Precision Refinement for Media-Processor SoCs: fp32→ fp64 on Myriad

Tomasz Szydzik[1] David Moloney[2]

[1]Insititute for Applied Microelectronics, University of Las Palmas of Gran Canaria, [2]Movidius Ltd.

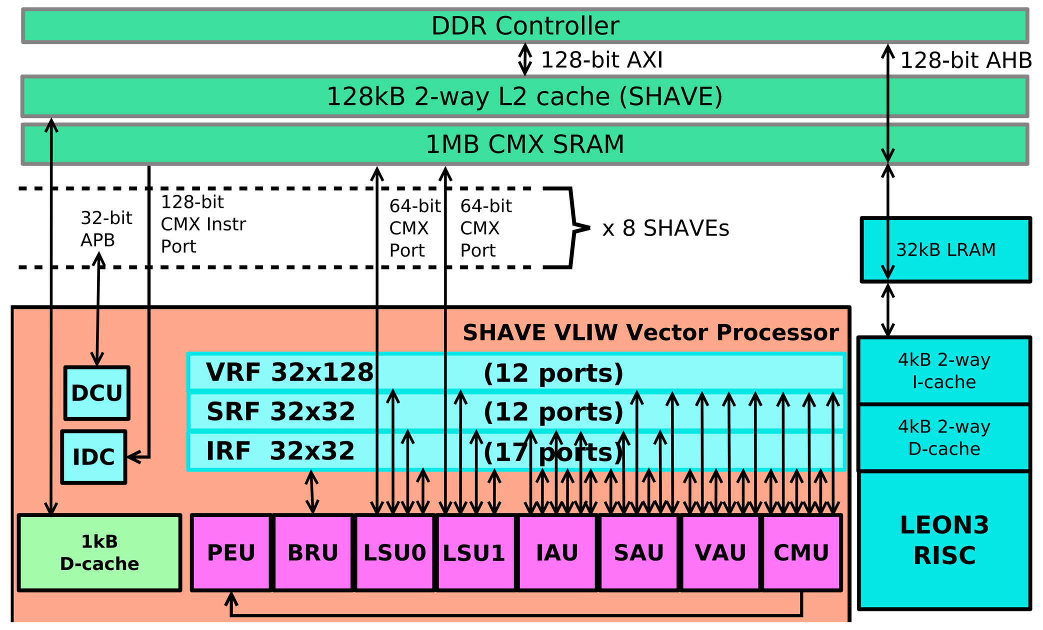
Myriad 1 media-processor SoC

Myriad 1 media-processor SoC [1]

Myriad architecture prioritises power-efficient operation and area efficiency. In order to guarantee sustained high performance and minimise power the proprietary SHAVE (Streaming Hybrid Architecture Vector Engine) processor was developed. Data and Instructions reside in a shared Connection MatriX (CMX) memory block shared by all Shave processors. Data is moved between peripherals, processors and memory via a bank of software-controlled DMA engines.

Myriad 1 architecture highlights [1]

