

14.4 A 21.5M-Query-Vectors/s 3.37nJ/Vector Reconfigurable k-Nearest-Neighbor Accelerator with Adaptive Precision in 14nm Tri-Gate CMOS

Himanshu Kaul, Mark A. Anders, Sanu K. Mathew, Gregory Chen, Sudhir K. Satpathy, Steven K. Hsu, Amit Agarwal, Ram K. Krishnamurthy

Intel, Hillsboro, OR

Energy-efficient k-nearest-neighbor (kNN) computations are key building blocks for computer vision, classification, and machine-learning workloads [1-3]. Determining distances to high-dimensional vectors within a large vector database results in high compute cost. Adaptive precision improves energy efficiency by eliminating a majority of vectors without costly full-precision computation, with as-needed precision refinement to guarantee kNN accuracy of closely matched vectors. A special-purpose on-die kNN accelerator with 128-dimensions by 128 parallel reference vectors, targeted across mobile SoCs to multi-core microprocessors, and reconfigurable for either Manhattan or Euclidean distance, is fabricated in 14nm tri-gate CMOS [6]. Partial distance compute circuits, 2b window-based sort, MSB-to-LSB-based selective distance refinement, robust ultra-low voltage circuits, and state tracking control to selectively resume next-nearest candidates enable nominal energy efficiency of 3.37nJ/query vector or 9.7TOPS/W (measured for 21.5M vectors/s, 16 cycles/vector at 750mV, 25°C) with a dense layout occupying 0.333mm² (Fig. 14.4.7) while achieving: i) scalable performance up to 26.4M vectors/s, 114mW measured at 850mV, ii) 2-cycle latency and 43pJ energy to find each subsequent nearest neighbor, iii) up to 5.2× higher throughput while maintaining full-precision kNN accuracy, iv) 16× search-space reduction for next-nearest neighbor, v) ultra-low voltage operation measured at 360mV, 1.1M vectors/s, 1.44mW, and vi) peak energy efficiency of 1.23nJ/vector at 390mV (near-threshold), 25°C.

The kNN accelerator finds the nearest-neighbor indices using in-memory distance-compute circuits, a distributed sort network and efficient incremental search (Fig. 14.4.1). Parallel Manhattan and Euclidean distance computations to the 128 reference vectors (128-D per vector) start with low-accuracy from the MSB using partial distance compute circuits. The sort network finds the minimum within current accuracy bounds, eliminating vectors outside these bounds for up to 6× compute reduction. The partial distance compute circuits refine the remaining distances, with a successive 2b of the accumulated distance available for further sorting in each iteration. Search-space control circuits coordinate these partial distance iterations, vector eliminations and 2b sorting windows until the nearest neighbor is found. Since the last eliminated group of vectors will contain the next nearest neighbor, tracking these groups enables an efficient incremental search within only the last group. Early vector elimination improves throughput up to 5.2×, and narrow datapath width reduces accelerator area by 46%, while maintaining full-precision kNN accuracy.

Complete Euclidean and Manhattan distances from dimension-wise 8b absolute differences require accumulation over 6 and 4 partial-compute iterations, respectively (Fig. 14.4.2a). Iterative refinement of accumulated distance is interleaved as needed with successive sort operations on 2b windows from MSB to LSB. Area-optimized absolute difference uses multi-bit latches for reference vector storage and ripple-carry static mirror adders (Fig. 14.4.2b). The MSB carry-out connects to the LSB carry-in for 2's complement absolute value increment, replacing a separate incrementer for 30% lower area. Use of partial-squaring operations reduce each 8b multiplier to a pair of 2b multipliers, with the shuffle circuit selecting the appropriate inputs from the absolute difference. Narrow partial distance bit-widths also reduce compressor tree area and energy. The 4b products from each dimension are added separately to relocate the left-shift and zero-select circuits after the compressor tree, reducing them from one per dimension to one per vector. The shared partial-squaring and compressor-tree circuits are multiplexed between every pair of vectors using the *vector select* signal for 28% lower area. The previously accumulated distance is shifted left 2b in every compute iteration, enabling replacement of an 11b right shift in the critical path for incremental distance alignment with only a 1b right shift (Fig. 14.4.3). This also decreases the radix-2 carry-merge adder width to 12b for incremental distance summation, with a non-critical 10b incrementer for the upper bits. These optimizations reduce accumulator delay by 37% and area by 48%. Each distance refinement can affect previously sorted 2b windows by at most 1, which must be included in each partial-sort iteration. Therefore, the 12:1 multiplexers forward a

