中。 |

## SPU指令

|  |  |
| --- | --- |
| **SPU接口** | **含义** |
| bitdup(A) | 将A的每一比特位（以第i位为例）的值复制到目的标量的第2\*i位和2\*i+1位，A限制为short类型或者unsigned short类型，返回值为int类型 |
| cntbits([zeros],A) | 返回A的所有比特位中1或0（默认为1）的个数，zeros为可选项 |
| ffb([clr,set],[lsb,msb],A) | 返回A中从最高标志位(MSB)或者最低标志位(LSB)到第一个set位(1)或者cleared位(0)的距离，源操作数和目的操作数均要求是32位，源操作数为unsigned int类型 |
| ffb(sign,[sb],A) | 返回从MSB符号位到第一个不同于该位的比特位之间的距离，用于计算从源操作数中给定位置开始一连串相同比特位的个数；sb即lsb/msb。NOTE:ffb for find first bit |
| fpextract(mant,A) | 提取浮点数A的尾数(Mantissa)部分，返回值为unsigned int |
| fpextract(expn,A) | 提取浮点数A的指数（Exponent）部分 |

## VPU基础指令

|  |  |  |
| --- | --- | --- |
| 128bit位向量数据类型：char16\uchar16 short8\ushort8  256bit位向量数据类型：uchar32\char32 ushort16\short16 uint8\int8 float8  512bit位向量数据类型：uint16\int16 ushort32\short32 long8\ulong8  r: 大小关系参数{ gt(>) ge(>=) lt(<) le(=<) eq(=) neq(!=)}  同类型: 参数组合时不仅指(int,int)……也包含(int,int8,int16)……为同类型,但不包含(int,uint)  符号Z函数参数说明: 带Z符号类型参数的函数均无返回值，结果存放在带Z符号类型参数  部分函数通用参数说明：  init ：根据用户定义的值初始化寄存器或累加器  splitsrc ：源操作数的高位和低位作为两个独立的源操作数  square ：源操作数平方处理  sat(u) ：数据类型转换时，超过目的数据类型(有/无符号)范围值，则以其极值代替  rnd ：开启数据舍入，在做移位运算时数据加 1 ，例如111右移1位时，无rnd结果为011，有rnd结果为100  psl ：选择逻辑位移否则就是默认算术位移。逻辑移位不考虑符号，空处直接补0，算术移位则要保证符号位不变  absexp ：返回指数的绝对值 | | |
| vabscmp(r , A, B) | 判断向量元素绝对值的大小关系是否成立. A和B为256bit位的有符号向量. 若成立则返回值对应bit位置1, 否则置0 | 返回值Z数据类型和A类型的对应关系：  char32A:uintZ;short16A  :ushortZ; int8A:ucharZ |
| vabscmpmov(r,A,B)  |(r, A, B, Z0, Z1)  PS：| 表示其他参数组合形式。下同！ | 判断向量绝对值的大小关系是否成立. A和B为256bit位的有符号向量. (r,A,B)表示若AB关系满足r 则返回值为A否则为B; (r,A,B,Z0,Z1)表示若AB关系满足r则Z0=A;Z1=B否则Z0=B;Z1=A. | (r,A,B)参数组合函数有返回值; (r,A,B,Z0,Z1)参数组合函数无返回值。 |
| vabssub(A, B) | 计算A与B之差的绝对值. A为256bit位的标准向量类型, B为立即数或者256bit位标准向量类型, AB为同类型. | 返回值为与A同类型的256bit无符号向量。 |
| vabssubacc(A, B, W)  函数功能:  W=W+|A – B| | A为256bit位的标准向量类型, B为立即数或256bit位的标准向量类型. AB为同类型W 为512bit位(无long型)或256bit位的无符号向量类型 | 返回值Z与W相同，W和A的对应关系  参数数据类型组合：  u\ch32.A:us32.W  u\sh16.A:uint16.W  u\int8.A:uint8.W |
| vaccadd(A, B)  |(A, B, C) | 计算A+B或者A+{B C}. A为512bit位或者256bit位的标准向量类型, B为256bit位的标准向量类型或(u)short8型或立即数(无char型). C为256bit位的标准向量类型. | BC为同类型.  返回值的类型为和A相同类型的有符号向量 |