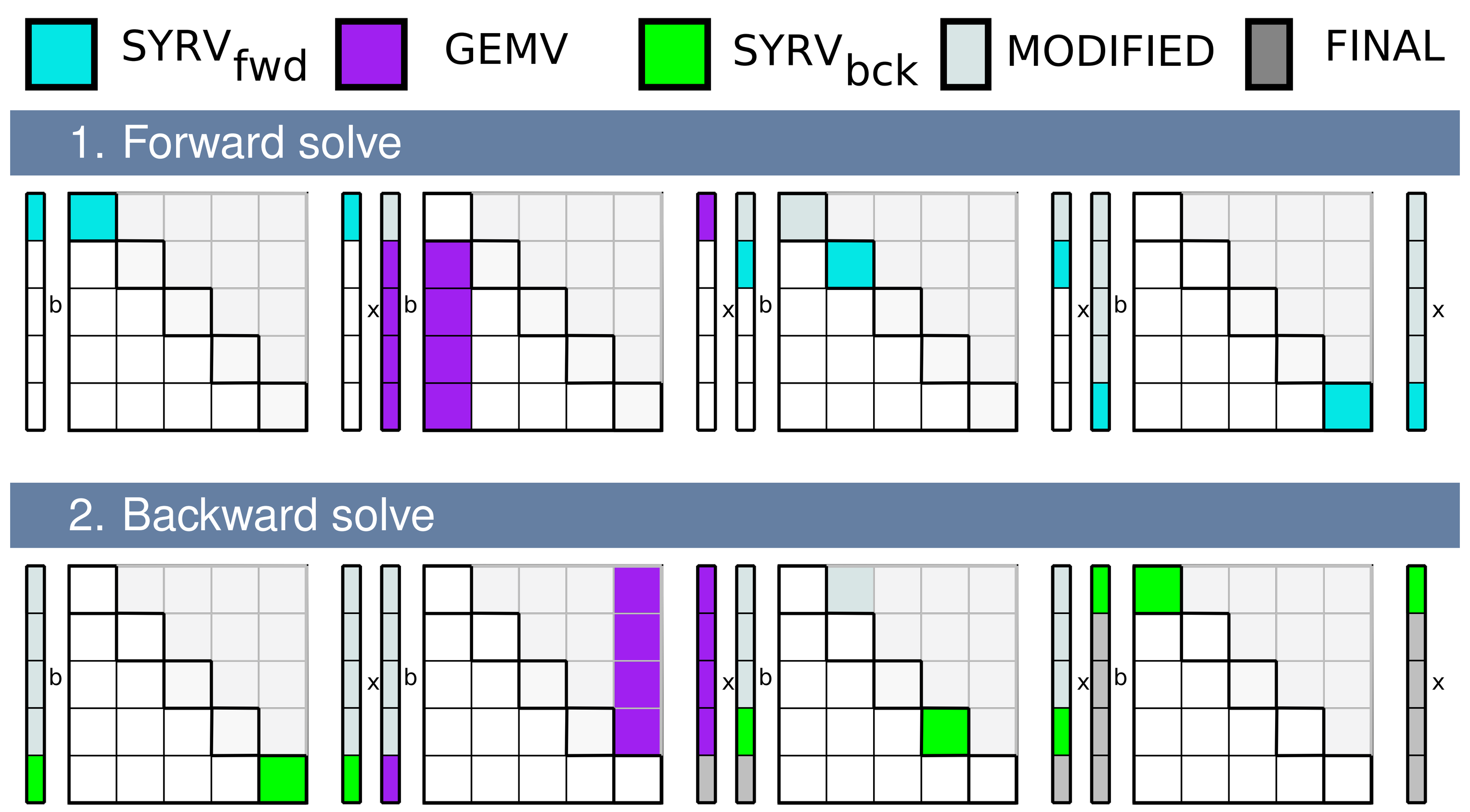
- 65nm ultra-low power architecture ($\leq 0.35W@180MHz$) with 11 power islands.
- ISA-level hardware support for SIMD, matrix transpose, sparse data, sqrt@fp16, predicated execution, etc.
- Heterogeneous SoC: 1 Leon3@fp64 + 8 SHAVE@fp32.
- 32KB LRAM, 1MB CMX, 16/64MB DDR, DMAs.
- Power efficiency of 1Tops/W (max 8-bit equivalent).
- No fp64 on Shaves.



Parallelism exploitation

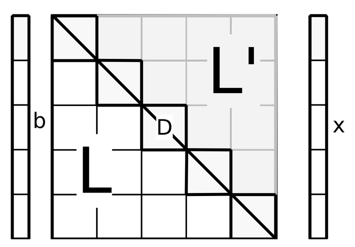
Our approach: process tile-by-tile [3]