carry-out or prior difference (Y) along with the correct accumulator 2b window to the sort network. The *precise* signals indicate whether all other *psums* exceed the minimum by more than 1, signifying that the same minimum has been found as with full-precision calculations. Decreasing comparator width from 23b to 3b simultaneously reduces sort network area by 3.6× and latency by 2×. Overall energy to find the nearest neighbor reduces up to 3× compared to a full-precision design.

Search-space control enables tracking: i) the vectors that could be possible winners until only one remains, ii) the current 2b window indicated by a global pointer, and iii) 2b windows with eliminated vectors for the next nearest neighbor search. The control-circuit location is divided among vector-compute blocks and a centralized global control (Fig. 14.4.4). The *compare psum* circuit uses the global minimum to determine vector elimination and nearest neighbor selection. Early removal of vectors while moving MSB to LSB for nearest-neighbor search results in 47% compute reduction and a 2b distance sort reduction of 3.8×. The global binary mask stores a one when vectors are eliminated as the global pointer advances MSB to LSB. The global pointer rolls back to the closest 2b window in this binary mask since the vectors removed in this window contain the next-nearest neighbor. Each vector stores the pointer position at elimination, and during rollback the *compare pointer* circuits re-enable only the closest vectors. This search for the next nearest neighbor within a selective subset of the remaining vectors results in 16× search-space reduction.

The adaptive precision kNN accelerator finds the nearest neighbor out of 128×128-D randomly generated vectors for 21.5M query vectors/s at 16 cycles/vector with Manhattan distance measured at 338MHz, nominal 750mV, 25°C (Fig. 14.4.5). Corresponding total power consumption is 73mW (1.8mW active leakage component), resulting in energy-per-query vector of 3.37nJ or 9.7TOPS/W (10P = 8b add/multiply/compare). As k scales from 1 to 10, mean increase in latency and energy per k is 2 cycles and 43pJ. Nominal Euclidean operation is 20.3M vectors/s at 17 cycles/vector, consuming 78mW or 3.84nJ/vector. Mean increase in latency and energy per k is 4 cycles and 87pJ. Peak Manhattan (Euclidean) performance scales up to 26.4M (24.8M) vectors/s with power consumption of 114mW (123mW) at 850mV (Fig. 14.4.6). Datapath circuits optimized for reliable ultra-low-voltage operation [4] enable robust functionality measured down to 360mV, with k=1 throughput scaling to 1.1M vectors/s, 1.44mW for Manhattan and 1.04M vectors/s, 1.63mW for Euclidean computations. Energy decreases with supply voltage for a peak efficiency of 1.23nJ per query vector or 26.5TOPS/W with Manhattan (k=1) mode at 390mV (near-threshold), representing a 2.73× energy efficiency improvement over nominal 750mV operation. Energy-efficiency measurements also improve with algorithm-dependent SURF-128 [5] scale- and rotation-invariant feature vector matching for object recognition. These feature vectors achieve an additional 15% (Manhattan) and 20% (Euclidean) energy savings at 750mV, resulting in 2.86nJ/vector and 3.07nJ/vector, respectively.