| vacccast(sw, A) | 向量A数据类型转换。A为256bit位(无int)或者512bit位的标准向量类型。sw:{satu:无符号数极值 sat:有符号数极值} | sw:A的数值若是超过目标类型范围，则以目标类型极值替换 |
| vaccshiftr(A, B)  返回值Z说明：  intA时,Z=A>>B[4:0]  shortA时,Z=A>>B[3:0]  charA时,Z=A>>B[2:0] | 右移运算。A为uchar型或者512bit位标准向量类型或者256bit位的无符号向量。B为512bit位的标准向量类型或者256bit位的无符号向量。返回值为同A或B相同数据类型的512bit位的标准向量类型 | B[x:0]表示B[i]元素二进制形式Dx~D0位作为右移次数intA: B[0]=7或39均右移7次B[4:0] {D4~D0:00111} |
| vshiftl(A, B)  说明：B[5]等价D5  B[x:0]参考上面vaccshiftr() | 和A同类型的有符号向量Z(返回值)说明:  if(!B[5]) Z=intA<<B[4:0] else Z= intA>>|B[4:0]  if(!B[4]) Z=(s)A<<B[3:0] else Z= (s)A>>|B[3:0]  if(!B[3]) Z=(c)A<<B[2:0] else Z= (c)A>>|B[2:0] | A为256bit位标准型向量，B为256bit位的标准型向量或立即数(无(u)short型、uchar型) |
| vshiftr(A, B)  PS: Z未说明时默认作为返回值符号。 | A为256bit位的标准型向量。B为256bit位的无符号向量或无符号立即数(无short型)。返回值类型为和A相同数据类型的有符号向量。 | intA时,Z=A>>B[4:0]  shortA时,Z=A>>B[3:0]  charA时,Z=A>>B[2:0] |
| vsplit(A, B, Z0, Z1) | A为short16或ushort16型向量，B为uchar型，Z0和Z1均为short16型。 | 执行结果:Z0=A>>B  Z1=A&((1<<B)-1) |
| vaccsub(A,B)  |(A,B,C)  返回值类型说明：  A同类型的512bit位的有符号向量 | 参数只有AB时,A为512bit位标准型向量或256bit位向量(无char型),B为256bit位标准型向量或(u)int型或(u)short8型向量  参数有ABC时,A为512bit位向量(无(u)long型)BC为同类型256bit位向量(无char型) | 计算A ,B之差{A[i]-B[i]} || 向量A共2N个元素，前N个和向量B相减加，后N个和向量C相减 |
| vadd(A, B) | A为256bit位的标准向量类型，B为立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同类型  计算A与B之和 |
| vaddmac(A, B, C, W)  |(A,B,C,D,E)  |(rnd,A,B,C,D,E)  |(psl,A,B,C,D,E)  |(rnd,psl,A,B,C,D,E) | AB同为256bit位(无int)或128bit位(无short)标准型向量，C为256bit位标准型向量(无int型)或u\short型立即数,DW为512bit位标准型向量(无long型) ,E为uchar型立即数。返回值为512bit位或128bit有符号向量 | 计算W=W+(A+B)\*C； 计算[D+(A+B)\*C]>>E； |
| rnd:开启舍入(默认无舍入);psl逻辑右移(默认算术右移)。 |
| vaddmpy(A, B, C)  |(rnd,A, B, C, D)  |(psl,A,B,C,D)  |(rnd,psl,A,B,C,D) | AB同为256bit位(无int)或128bit位(无short)标准型向量,C为立即数(无int型)或256bit位(无int型)标准型向量,。返回值为256bit位或512bit位的有符号向量. | D为uchar型立即数 |
| 计算(A+B)\*C；计算[(A+B)\*C]>>D; |
| vaddsat(A, B)  char.SAT\_LOW=-128  char.SAT\_HIGH=127 | A为256bit位的标准型向量(无int型),B为标准型立即数(无int型),AB同类型，返回值类型同A数据类型相同的256bit位向量 | A+B之和限制在返回值类型范围,超过赋值SAT\_HIGH,低于则相反 |
| vand(A, B) | A为256bit位的标准向量类型，B为标准立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同bit位数.  按位与运算A&B |
