linaro™ Arm Solutions at Lightspeed

Mind the Gap

LLVM Toolchain for Windows on Arm

Omair Javaid

Windows on Arm + LLVM

A Success Story in Progress

- Native LLVM toolchain on Windows on Arm is **here and working**
- Major components already **build and run natively**
- Code generation quality and runtime performance are **strong**
- We are now focused on **closing the final capability gaps**
- Goal: **Parity with Linux and Windows x64 developer experience**

ARM

LLVM Delivers to Windows on Arm

Native, Production-Ready Toolchain

- **Clang** + **LLD** produce native Arm64 PE/COFF binaries
- **clang-cl** provides MSVC command-line + **ABI** compatibility
- **Flang** provides first native Fortran compiler on Windows on Arm
- **LLDB** supports native debugging workflows
- **OpenMP** enables parallel scientific + HPC workloads
- **llvm-mingw** provides GNU toolchain support using LLVM

LLVM Pushed Windows on Arm Forward

Backend Reuse — Rapid Bring-Up

- **AArch64 backend** — shared across Windows, Linux, MacOS
- **COFF, SEH, CodeView support** — shared across x64 and Arm Windows
- One backend to optimize — performance benefits arrive everywhere
- Native performance delivered early — MSVC catching up

Multi-Language Enablement — Bigger Ecosystem

- Same backend serves C/C++, Rust, Swift, Fortran, R
- No new backend investments for each language

LLVM in Action!!!

Enabling real native applications on Windows on Arm

- **Native Arm64 Windows apps** like Python, Blender, LibreOffice
- **Browsers** powered by Clang — Chrome, Edge & Firefox
- **Languages** powered by LLVM — C/C++, Fortran, Rust, R/Swift/Julia progressing
- **WorksOnWoA.com** tracks the growing native WoA app ecosystem





Native Performance Today

LLVM performance is a strength, not a gap

- Native Arm64 builds outperform x86/x64 emulation by a large margin
- Performance: **clang-cl > MSVC in SPEC 2017** and real workloads
- Flang performance acceptable but requires validation
- Windows LLVM builds are **slow compared to Linux/MacOS**
  - NTFS Filesystem overhead
  - Windows Process creation overhead

Debugging, Linking & Tooling Gaps

LLDB

- **Arm64ec mixed-mode debugging** is unsupported
- **x64 emulated binary debugging** by native Arm64 LLDB not yet possible
- **SVE/SME register visibility** missing
- **Hardware watchpoints** supported is **limited** and **no hardware breakpoints**
- **LLDB-DAP** experience on Windows is **flaky** at best
- **PDB debugging experience** lags behind MSVC
  - No edit-and-continue
  - Native PDB reader in LLDB needs more work — tests still failing

Debugging, Linking & Tooling Gaps

LLD

- **Arm64ec linking** not supported. Projects must still fall back to link.exe
- **LTO** on Windows on Arm needs validation — capability and performance impact are unclear.

Flang

- lacks **MSVC-style command-line** driver compatibility, unlike ifort on Windows x64
- No performance data available. SPEC 2017 does not build out of the box.

Sanitizers

- Support on Windows on Arm remains incomplete and poorly validated
- Not part of native Windows on Arm releases.

Armv9 Enablement Path

LLVM is ready — platform + hardware must catch up

- **LLVM already supports Armv9 features**
  - SVE, SME, PAC, BTI in the AArch64 backend
- **Windows debugging infrastructure is still behind**
  - SVE/SME register state not yet exposed to LLDB
- **Hardware availability is limited on Windows**
  - SVE exists in the cloud today, but not widely validated for WoA
- **Windows ABI requirements** for Armv9
  - Unwind info and debug state must be fully defined and implemented

Testing: CI and Releases

- Linaro maintains multiple Arm64 Windows **buildbots**
  - Single-stage bots are stable
  - Testing Clang, Flang, LLD, LLDB, OpenMP, and compiler-rt
  - 2-stage and Test-Suite bots face stability and resource constraints
  - LNT recently enabled and runs a staging testsuite Flang buildbot
  - Testsuite only runs Flang unit tests, full llvm-testsuite enablement in the works

Testing: CI and Releases

- Linaro produces native **Windows on Arm LLVM Release**
- **GitHub Actions** native Windows on Arm runners now available
- Windows on Arm release will migrate to Github actions soon.
- **SVE testing and validation** suffers from Armv9 hardware availability

GCC Ecosystem Growth & Limitations

GCC Status on Windows on Arm

- GCC cross-compiler for Windows on Arm is available now.
- C++ exception handling (SEH) not fully enabled — work in progress
- Toolchain still cross-compile only — native GCC runtime evolving