Acknowledgments:

The authors thank M. Haycock, M. Mayberry, H. Jiang, S. Borkar, V. De, M. Kounavis and F. Ghenassia for encouragement and discussions. This research was, in part, funded by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References:

- [1] Y. Kim et al., "A 0.5V 54pW Ultra-Low-Power Recognition Processor with 93.5% Accuracy Geometric Vocabulary Tree and 47.5% Database Compression," *ISSCC Dig. Tech. Papers*, pp. 330-331, 2015.
- [2] I. Hong et al., "A 125,582 vector/s Throughput and 95.1% Accuracy ANN Searching Processor with Neuro-Fuzzy Vision Cache for Real-time Object Recognition," *IEEE Symp. on VLSI Circuits*, pp. 184-185, 2013.
- [3] T.-W. Chen et al., "A Multimedia Semantic Analysis SoC (SASoC) with Machine-Learning Engine," *ISSCC Dig. Tech. Papers*, pp. 338-339, 2010.
- [4] S. Hsu et al., "A 280mV-to-1.1V 256b Reconfigurable SIMD Vector Permutation Engine with 2-Dimensional Shuffle in 22nm CMOS," *ISSCC Dig. Tech. Papers*, pp. 178-179, 2012.
- [5] H. Bay et al., "SURF: Speeded Up Robust Features," *J. Computer Vision and Image Understanding*, 110(3):346-359, 2008.
- [6] C.-H. Jan et al., "A 14 nm SoC Platform Technology Featuring 2nd Generation Tri-Gate Transistors, 70 nm Gate Pitch, 52 nm Metal Pitch, and 0.0499 μm² SRAM Cells, Optimized for Low Power, High Performance and High Density SoC Products," *IEEE Symp. on VLSI Tech.*, pp. 12-13, 2015.

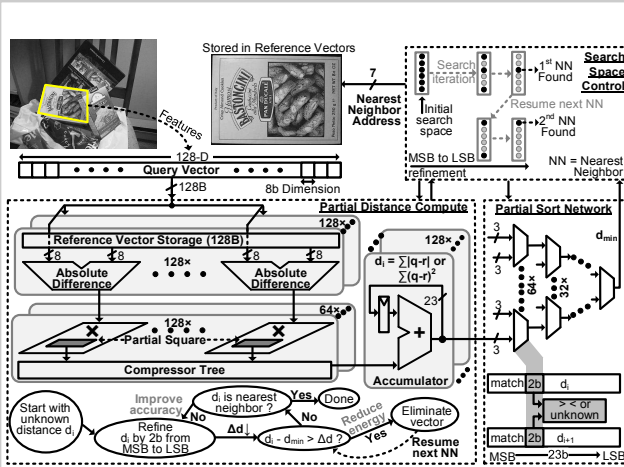


Figure 14.4.1: kNN accelerator organization.

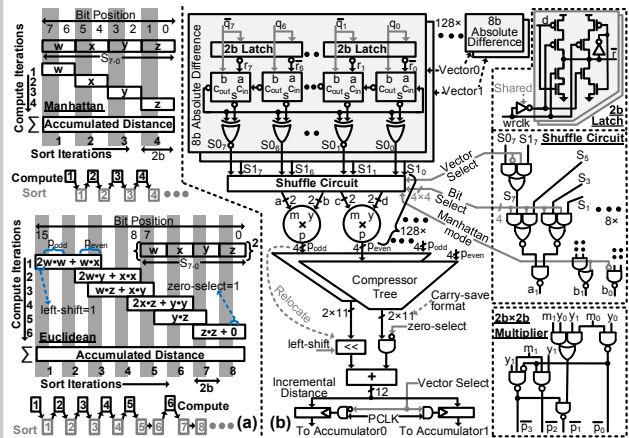


Figure 14.4.2: Partial distance compute (a) iterations and (b) circuits.