| vclip(A, B, C) | ABC数据类型相同,A为256bit位的标准型向量,BC均为立即数或256bit位标准型向量 | 判断max(min(A, C), B)  的结果作为返回值 |
| vcmp(r, A, B)  |(A, B, C)  |(r,vpr, A, B, C)  | (vpr, A, B, C , D) | 有参数r时判断AB是否满足r,无r时判断  B>A<=C是否成立,判断结果再与C或者D进行逻辑运算。vpr表示逻辑操作{xor,or,and},返回值均为无符号立即数。 | A为256bit位标准型向量,BC为256标准型向量或立即数,D为立即数。 |
| vcmpacc(r, A, B, C, W) | A为256bit位标准型向量，BC为立即数或256bit位的标准型向量，W为512bit位(无long型)或256bit位标准型向量(无char型) | AB关系是否满足r?  if true W=W+C |
| vcmpmov(r,A,B)  |(r, A, B, C, D) | AC为256bit位的标准型向量,BD为立即数或256bit位的标准型向量。返回值为和A同数据类型256bit位的有符号向量。 | AB关系是否满足r?  if true Z=A else Z=B  if true Z=C else Z=D  NOTE: B和D最多有一个是向量 |
| vcntbits(A)  |(zeros, A) | A为256bit位的标准型向量，返回值为A同数据类型的无符号向量 | 按位计0或1的个数(默认计1的个数) |
| vdeinterleave(A)  |(odd,A) | 把A中元素分离到返回值Z中，默认分离偶数位元素，若有参数odd,则分离奇数位 | A为256bit位的标准型向量(无int型) |
| vdeinterleave3(A, B, C, Z0, Z1, Z2) | 把ABC依次串联，新的向量中的[3i]元素分离到Z0[i]，[3i+1]到Z1[i]，[3i+2]到Z2[i]。 | ABC均为256bit位的标准型向量(无int型) |
| vhist([decrement],[A],M,N,  [condition]) | 向量元素数值频率统计，返回void类型，可通过A分别为N配置元素值个数，M为统计结果，结果存储方式较奇怪需注意，前32个数据分别为16个向量元素的“0-1”次数、16个向量元素的“2-3”次数，依次类推，总输出数据个数为256×16； | 默认为自加（increment），decrement为自减；A、N为short16型；有A时M为short\*或者int\*型，无A时M为short\*型；condition为U16类型，分别控制16个元素有效性； |
| vintramax(A) | A为short16或int8型向量 | A中最大值Z=max(A) |
| vintramin(A) | A为short16或int8型向量 | A中最小值Z=min(A) |
| vintrasum(A)  |(A,B) | A为256bit位的标准型向量, A为256bit位的标准型向量(无int型)。返回值为int8或uint8型向量 | A所有元素和sum(A) |
| vinterleave(A,B)  |(sat(u),A,B) | AB 均为256bit位标准型向量(无char型),返回值为256bit位的标准型向量(无int型) | AB返回向量Z[2i]=A[i]  Z[2i+1]=B[i]。 |
| vinterleave3(A,B,C,Z0,  Z1,Z2) | ABC向量元素按A[i]B[i]C[i]顺序排列,Z0,Z1,  Z2分别占前1/3，中间1/3，后1/3元素. | ABC为u\char32型或  u\short16型,无返回 |
| vlut(A,B,C,W) | 以向量C中第i个元素值为索引值index，当索引值index小于A的元素个数时，以A[index]作为输出向量第i个元素值；否则，以index减去A的元素个数作为索引值返回B中对应值到输出向量第i个元素值 | 返回值、W为相同的有符号256bit向量；A、B均仅支持256bit向量；C为char32型；W含义(?) |
| vinv([absexp,] A)  |([absexp,]A,B,Z0,Z1)  |([absexp,]A,Z0,Z1) | <==: uintA返回值为int型  <==: ui8A ui8B us16Z0 s16Z1  <==: us16A us16Z0 s16Z1; ui8A us8Z0 us8Z1 | 1/A=Z0 \* 2^(Z1-8):计算向量A元素或int A的倒数，得到的浮点数由Z0(表示尾数)和Z1(表示指数)组成 |