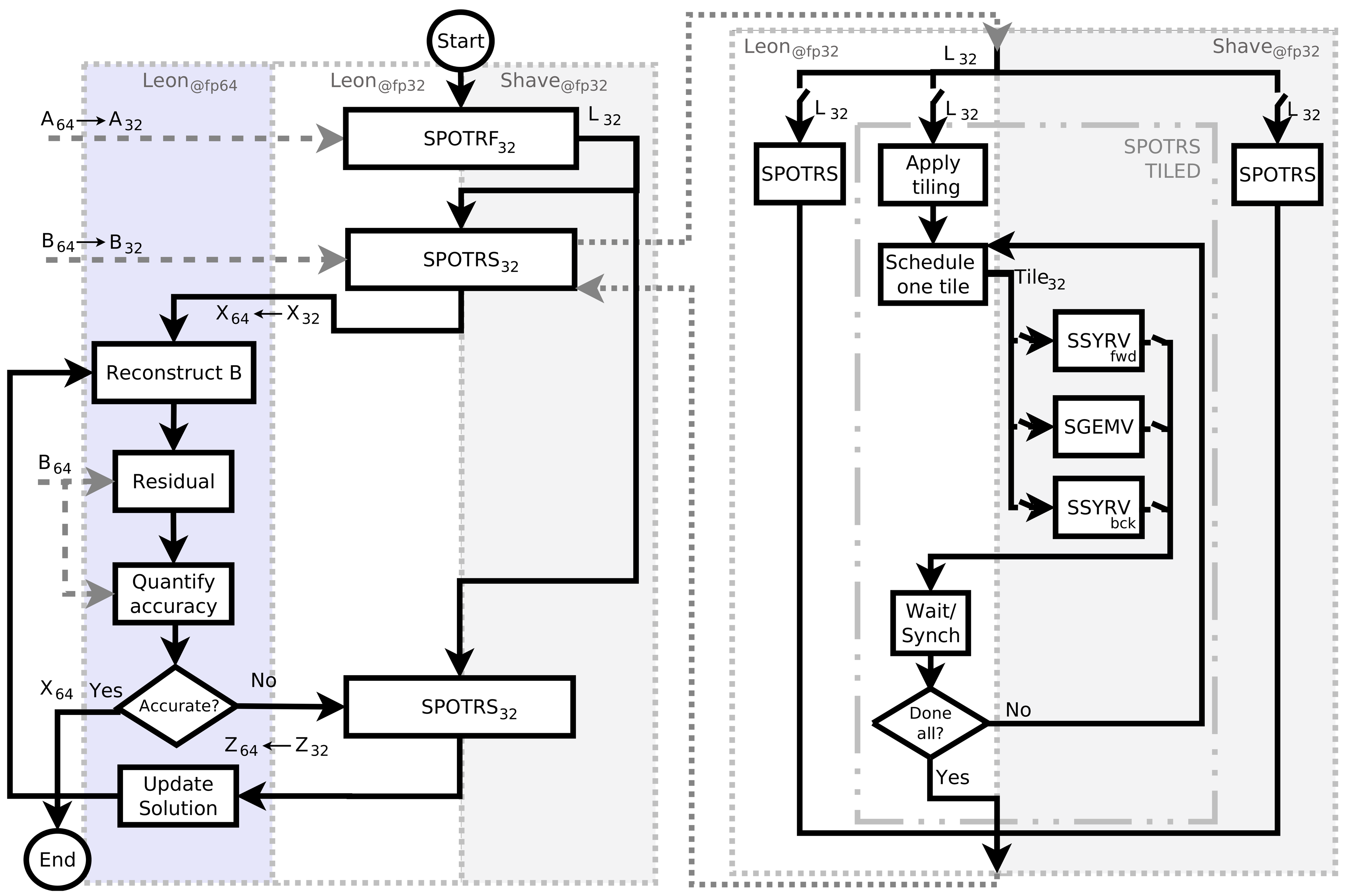
- Partition SPOTRS into sub-ops: $SYRV_{fwd}$, GEMV, and $SYRV_{bck}$.
- Process tile-by-tile (scheduler & fp64@Leon).
- Schedule on multiple tiles concurrently.
- Tiered code: naive C + intrinsics + assembly kernels.



POTRS sub-ops

1. $SYRV_{fwd}$: $x = D \setminus b$
2. GEMV: $b = b - L * x$
3. $SYRV_{bck}$: $b = D' \setminus x$