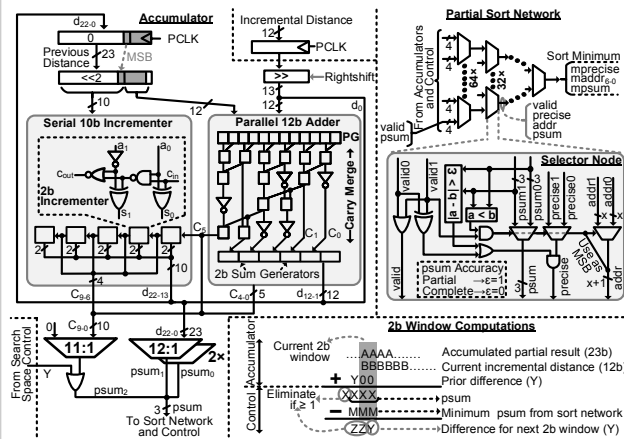


Figure 14.4.3: Accumulator and sort circuits.

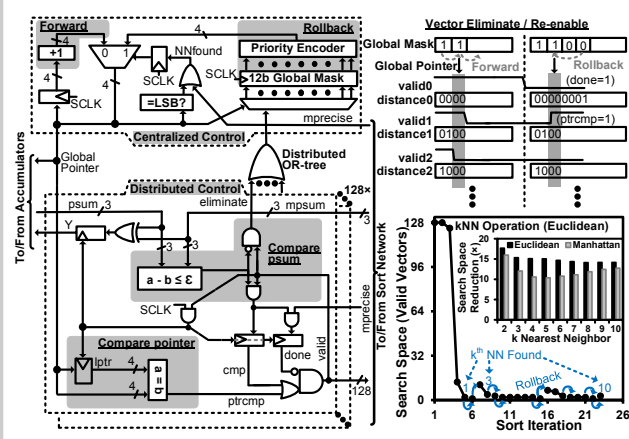


Figure 14.4.4: Search space control circuits.

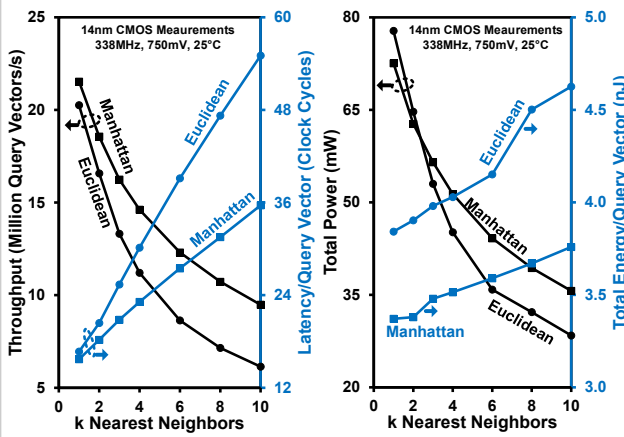


Figure 14.4.5: 14nm CMOS measured throughput, latency, total power and energy vs. number of nearest neighbors computed (k).

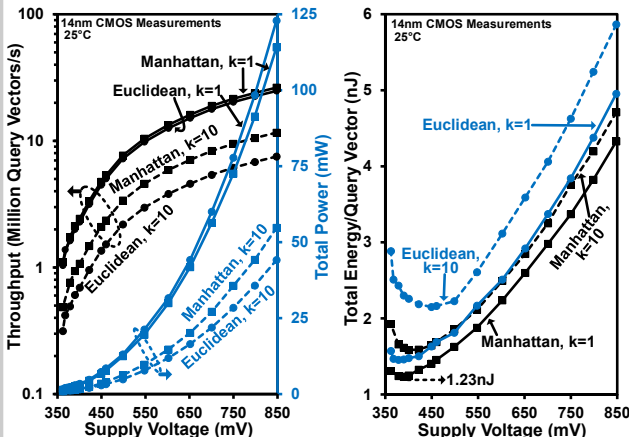


Figure 14.4.6: 14nm CMOS measured throughput, total power and energy vs. supply voltage.