| vnand(A, B) | A为256bit位的标准向量类型，B为标准立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同bit位数.  按位与非Z=~(A&B) |
| vnor(A, B) | A为256bit位的标准向量类型，B为标准立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同bit位数.  按位或非Z=~(A | B) |
| vnot(A) | A和返回值均为256bit位的有符号向量 | 按位取非Z=~A[bit] |
| vneg(A) | A和返回值均为256bit位的有符号向量 | 取反运算Z=-A |
| vxor(A, B) | A为256bit位的标准向量类型，B为标准立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同bit位数.  按位异或Z=A ^ B |
| vor(A, B) | A为256bit位的标准向量类型，B为标准立即数或者256bit位的标准型向量。返回值为和A同类型的256bit位有符号向量 | AB为同bit位数.  按位或运算Z=A | B |
| vmac(A, B, W)  |(A, B, C, D)  |(rnd,A,B,C,D)  |(psl,A,B,C,D)  |(rnd,psl,A,B,C,D) | AB为立即数(A仅u\char)、256bit位或128bit位标准型向量,C为256bit位或512bit位的标准型向量(无long),D为uchar型立即数,W为512bit位的标准型向量,返回值类型为256bit位或者512bit位 的标准型向量 | 计算W=W+(A\*B)  计算[C+(A\*B)]>>D  参数A,B,C,D中的C,D也可以是D,W即A,B,D,W |
| vmad(A, B, C, D)  |( init,A,B,C,D,E)  |( A,B,C,D,E)  |(rnd,A,B,C,D,E)  |(psl,A,B,C,D,E)  |(rnd,psl,A,B,C,D,E) | 参数:(u)int8A, (u)shortB,(u)int8C,(u)shortD  多一种参数类型组合:  (u)char16A,(u)charB,(u)char16C,(u)charD  参数ABCD类型组合:  (u)char32A,(u)charB,(u)char32C,(u)charD  (u)char16A,(u)shortB,(u)char16C,(u)shortD  (u)short16A,(u)charB,(u)short16C,(u)charD  (u)short16A,(u)shortB,(u)short16C,(u)shortD  适用第二种到第六种参数 | 返回值int8或uint8  返回值为512bit位的标准型向量(无long型)  参数E说明:  E均为uchar型立即数  返回值说明：  返回值为256bit位的标准型向量(无int8型) |
| vmad(splitsrc,rnd,psl  A,B,C,E)  |(splitsrc,init,A,B,C,E)  |(splitsrc,rnd,A,B,C,E)  |(splitsrc,psl,A,B,C,E)  |(square,init,A,B,C,E)  |(square,rnd,A,B,C,E)  |(square,psl,A,B,C,E)  |(square,psl,rnd,A,  B,C,E) | 参数ABC类型组合：  (u)s16A,(u)s16B,(u)c32C  (u)c32A,(u)c32B,(u)c32C  (u)s16A,(u)s16B,(u)s16C  参数E为uchar型立即数。  适用左方所以参数类型。  返回值为(u)int16型或256bit位(无(u)int8型)的标准型向量 | 返回值Z说明：  Z=(A\*B+C\*D)  Z=(A\*B+C\*D)>>E  Z=(A\*Cl+B\*Ch)  Z=(A\*Cl+B\*Ch)>>E  Z=(A\*A+B\*C)  Z=(A\*A+B\*C)>>E |
| vmax(A, B)  |(A, B, C) | ABC均为256bit位的标准型向量， | 取(A, B)或(A, B, C)中的元素最大值 |
| vmin(A, B)  |(A, B, C) | ABC均为256bit位的标准型向量 | 取(A, B)或(A, B, C)中的元素最小值 |
| vmed(A, B, C) | ABC均为256bit位的标准型向量 | 取(A, B, C)元素中间值 |
