\Orchestrating a brighter world

NEC

NEC SX-Aurora as a Scalable Vector Playground

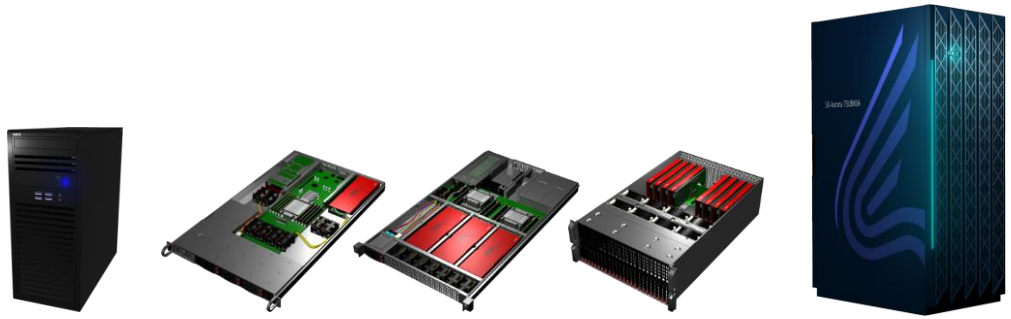
Kazuhisa Ishizaka, Yoshiyuki Ohno, Yuta Ideguchi, Erich Focht, Simon Moll

simon.moll@emea.nec.com

NEC Corporation

SX-Aurora TSUBASA Vector Computer

High performance computer ranged from deskside to cloud and supercomputer



Vector Processor on a Vector Engine PCIe accelerator card

Vector Engine (VE)

(PCIe accelerator card)

Vector Processor with 6 HBM2s

Theoretical Performance

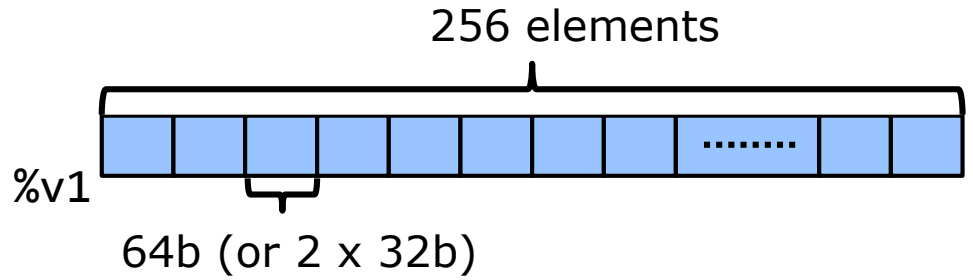
Computation:
FP32: 4.30 TFlops
FP64: 2.15 TFlops