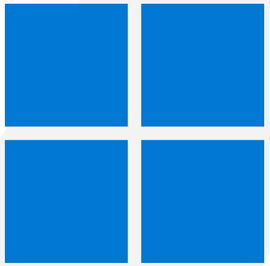
ARM

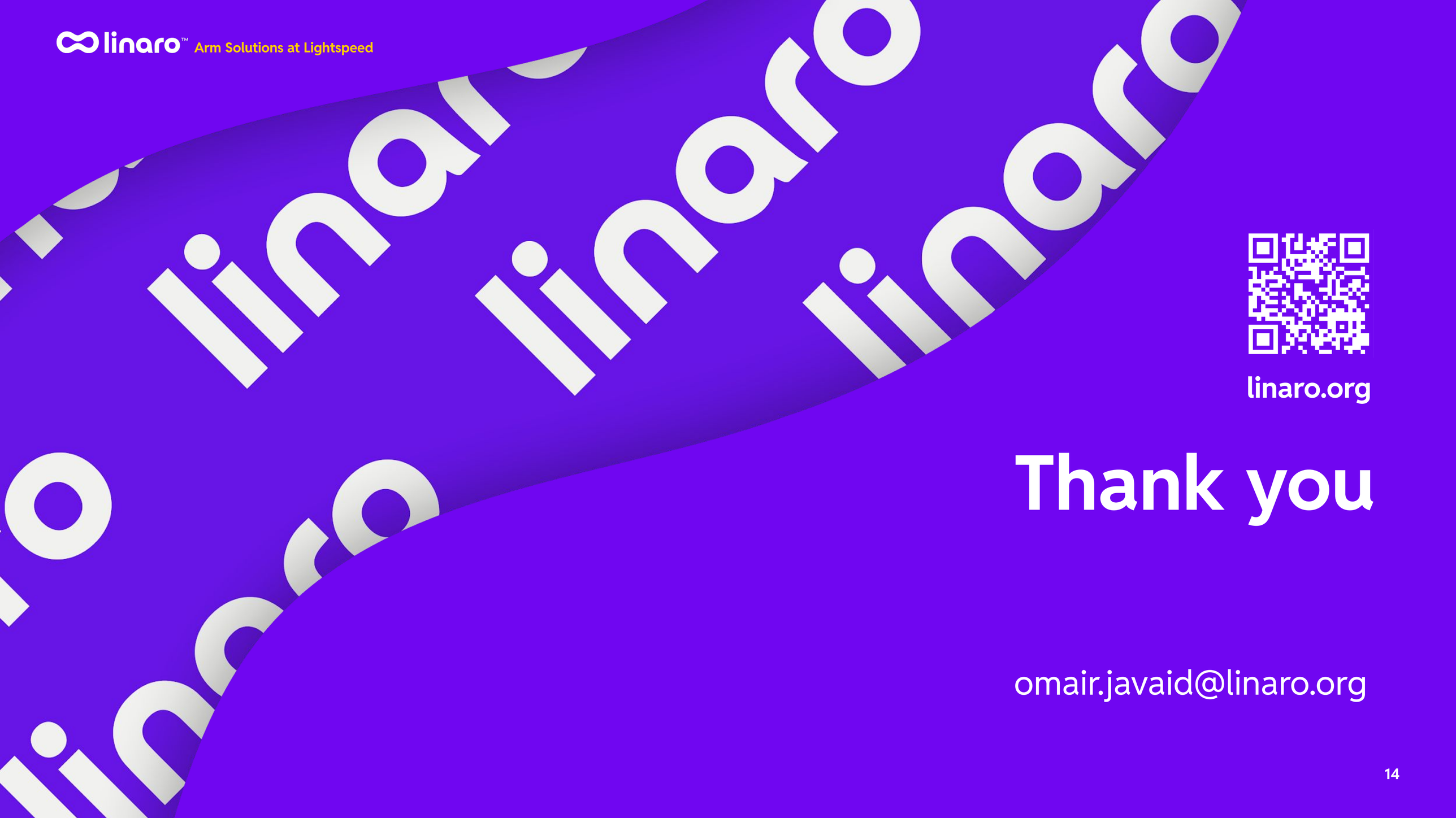
GCC Ecosystem Growth & Limitations

Why GCC Helps LLVM

- Enables more llvm-testsuite coverage — GNU-dependent tests can run
- Expands ecosystem testing — unblocks binutils and various GNU-first tools
- Allows validation and performance comparisons

ARM





linaro.org

Thank you

omair.javaid@linaro.org