| vmpy(sb,A, B)  |(A,B)  |(rnd,A, B, D)  |(psl,A, B, D)  |(rnd,psl,A, B, D)  |(splitsrc,A,B,C,D)  |(splitsrc,rnd,A,B,C,D)  |(splitsrc,psl,A,B,C,D)  |(splitsrc,rnd,psl,A,  B,C,D) | 组合:sb, (u)in8A,(u)in8B|(u)inB|(u)s8B|(u)sB  组合:(u)c32A,(u)c32B|(u)cB; (u)c16A,(u)s16B  |(u)c16B|(u)cB|(u)sB; (u)s16A,(u)s16B|(u)sB  参数组合: (u)c32A,(u)c32B|(u)cB; (u)c16A,  (u)s16B|(u)sB; (u)s16A,(u)s16B|(u)sB  适用左方对应区参数类型组合。  适用左方的参数组合  (u)c32A, c16B, c16C; (u)32A, uc16B, uc16C;  此组合返回值为char32型向量  (u)c32A, s16B, s16C; (u)32A, us16B, us16C;  此组合返回值为short32型向量 | 返回值为int8型向量  返回值类有：int8,int16,  short16,short32  返回值类型为256bit位的有符号向量  参数D说明:  D均为uchar型立即数 |
| vmpy(A,B,D,Z0,Z1)  |(rnd,A,B,D,Z0,Z1)  |(psl,A,B,D,Z0,Z1)  |(rnd,psl,A,B,D,Z0,Z1) | 适用左方所有参数类型组合：  (u)char32A,(u)shortB,ucharD  Z0,Z1均为short16型向量, 函数无返回值. | 计算A\*B  计算(A\*B)>>D  计算{Zl=(Al\*B)>>D;  Zh=(Ah\*B)>>D} |
| vmpyadd(A,B,C)  |(A,B,C,D)  |(psl,A,B,C,D) | : (u)c32A (u)c32B (u)c32C|(u)sC 或 + ucD  : (u)s16A(u)s16B(u)c16C|(u)s16C或+psl ucD  : (u)c32A (u)cB (u)sC 或 +ucD | 返回值为A相同元素个数的int16型或short32型；D为uchar型  计算A\*B+C 或(A\*B+C)>>D |
| vmpymat4x4(part,A,B)  |(part,transp,A,B)  |(part,acc,A,B,W)  |(part,acc,transp,A,B,W) | A 和B为(u)short16型向量，W为int8型向量，返回值为int8型; acc: 要求结果累加;  part: 选项有hi和lo表示向量高位和低位;  transp : 得出第一个输入向量的转置矩阵; | 计算if(transp) (transp)(A) \*B else A \*B  计算if(transp) W=W + (transp)(A)\*B  else W=W + A \* B； 返回值为A相同类型short16或char32型向量 |
| vmpynorm(A,B,C)  |(rnd,A,B,C)  |(psl,A,B,C)  |(rnd,psl,A,B,C) | :[rnd] [psl] (u)s16A (u)s16B|(u)sB (u)s16C  :[rnd] (u)c32A (u)c32B uc32C  参数组合数量应和有无开关或一开关或二开关相关 | 返回值的类型为A相同类型的有符号向量  计算((A\*B)+rnd)>>C  rnd:二进制lsb位加1, 对于11或111或有意义,而10或110或1010等右移可能无影响 |
| vsqrt([absexp,] A)  |([absexp,]A,B,Z0,Z1)  |([absexp,]A,Z0,Z1) | <==: uintA 返回值为int型  <==: ui8A ui8B us16Z0 s16Z1  <==: us16A us16Z0 s16Z1; ui8A us8Z0 us8Z1 | sqrt(A)=Z0 \* 2^(Z1-8):计算向量A元素或int A的开根号值，得到的浮点数由Z0(表示尾数)  和Z1(表示指数){-8<=Z1<=8}组成 |
| vsprti([absexp,] A)  |([absexp,]A,B,Z0,Z1)  |([absexp,]A,Z0,Z1) | <==: uintA返回值为int型  <==: ui8A ui8B us16Z0 s16Z1  <==: us16A us16Z0 s16Z1; ui8A us8Z0 us8Z1 | sqrti(A)=Z0\*2^(Z1-16):计算向量A元素或intA开根号值的倒数，得到的浮点数由Z0(表示尾数)和Z1(表示指数)组成 |
