ARMv8-A Scalable Vector Extension for Post-K
Post-K Supports New SIMD Extension

- The SIMD extension is a 512-bit wide implementation of SVE.
- SVE is an HPC-focused SIMD instruction extension in AArch64.
  - Co-developed with ARM, taking advantage of Fujitsu’s HPC technologies.
  - SVE and Advanced SIMD (NEON) are available, concurrently.
- FUJITSU’s microarchitecture and compiler technologies maximize the execution performance of the Post-K CPU with SVE.

Post-K CPU core

- ARMv8-A with SVE
- FUJITSU’s microarchitecture

FUJITSU’s compiler

= Good performance with practical applications
<table>
<thead>
<tr>
<th>Features</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-lane predication</td>
<td>High vectorization rate</td>
</tr>
<tr>
<td>Fault-tolerant speculative vectorization</td>
<td>Wider SIMD (512-bit wide for Post-K)</td>
</tr>
<tr>
<td>Gather-load and scatter-store</td>
<td>Efficient utilization of vector</td>
</tr>
<tr>
<td>Horizontal and serialized vector operations</td>
<td>e.g. Gather/Scatter for packed 32-bit FP,</td>
</tr>
<tr>
<td></td>
<td>Packed SIMD for 1-, 2-, 4- and 8-byte integer</td>
</tr>
<tr>
<td>HPC-focused instructions</td>
<td>Highly optimized executables</td>
</tr>
<tr>
<td>e.g. Reciprocal inst., Math. acceleration inst., etc.</td>
<td></td>
</tr>
<tr>
<td>Scalable vector length</td>
<td>Binary portability between different vector-length CPUs</td>
</tr>
</tbody>
</table>
Additional Vectorization Targets in Post-K

- Post-K targets new kinds of loops
- Predicate register and fault-tolerant speculative load enable additional vectorization
- Vectorization of non-"SVE-specific" loops is improved, as well

Vectorization targets as before

// Loop including if-statements
for (int i=0; i<n; ++i) {
  if (mask[i] !=0) { a[i] = b[i]; }
}

// Short-iteration Loop
for (int i=0; i<VL/2; ++i) {
  a[i] = b[i] * c[i];
}

and loops vectorized in K computer and FX100

Additional vectorization targets

// While Loop
do {
  b[i] = a[i];
} while(a[i++] != 0);

// Data-dependent loop
for (int i=0;i<n;++i) {
  a[b[i]] = a[i];
}

etc.
Vectorization of While-loop

- While-loop is vectorized with fault-tolerant speculative instructions
- Reduces the # of iterations by using SIMD instructions and improves the execution performance of the while-loop

```c
int a[n], b[n];
...
do {
    b[i] = a[i];
} while(a[i++] != 0);
```

**Target loop**

```c
.setfr
.ptrue p7.s, all
.loop:
    ldff1w z0.s, p7/z, [x0, x2, LSL #2]
    rdffr p0.b, p7/z
    cmpeq p1.s, p0/z, z0.s, #0
    brkb p1.b, p0/z, p1.b
    st1w z0.s, p1/z, [x1, x2, LSL #2]
    incp x2, p1.s

b.last .loop
```

**Vector code w/ SVE**

```c
Scalar code w/o SIMD
```

### Graph

**512-bit SIMD effect for the while-loop**

- Reduced to 1/8
- 4 instructions X iterations
- 8 instructions X/16 iterations

---

Copyright 2016 FUJITSU LIMITED
Accelerates indirect array accesses

- Supports gather/scatter for 32- and 64-bit FP, and 1-, 2-, 4- and 8-byte integers
- Post-K compiler optimizes and vectorizes indirect accesses, making use of the 512-bit vector registers to the greatest extent possible

Sample code of array access

```c
int index[n]
float P[n], Q[n];
for (i=0; i<n; ++i)
{
    P[i] = Q[index[i]];
}
```

Gathered 16 elements of 32bit-FP with 1 instruction

```c
REG. DEST.
memory Q  [15][14][13] · · · · · · · · · [3] [2] [1] [0]
REG. INDEX
14 1 · · · · · · 13 · · · · 0 3 15 2
```
Supports SIMD instructions for 1-, 2-, 4- and 8-byte Integers

Vectorizes integer operations by utilizing vector registers to the greatest extent as possible

The max number of elements for SIMD

<table>
<thead>
<tr>
<th>Element Type</th>
<th>HPC-ACE2 (256-bit)</th>
<th>SVE (as 512-bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT 8-byte</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4-byte</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>2-byte</td>
<td>x</td>
<td>32</td>
</tr>
<tr>
<td>1-byte</td>
<td>x</td>
<td>64</td>
</tr>
<tr>
<td>FP 8-byte</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4-byte</td>
<td>8*</td>
<td>16</td>
</tr>
</tbody>
</table>

* Only some fundamental operations are available

char A[n], B[n], C[n];
for(int i=0;i<64;++i) {
    A[i] = B[i] + C[i];
}

ld1b z1.b, p0/z, [x1]     // "B"
ld1b z2.b, p0/z, [x2]     // "C"
add  z1.b, p0/m, z2.b, z0.b
st1b z1.b, p0,  [x0]     // "A"

512-bit wide vector register utilization

64-way SIMD

8-byte


4-byte


2-byte


1-byte

4-Operand Multiply and Add with MOVPRFX

- SVE supports 3-operand Multiply and Add instructions of 32- and 64-bit FP and 8-, 16-, 32- and 64-bit integers as SIMD