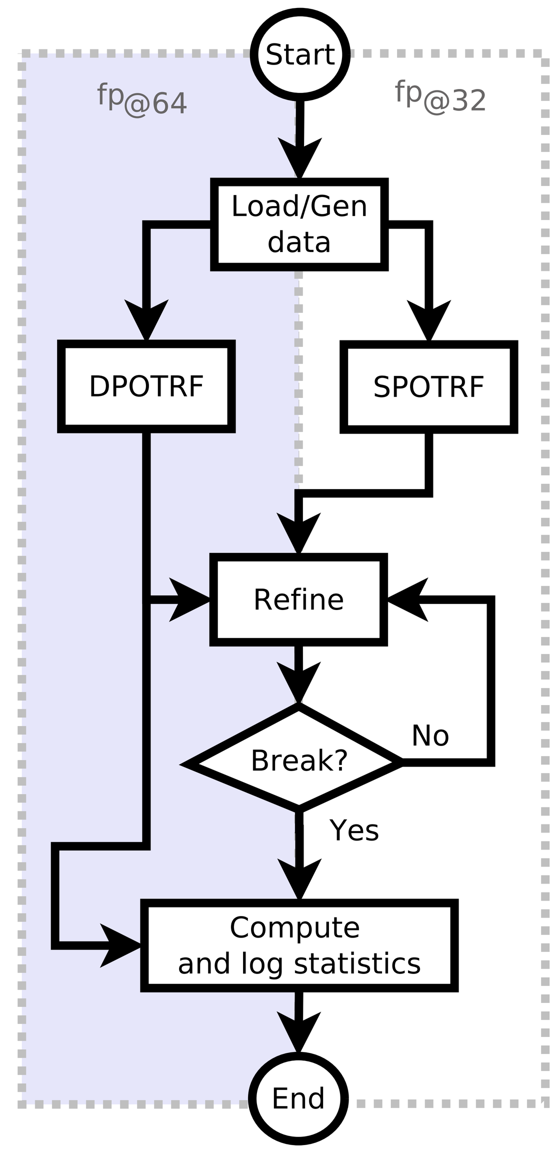


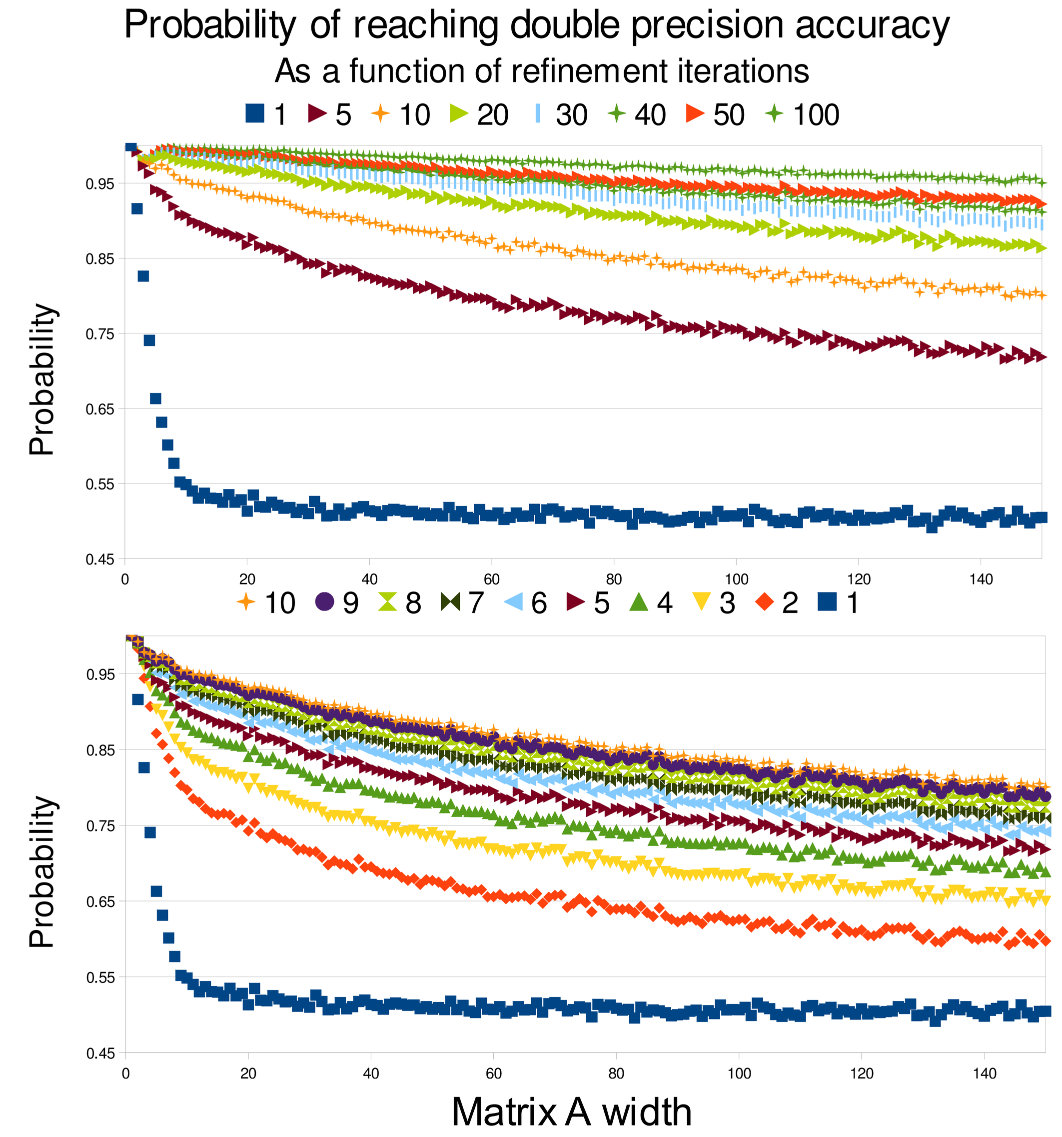
Fp64 on Myriad: mixed precision with $fp32 \rightarrow fp64$ refinement [2]