| vsubsquare(A, B)  函数功能：计算(A-B)^2的值 | A为256bit位的标准型向量(无int型),B(无int型)为立即数或256bit位的标准型向量。返回值为512bit位的无符号向量 | 返回值Z类型组合:  u\char32.A:ushort32.Z  u\short32.A:uint16.Z |
| vsubsat(A, B)  例:uchar.SAT\_LOW=0  uchar.SAT\_HIGH=255 | A为256bit位的标准型向量(无int型),B为标准型立即数(无int型),AB同类型，返回值类型同A数据类型相同的256bit位向量 | A+B之和限制在返回值类型范围,低于赋值SAT\_LOW,超过则相反 |
| vusint2fp(A) | 将A转换为浮点型 | 返回值为float8型；A为uint8型 |
| vffb(clr\_set, sb\_sw, A)  |(sign\_sw, A) | A从lsb最低位|msb最高位开始 按位计数到出现clr(0)或set(1) ，返回间隔的位数  ||从符号位开始计数到第一个和符号位值不同的位，返回间隔的位数 | A为256bit位的标准型向量，返回值为和A相同的256bit位的无符号向量. |
| vfpcombine(A, B, C)  PS:Z未定义作为返回值符号 | if(B=-126) Z=A\*B(C<<31|(A&0x7FFFFF);  else Z=((C<<31)|((B+23+127)<<23)|(A&0x7FFFFF); | A位uint8\ushort8型向量,B 为int8\short8型向量,C为uchar型 |
| vfpextract(A, Z0, Z1) | void类型，分别提取浮点向量A元素的尾数(Mantissa)部分和指数（Exponent）部分到Z0和Z1 | A为float 8型向量，Z0位uint8型向量，Z1为int8型向量 |
| vperm(A, B, C) | AB为float8型向量或256bit位标准型向量，C为256bit位的uchar32型向量。返回值为和A同数据类型的有符号向量。 | 若C[i]值小于向量A元素数N则Z=A[C[i]]否则Z=B[C[i]-N] |
| vcast(sat,A)  |(A,B) | A为char16,(u)int8或(u)short16型；  B为(u)int8或(u)short16型； | 类型转换Z=(typeZ)A  |Z=(typeZ){A,B} |
| 下面vf开头的函数均只用于处理浮点型数据{float8 float} | | |
| vfpadd(A, B) | 计算A和B之和 | vA vB|rB |
| vfpcmp(r, A, B, C) | 判断A和B的关系，结果和C相与{[Z=(A?B)]&C} | vA vB rC[uchar] |
| vfpinv(A) | 计算A的倒数【Z=1/A】 | vA vB|rB |
| vfpmpy(A, B) | 计算A与B之积【Z=A\*B】 | vA vB|rB |
| vfpadd(A, B) | 计算浮点型A和B之和 | vA vB|rB |
| vfpsqrt(A) | 计算A的平方根【Z=SQRT(A)】 | vA|rA |
| vfpsqrti(A) | 计算A平方根倒数【Z=1/[SQRT(A)]】  或者Z[i]=1/[SQRT(A[i])] | vA  rA |
| vfpsub(A) | 计算A与B之差【Z=A-B】 | vA|vB |

## VPU滑窗（SW）指令

|  |  |  |  |
| --- | --- | --- | --- |
| **指令名** | **调用方式** | **指令说明** | **指令控制字** |
| **vspmac** | init模式  int(16|8) vspmac(init,(u)(c32|s16)A, (u)(c32|s16)B,(u)(s16)C, (U32)) | A、B为32bit交错的输入源，可选择A、B的4个元素与系数向量C的4个元素做乘加， A在前B在后，D[31:24]来控制4个A元素、4个B元素的选择，与C的4个元素做乘加，再与初始值1 << (D[5:0]-1)相加. | D[5:0]init模式初始值，右移模式右移值  D[7:6]无用  D[13:8]源偏移值  D[15:24]无用  D[20:16]系数向量偏移值  D[23:21]无用  D[31:24]控制向量A、B的元素挑选，不能选多于4个 |
