**3.B Accelerators for Machine Learning: Rob**

⟨⟨Can you say more about the applications, design challenges, and programmability issues? And about programming via a fine-grain (virtual) ISA, with a compiler, and not just a library of coarse-grain operations?⟩⟩

I CAN DO A PICTURE IF YOU WANT IT, BUT NOT SURE THERE IS SPACE…

The key to designing effective accelerators for a domain is to identify the resource-intensive primitives used across key algorithms in that domain, and then design hardware to make these primitive 100x-1000x more efficient (in terms of energy or performance) compared with software execution on more general-purpose CPU, GPU or vector cores. *Probabilistic Graphical Models* (PGMs) [KFBOOK] are a highly promising example of such a set of primitives in *Machine Learning* (ML) [MLBOOK] applications.

PGMs naturally address the two problems at the core of most ML applications: uncertainty and complexity. Roughly speaking, PGMs encode “beliefs” about objects in their nodes, and relationships (joint or conditional probabilities, or other numerical affinities) in edges connecting these objects. *Inference* computations on these graphs can answer questions about probabilities for sets of critical outcomes, or find the most likely joint outcome across all nodes. These methods are finding use in a wide range of tasks, from computer vision, to medical diagnosis, to molecular drug design. Inference methods are themselves remarkably broad as a class of computations, spanning continuous numerical optimization, discrete combinatorial optimization, and even random Monte Carlo-style sampling [ICCVTUT, KFBOOK]. Their broad range of applications and solution methods, combined with the relentlessly increasingly scale of the ML tasks in which we would like to deploy them, makes them an attractive target for us.

We are already seeing some nascent research activity in novel architectures, e.g., GPU solutions [CUDACUTS] and even a custom VLSI design for one popular inference method (loopy belief propagation) [LBPVLSI]. The key problem is how to implement a “macro” framework for a particular class of inference methods – which tend to be relatively fixed, algorithmically speaking – with an appropriately programmable “micro” level core, that allows us to retarget an inference engine from one task (e.g., vision) to another (e.g., protein design) without a detailed redesign of the hardware. We already have experience with fully custom hardware in this realm [CHOI]. For example, a prototype implementation of the convergent TRW-S method [trws] on a high-performance FPGA platform [CONVEY] is running faster than a 90nm custom VLSI competitor [LBPVLSI], and yielding superior results, for a standard stereo vision benchmark (FIGURE). The reason is that our architecture has a much higher bandwidth parallel memory architecture, and is correctly exposing most of the (admittedly limited) concurrency in this method. But this engine is not programmable, which is a key failing if our goal is to accelerate a wide portfolio of ML inference applications Our goal in this effort is to explore how to design the appropriate, fine-grain ISAs needed to allows for easy, compiler-driven mapping from key PGM applications to our accelerators.

An additional attraction of these inference methods is that they seem to be natural targets for a more stochastic silicon fabric. Many of these methods are iterative, and deal with imperfect measures of belief or likelihood as their computational core. There is reason to think that such algorithms are intrinsically more robust to statistical errors in these computations, and that we may be able to trade convergence time (i.e., more cycles of update) for increased tolerance of these low-level fluctuations. We already have significant experience with such stochastic fabrics in the realm of communication ICs (summarized next); we propose to expand our focus to include these important new ML algorithms as well.

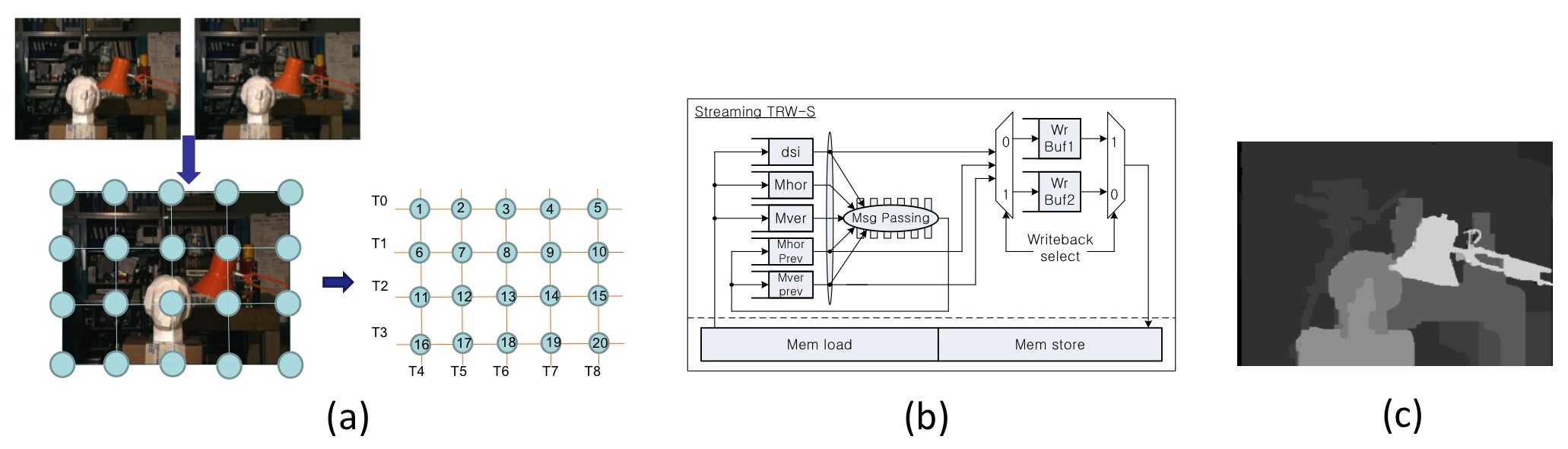


Fig XX. (a) A pair of stereo images is modeled via a 4-connected *Markov Random Field* (MRF) graphical model, from which inference extracts a disparity (i.e., depth) map of the scene. (b) Our streaming, dataflow-style architecture for TRW-S inference [TRWS]. (c) Resulting 16-level disparity map; this accelerator runs at ~40 frames/sec [CHOI].

REFS

KFBOOK: D. Kolla and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 1st Edition, the MIT Press, 2009.

MLBOOK: T. Mitchel, *Machine Learning*, McGraw-Hill, 1997.

ICCVTUT: A. Blake, P. Kohli, M.P. Kumar and C. Rother, “Introduction to MAP Inference in Discrete Models,” tutorial at IEEE Int’l Conference on Computer Vision, 2009. (URL: <http://research.microsoft.com/en-us/um/cambridge/projects/tutorial/>)

CUDACUTS: V. Vineet and P. J. Narayanan, “CUDA cuts: fast graph cuts on the GPU,” In CVPR Workshop on Visual Computer Vision on GPU's, 2008.

LBPVLSI; C.K. Liang, C.C. Cheng, Y.C. Lai, L.G. Chen and H.H. Chen, “Hardware-efficient belief propagation,” IEEE Trans. On Circuits and Systems for Video Technology, vol. 21, no. 5, pp. 525-537, May 2011.

CHOI: J. Choi, “Hardware implementation of MRF MAP estimation on FPGA platform,” Dept of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Ph.D. qualifying examination report, unpublished mss, November 2011.

TRWS: V. Kolmogorov, “Convergent tree-reweighted message passing for energy minimization,” IEEE Trans. PAMI, vol. 28, no. 10, pp. 1568-1583, Oct. 2006.

CONVEY: “Convey HC-1 Family”. [online]. Available: <http://www.conveycomputer.com>.