Study case: matrix decomposition with precision refinement [3]

If A and b are known and A is symmetric and positive definite, then we can solve the linear equation $A \cdot x = b$ by first computing the Cholesky matrix decomposition (SPOTRF) $A = L \cdot L^T$, then solving (SPOTRS) $L \cdot y = b$ for y by forward substitution, and finally solving $L \cdot x = y$ for x by back substitution [4]. The solution x is iteratively updated with values z^i derived from the residual computer for the i^{th} iteration until x^i is accurate enough .

1. $A_{(32)}, b_{(32)}, r_{(32)} \leftarrow A_{(64)}, b_{(64)}, b_{(64)}$
2. $L_{(32)}, L^T_{(32)} \leftarrow SPOTRF(A_{(32)})$
3. repeat:
4. $z^{(i)}_{(32)} \leftarrow SPOTRS(L_{(32)}, L^T_{(32)}, r^{(i)}_{(32)})$
5. $z^{(i)}_{(64)} \leftarrow z^{(i)}_{(32)}$
6. $x^{(i+1)}_{(64)} \leftarrow x^{(i)}_{(64)} + z^{(i)}_{(64)}$
7. $r^{(i+1)}_{(64)} \leftarrow b_{(64)} - A_{(64)} x^{(i)}_{(64)}$
8. $r^{(i+1)}_{(32)} \leftarrow r^{(i+1)}_{(64)}$
9. until $x^{(i+1)}_{(64)}$ is *accurate enough*