Memory:
Capacity: 48GB
Bandwidth: 1.2 TB/s

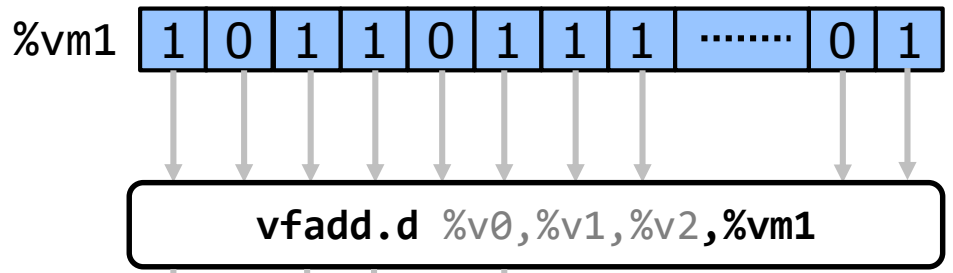
Vector Engine(VE) Scalable Vector ISA

Basics

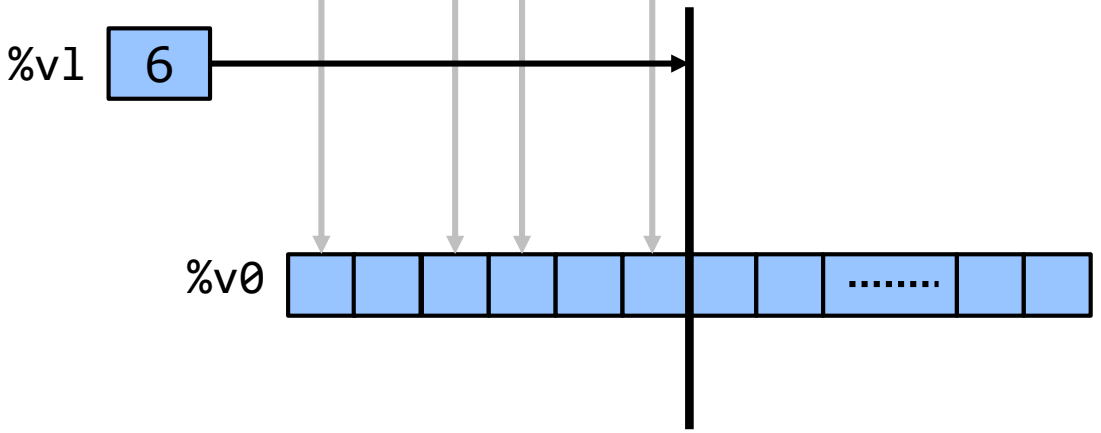
- **Wide vector registers (256 x 64bit)**
  - 64 Vector registers (%v***i***)



- **Full vector predication**
  - 16 Vector mask registers (%vm***i***)
  - ***Explicit*** operand



- **(Active) Vector Length Register**
  - One, global, VL Register
  - ***Implicit*** dependence



Vector Engine Challenges in LLVM

1) Implicit dependency through VL register

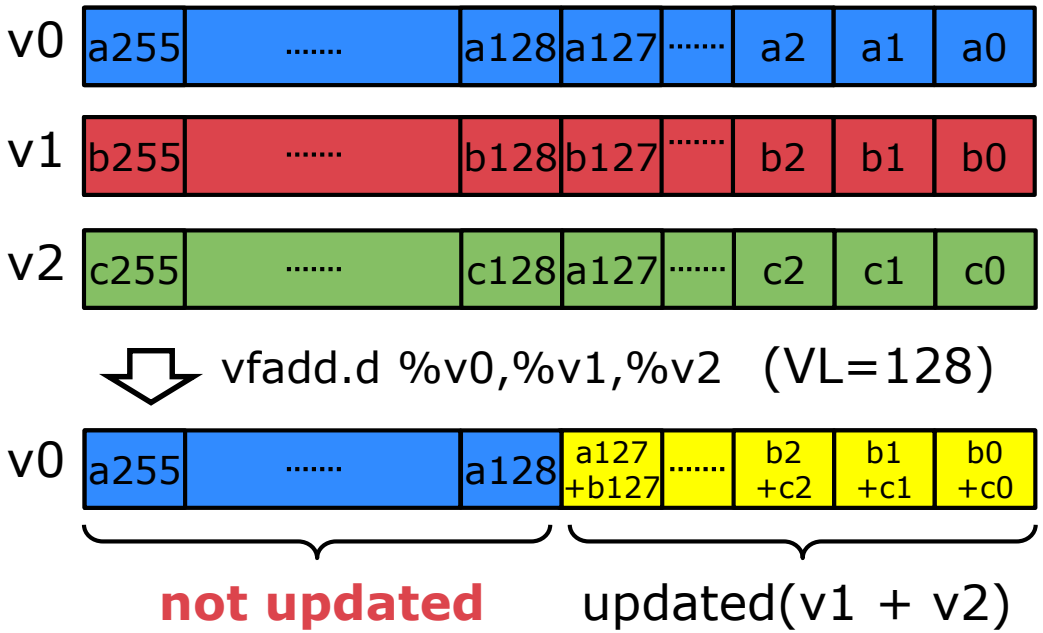
```
lvl %s37
vld %v3,8,%s0
vld %v4,8,%s1
vfmad.d %v3,%v3,%s2,%v4
```

define VL

use VL

How do we implement implicit def-use?

2) Partial update of a destination vector register



How do we introduce partial update?

Our Solution: IR and Backend

1. Vector IR based on LLVM-VP

- VL as parameter

```
$v3 = vfadd.d $v4,$v5,$pt,$vl
# for i=0,256
#     v3[i] = i < vl ? v4[i] + v5[i] : pt[i]
```

+

2. Automatic LVL generation in backend

- Inserting LVL instruction from $vl argument in IR
- Minimizing LVL instruction by current VL inference

Status and Roadmap

Status

- LLVM-VE is available at https://github.com/sx-aurora-dev/llvm
- Scalar code backend + vector intrinsics
- Application: TensorFlow for SX-Aurora

Roadmap

- Upstreaming! https://reviews.llvm.org/D69103
- Vector predication with LLVM-VP (D57504)

Welcome collaborators

- *for automatic vectorization, etc.*

Come see us at the poster session!

"NEC SX-Aurora as a Scalable Vector Playground"