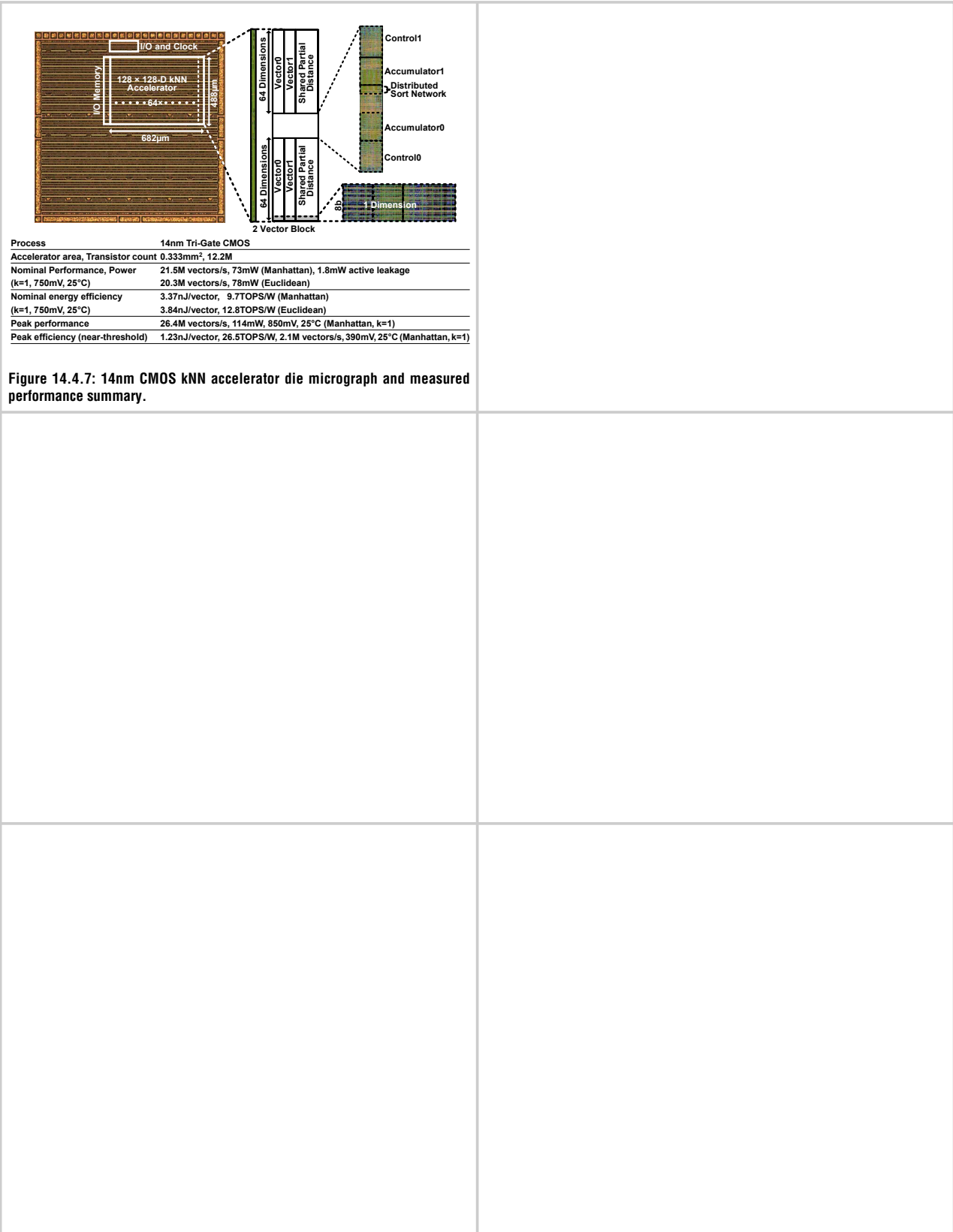


Figure 14.4.7: 14nm CMOS kNN accelerator die micrograph and measured performance summary.