| Accumulate模式  (u)int(16|8) vspmac(accumulate,(u)(c32|s16)A,(u)(c32|s16)B,(u)(s16)C,(U32)D,(u)i(16|8)W) | A、B为32bit交错的输入源，可选择A、B的4个元素与系数向量C的4个元素做乘加， A在前B在后，D[31:24]来控制4个A元素、4个B元素的选择，与C的4个元素做乘加，再与W相加 |
| psl移位模式  s(16|8) vspmac([psl],(u)(c32|s16)A,(u)(c32|s16)B,(u)(s16)C, (U32)D, (u)i(16|8)E) | A、B为32bit交错的输入源，可选择A、B的4个元素与系数向量C的4个元素做乘加， A在前B在后，D[31:24]来控制4个A元素、4个B元素的选择，与C的4个元素做乘加，再右移D[5:0]位 |
| **vsspmac** | Accumulate模式  (u)int(16|8) vsspmac(accumulate,(u)(s16)A,(u)(s16)B,(u)(c32|s16)C,(U32)D,(U8)E,(u)(i16|i8)W) |  | D[5:0]右移模式下右移值，不能大于31  D[7:6]滤波器个数(1,2,4,***~~8~~***)  D[13:8]源起始偏移，不能大于3  D[15:14]无用  D[20:16]系数向量起始偏移，不能大于1  D[31:24]控制向量A、B的元素挑选，不能选多于4个 |
| **vmswmac3** | Acc累加模式  (u)(s32|i16)vmswmac3(accumulate, (u)(c32|s16) A, (u)(c32|s16) B,  (u)(c32|s16) C, (U32)D,(u)(s32|i16) W) | A、B为32bit交叠的输入源，A在前B在后，取两个元素与系数向量乘加，再与W相加。系数向量偏移时偏移D[20:16]×16位 | D[5:0]右移模式下右移值，Z为char类型不能大于8，short类型不能大于31  D[7:6]无用  D[13:8]源起始偏移，不能大于3  D[15:14]无用  D[20:16]系数向量起始偏移，不能大于1  D[31:21]无用 |
| psl移位模式  (c32|s16)vmswmac3([psl], (u)(c32|s16) A, (u)(c32|s16) B,  (u)(c32|s16) C, (U32)D,(u)(s32|i16) E) | A、B为32bit交叠的输入源，A在前B在后，取两个元素与系数向量乘加，再与E相加，最用D[5:0]右移。系数向量偏移时偏移D[20:16]×16位 |  |
| **vswmac5** | 移位模式  short(16|8) vswmac5(psl,A,B,C,D,W) | 向量A的4个数值与向量C的4个数值乘加，与W求和之后，右移D[5:0]位，再滑窗计算向量A的剩余元素，B接在A后方防止越界。  1.返回类型为s16时，A、B为相同的(u)c32|(u)s16向量，C为(u)c32|(u)s16向量，但A、B、C不能同时为(u)s16向量，D为U32型参数设置字，W为(u)i16型累加偏置  2.返回类型为s8时，A、B为相同的(u)s16向量，C为(u)s16向量，D为U32型参数设置字，W为(u)i16型累加偏置 | D[5:0]psh移位模式下移位次数  D[7:6]无用  D[13:8]源起始偏移  D[15:14]无用  D[20:16]系数向量起始偏移  D[31:21]无用 |
| 累加模式  int(16|8) vswmac5(accumulate,A,B,C,D,W) | 与移位模式相比，没有移位运算，且返回类型不同。  1.返回类型为i16时，A、B为相同的(u)c32|(u)s16向量，C为(u)c32|(u)s16向量，但A、B、C不能同时为(u)s16向量，D为参数设置字U32型，W为累加偏置(u)i16型  2.返回类型为i8时，A、B为相同的(u)s16向量，C为(u)s16向量，D为参数设置字U32型，W为累加偏置(u)i16型 |
| long8 vswmac5 | 指令详见文档 | 指令详见文档 |
| **vswsub** | char型，返回V  V vswsub(sat,((u)c32)A,((u)c32)B,((u)c32)C,(U32)D,(uc32)E,(uc32)F) | 向量A滑窗减系数向量C，B接在A后方防止越界，作用不大，D为参数设置字，结果存储至E和F中，E为移0位和移1位结果， F为移2位和移3位结果  备注：源有起始偏移时A最左数不要，A依次减C，补充B的最左数到A的最右，再次减C，存储至E中。类似地得到F。  **char类型只计算前16个数据** | D[7:0]  无用  D[12:8] 源起始偏移  D[23:14] 无用  D[27:24] 溢出保护值.  D[31:28] 无用  (1<<D[27:24])-1 |
