



Poenaru, A. (2017, Nov 16). Comparison of Vector Architectures. Unpublished.

Other version

[Link to publication record in Explore Bristol Research](#)  
PDF-document

## University of Bristol - Explore Bristol Research

### General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:  
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/>

# SVE Users' Meeting

Andrei Poenaru

University of Bristol, UK

# Comparison of Vector Architectures

Master's Project

## Performance analysis

- Static code analysis
  - NEON vs SVE vs AVX2
- TSVC – 155 loops implementing different code patterns
  - Static code analysis and run time
- Benchmarks:
  - STREAM
  - MegaStream – UoB-developed, inspired by SNAP
- Mini-apps:
  - BUDE – compute-bound
  - MiniFMM – task-based parallelism, compute-bound

## Insight on generated code

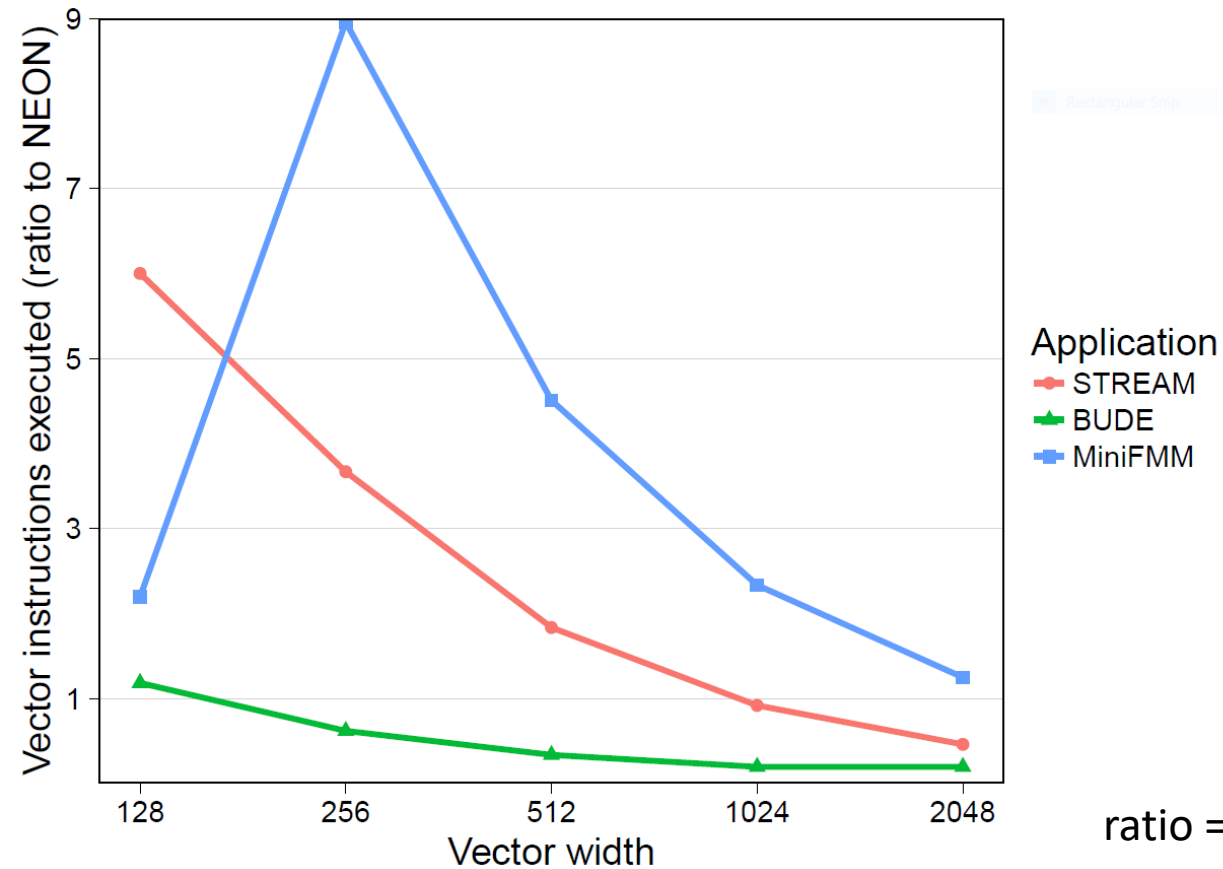
- Using a combination of `cachegrind`, `objdump` and `python`, we can see how generated code varies between platforms and compilers
  - Instruction counts
  - Types of instructions
  - Memory access patterns
- Varying the target platform and application may highlight patterns
  - How do ratios of types of instructions compare? What do they say about application performance?
  - Are there any cases that cause trouble, likely resulting in slow performance?
- Results would guide early Isambard experiments

## T SVC on AVX2 (Intel 16) vs NEON and SVE (armclang 1.1)

Vectorised on	Count	Weaknesses
AVX2, NEON, SVE	37	—
AVX2, NEON, SVE	35	Control flow, gathers/scatters, 2d arrays
AVX2, NEON, SVE	2	Product reductions
AVX2, NEON, SVE	39	Complex reductions, scalar/array expansions, loop peeling, switch statements, outer loops
AVX2, NEON, SVE	42	Variable loop bounds, expansions, complex reductions
<b>Total</b>	<b>155</b>	

Key: **green** – vectorised, **red** – not vectorised

# SVE instruction count scaling with vector width



$$\text{ratio} = \frac{\text{SVE instruction count}}{\text{NEON instruction count}}$$

## Challenges

- ARMIE 1.0 only outputs statistics about SVE instructions executed
    - Can see how changing the vector width affects instructions executed...
    - ... but not the bigger picture
  - No SVE support in third-party simulators
  - Early version of the Arm HPC Compiler meant SVE likely did not show its full potential in the TSVC collection
-



# Current Work

## Taking Advantage of SVE in LLVM

- Goal: Investigate and implement ways to
  - vectorise code that is hard to do without SVE
  - reduce the cost of vectorising certain parts of code vs plain ARMv8
- Current status
  - Early stages, evaluating LLVM opportunities
  - Vectorisation areas investigated: speculative vectorisation, pointer-chasing loops, SLP
- Evaluation:
  - Benchmarks: TSVC-like cases, mini-apps
  - Simulator with SVE support, e.g. ARMIE 1.3+, Gem5

## SVE Performance Evaluation

- Plan to re-evaluate ARMIE as a tool for benchmarking SVE
  - What can we learn using the data provided by the newer versions?
  - What extra functionality would be welcome?
- Investigate how we can analyse the performance of SVE in parallel applications, e.g. using OpenMP
  - Arm Code Advisor
  - Third-party simulator support?

---

# GROMACS

- GROMACS is a software suite for molecular dynamics simulations with a focus on performance
  - Widely used both as a benchmark and in production (2<sup>nd</sup> most used code on the UK ARCHER system)
  - Kernels written in x86 vector intrinsics massively boost performance
- Planned work:
  - Add SVE support to kernels
  - Investigate how performance varies with different compilers and how this relates to their support of intrinsics

# Questions