- FMLA is floating-point Multiply and Add with predication - overwrites the addend Z0

\[
\text{FMLA } Z0.D, \text{ P0/M, Z1.D, Z2.D } \quad // \quad Z0 \gets Z0+Z1\times Z2 \text{ with P0}
\]

- 4-operand Multiply and Add in Post-K CPU by prefixed MOVPRFX

- The prefixed MOVPRFX, copying the source operand Z0 to Z3, does not spend any out-of-order resources by combining the following instruction

- All source operands Z0, Z1 and Z2 are preserved, while the destination operand Z3 is updated

\[
\begin{align*}
\text{MOVPRFX } Z3.D, \text{ P0/M, Z0.D } & \quad // \quad Z3 \gets Z0 \\
\text{FMLA } Z3.D, \text{ P0/M, Z1.D, Z2.D } & \quad // \quad Z3 \gets Z3+Z1\times Z2 \text{ with P0}
\end{align*}
\]
Advantages of the Post-K Compiler

- Maximizes the execution performance of HPC applications
  - Covers a wide range of applications, including integer calculations
- Targets 512-bit wide vectorization as well as vector-length-agnostic
  - Fixed-vector-length facilitates optimizations such as constant folding
- Inherits options/features of K computer, PRIMEHPC FX10 and FX100
- Language Standard Support
  - Fully supported: Fortran 2008, C11, C++14, OpenMP 4.5
  - Partially supported: Fortran 2015, C++1z, OpenMP 5.0
- Supports ARM C Language Extensions (ACLE) for SVE
  - ACLE allows programmers to use SVE instructions as C intrinsic functions

// C intrinsics in ACLE for SVE
svfloat64_t z0 = svld1_f64(p0, &x[i]);
svfloat64_t z1 = svld1_f64(p0, &y[i]);
svfloat64_t z2 = svadd_f64_x(p0, z0, z1);
svst1_f64(p0, &z[i], z2);

// SVE assembler
ld1d  z1.d, p0/z, [x19, x3, lsl #3]
ld1d  z0.d, p0/z, [x20, x3, lsl #3]
fadd  z1.d, p0/m, z1.d, z0.d
st1d  z1.d, p0, [x21, x3, lsl #3]
Auto-vectorization Ability of FUJITSU Compiler for Post-K

- The # of instructions of NAS Parallel Benchmark measured on FUJITSU’s simulator
- The Post-K compiler is now comparable with the proven FX100, and is improving

### Vectorization rate up for TSVC* (Fortran and C)

<table>
<thead>
<tr>
<th>TSV</th>
<th>PRIMEHPC FX100</th>
<th>Post-K Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran (135)</td>
<td>96</td>
<td>111 (FX100+15)</td>
</tr>
<tr>
<td>C(151)</td>
<td>106</td>
<td>121 (FX100+15)</td>
</tr>
</tbody>
</table>

*s482 is one of the vectorized loops thanks to SVE for (int i = 0; i < LEN; i++) {
\[\text{a}[i] += \text{b}[i] \times \text{c}[i];\]
if (c[i] > b[i]) break;
}

Copyright 2016 FUJITSU LIMITED
Post-K Improves upon Features of K computer and FX100

- FUJITSU’s microarchitecture and compiler technologies maximize the execution performance of the Post-K CPU with SVE

<table>
<thead>
<tr>
<th>ISA Features of FX100</th>
<th>Post-K</th>
<th>SVE Instructions and Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation mask (predication)</td>
<td>✓ Enhanced</td>
<td>Almost all of the instructions are predicated</td>
</tr>
<tr>
<td>Fault-tolerant instructions</td>
<td>✓ Enhanced</td>
<td>First-fault load and non-fault load</td>
</tr>
<tr>
<td>Gather-load/Scatter-store (indirect load/store)</td>
<td>✓ Enhanced</td>
<td>Gather/Scatter for floating-point and integer</td>
</tr>
<tr>
<td>Horizontal and serialized vector operations</td>
<td>✓ Enhanced</td>
<td>ADDV, EORV, PNEXT, etc.</td>
</tr>
<tr>
<td>Floating-point FMA and Integer FMA</td>
<td>✓ Enhanced</td>
<td>FMAD, FMLA, MAD, MLA, etc.</td>
</tr>
<tr>
<td>Packed SIMD</td>
<td>✓ Enhanced</td>
<td>Packed/Unpacked SIMD for integer and floating-point are available</td>
</tr>
<tr>
<td>Element permutation and shuffles</td>
<td>✓ Enhanced</td>
<td>TBL, EXT, SPLICE, COMPACT, etc.</td>
</tr>
<tr>
<td>Reciprocal approx., trigonometric/exponential func.</td>
<td>✓</td>
<td>Inherits HPC-ACE2 as FRECPX, FTMD, FEXPA, etc.</td>
</tr>
<tr>
<td>Stride-load/store</td>
<td>✓</td>
<td>Combination of structure load/store, gather/scatter and element permutations</td>
</tr>
<tr>
<td>Sector cache</td>
<td>✓ Enhanced</td>
<td>FUJITSU’s extension using ARMv8-A tagged addressing</td>
</tr>
</tbody>
</table>
shaping tomorrow with you