| short型，返回s32  vswsub(sat,((u)s16)A,((u)s16)B,((u)s16)C,(U32)D) | 向量A滑窗减系数向量C，B接在A后方防止越界，作用不大，D为参数设置字，返回512位的s32  备注：滑窗方式为，A依次减C，然后A最左数不要，补充B的最左数到A的最右，再次减C，成为s32。 |
| **vswsad** | int16 累加模式  (u)i16 vswsad(accumulate,((u)c32|s16)A,((u)c32|s16)B,((u)c32|s16)C,(ui)D,((u)i32)W) | 累加向量A减系数向量C的绝对值，B接在A后，D为参数设置字，返回512位的i16结果  备注：滑窗方式为，A依次减去C[0]，取绝对值，再加累加偏置W，最后为i16，B作为补充接到A的后面。  **计算方法：**  **1.绝对差计算次数为0时，计算方法为W+abs(A[i]-C[0])，绝对差计算次数为1时，计算方法为W+abs(A[i]-C[0])+abs(A[i+1]-C[1])，依次类推**  **2.char类型只计算前16个数据** | D[5:0]psh移位模式下移位次数  D[7:6]绝对差计算次数  **源为s16时D[6]有效**  **源为c32时D[7:6]有效**  D[13:8] 源起始偏移  D[15:14]无用  D[20:16]系数向量起始偏移  D[31:21]无用 |
| short16 移位模式  s16 vswsad((psl),((u)c32|s16)A,((u)c32|s16)B,((u)c32|s16)C,(ui)D,((u)i32)E) | 累加移位向量A减系数向量C的绝对值，B接在A后，D为参数设置字，返回256bit的s16结果  备注：滑窗方式为，A依次减去C[0]，取绝对值，再加权重E，再移位D[5:0]，最后为s16，B作为补充接到A的后面。  **计算方法：**  **1.绝对差计算次数为0时，计算方法为(E+abs(A[i]-C[0]))>>**D[5:0]**，绝对差计算次数为1时，计算方法为(E+abs(A[i]-C[0])+abs(A[i+1]-C[1]))>>**D[5:0]**，依次类推**  **2.char类型只计算前16个数据** |
| **vmswmad** | s32 vmswmad(((u)c32)A,((u)c32)B,((u)c32)C,(ui)D)  i16 vmswmad((u)(c32|s16)A,(u)(c32|s16)B,(u)(c32|s16)C,(ui)D) | 向量乘积和，A、B为32bit交叠，A位于前，B位于后。A中取两个元素与C中两个元素乘积再求和，A超出时，B为候补，如D[5:0]有值，则作为初始值相加。使用参数偏移时，偏移值为D[20:16]×16，D[20:16]取值只能为0或1.  A、B为相同类型，返回s32时，A、B、C只能为(u)c32向量类型，C可与A、B不同。返回i16时，A、B可以为(u)(c32|s16)，C可以为(u)(c32|s16)，但不能同时为(u)c32 | D[5:0]初始值，不能大于31  D[7:6]无用  D[13:8]源偏移，不能大于3  D[15:14]无用  D[20:16]参数偏移，不能大于1  D[31:21]无用 |
| **vswmad** | s32|i16 vswmad(((u)c32)A,((u)c32)B,((u)c32)C,(ui)D) | 向量乘积和。向量元素A[i]乘以C[0]与A[i+1]乘以C[1]之和，最后再加上2∧(D[5:0]-1)，B接在A后，D为参数控制字，返回s32|i16结果。  A、B为相同的(u)c32向量且C为(u)c32向量时，需返回s32  非上述条件下，返回i16 | D[5:0]加偏指数  D[7:6]无用  D[13:8] 源起始偏移  D[15:14]无用  D[20:16]系数向量偏移  D[31:21]无用 |
| **vswmpy5** | i8|i16 vswmpy((u)(s16|c32)A, (u)(s16|c32)B, (u)(s16|c32)C,U32 D) | 4元素向量乘积和。  A[i]xC[0] + A[i+1]xC[1] + A[i+2]xC[2] + A[i+3]xC[3] + 2∧(D[7:0]-1)，B接在A后，D为参数控制字  A、B为相同的(u)s16向量，C为(u)s16向量时，需返回i8  非上述条件下，返回i16 | D[7:0]rnd模式下加偏指数  D[13:8]源起始偏移D[15:14]无用  D[20:16]系数向量偏移  D[31:21]无用 |
| **vsswmac5** | coming soon |  |  |