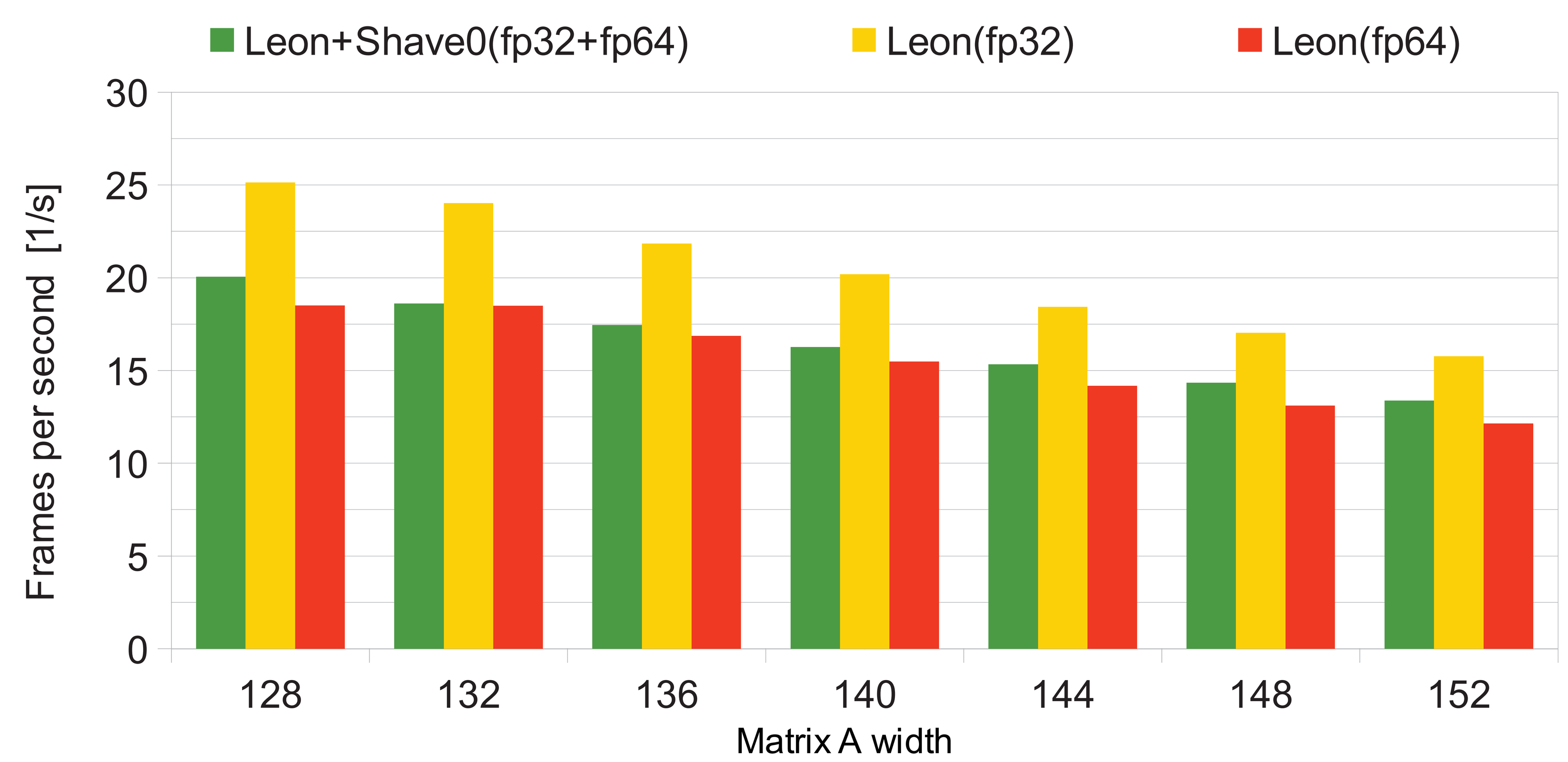


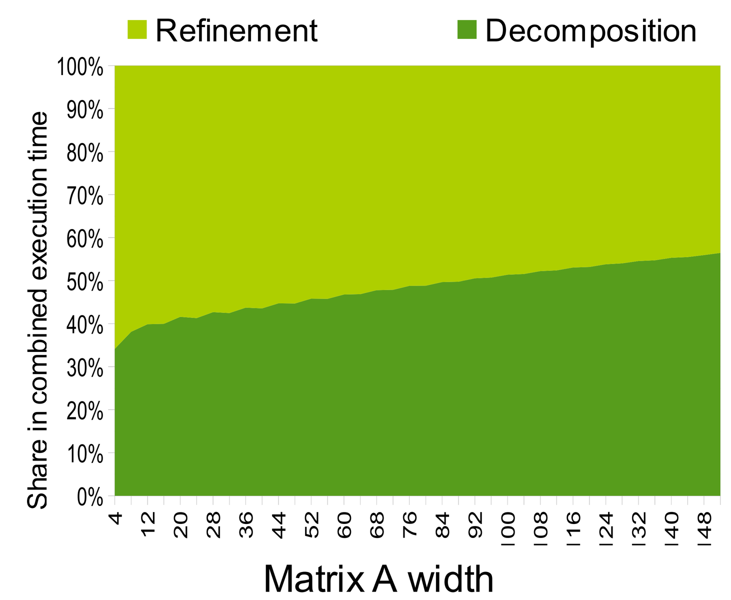


Octave Prototyping

- Refinement iterations: $1 \rightarrow 100$.
- Matrix width: $1 \rightarrow 150$.
- Input:
  - random (10k tests),
  - real-life (one test).
- Versions:
  - Octave-only code,
  - Octave with encapsulated C code.

Results and future work





Current state

- Larger matrices require more iterations (≥ 6)
- Refinement share oscillating around 50%.
- 50% of refinement execution time is spent on Leon.
- Refinement on Leon limits the performance.
- Little room for speed-up on Leon side.
- Shave side could be speed-up ∽ 3 times easily.

Future work

- Overlap i^{th} refinement with $i^{th} - 1$ refinement (static Shaves allocation of $3 + 5$).
- Port to Myriad 2 (expected speed-up of $\geq 8\times$ [1]).

[1] David Moloney, *1TOPS/W Software Programmable Media Processor*, HotChips 2011 (HC23), Stanford, California, August 23rd, 2011.

[2] Erwin Kreyszig, *Advanced Engineering Mathematics (tenth edition)*, John Wiley & Sons, Inc., 2011.

[3] Jakub Kurzak, Alfredo Buttari, and Jack Dongarra, *Solving Systems of Linear Equations on the CELL Processor Using Cholesky Factorization*, Transactions on Parallel and distributed systems, vol 19., NO. 9, 2008.

[4] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing (second edition)*, Cambridge University Press, 1992.

Acknowledgement

This work has been carried out under the HiPEAC PhD industrial internship programme and Excess project (www.excess-project.eu) while being co-founded by Movidius and the European Union under the grants EU ICT-287759 and 611183.