A 3mm^2 Programmable Bayesian Inference Accelerator for Unsupervised Machine Perception using Parallel Gibbs Sampling in 16nm

Glenn G. Ko[1], Yuji Chai[1], Marco Donato[1], Paul N. Whatmough[1,2], Thierry Tambe[1],
Rob A. Rutenbar[3], David Brooks[1], Gu-Yeon Wei[1]
[1]Harvard University, MA, [2]Arm Research, MA, [3]University of Pittsburgh, PA

Abstract

This paper describes a 16nm programmable accelerator for unsupervised probabilistic machine perception tasks that performs Bayesian inference on probabilistic models mapped onto a 2D Markov Random Field, using MCMC. Exploiting two degrees of parallelism, it performs Gibbs sampling inference at up to 1380x faster with 1965x less energy than an Arm Cortex-A53 on the same SoC, and 1.5x faster with 6.3x less energy than an embedded FPGA in the same technology. At 0.8V, it runs at 450MHz, producing 44.6 MSamples/s at 0.88 nJ/sample.

Introduction

Efficient unsupervised probabilistic machine perception is the next challenge for AI hardware. Prior emphasis on supervised learning with deep neural networks was limited to tasks with large labeled datasets that are expensive to train. In contrast, unsupervised Bayesian models are typically more powerful for problems that (*i*) have scarce labeled data, (*ii*) involve representing uncertainty, and (*iii*) have strong priors. Unfortunately, Bayesian inference workloads do not parallelize well on multi-threaded CPUs, requiring specialized hardware.

This paper describes the first silicon implementation of an unsupervised Bayesian inference accelerator, which computes a 2D probabilistic graphical model, Markov Random Field (MRF), using Gibbs sampling. An existing prior work describes augmenting server-class GPUs with an optical sampling unit to accelerate Gibbs sampling operations on MRFs [1]. Our CMOS-based Parallel Gibbs sampling MRF Accelerator (PGMA) performs chromatic and asynchronous, parallel Gibbs sampling.

Parallel Gibbs Sampling MRF Accelerator (PGMA)

Unsupervised perception tasks are essentially a pixel-labeling problem, solved by reassigning each pixel in an image to a *label* from a given set (Fig. 1). This is done by mapping the problem to an MRF and performing Bayesian inference using Gibbs sampling in order to compute a set of labels that minimize a cost function describing the distribution [1,2]. While the Gibbs sampling and MRF node update steps are traditionally performed in a sequential fashion, PGMA combines (1) asynchronous Gibbs sampling (AGS) and (2) chromatic Gibbs sampling (CGS) [2] to improve parallelization (Fig. 2). CGS allows conditionally-independent nodes in the MRF to be sampled concurrently, but does not relax the on-chip memory required for storing intermediate model parameters, which scale exponentially [2]. AGS parallelizes updates by dividing the MRF into multiple tiles, each sampled as independent graphs with occasional global updates. AGS reduces on-chip memory by only storing data needed for individual tiles.

Fig. 3 shows the 16nm SoC with a dual-core Arm Cortex-A53 cluster with 2MB L2$ and a dual-core PGMA cluster. A second SoC containing an embedded FPGA (eFPGA) is used for comparison [3]. PGMA includes two sub-graph tiles (SGT) that share a global graph cache (GGC) for label values corresponding to the nodes of the MRF. The GGC consists of 3x100KB SRAMs, sufficient for Standard Definition (SD) 640x480 images. The two SGTs copy labels corresponding to a sub-graph partition of the global graph into the 640x2 local graph cache (LGC) and perform Gibbs sampling on the partitioned sub-graph. Each SGT has an input data buffer (IDB) to store *data costs* for the input pixels, using 4x80KB SRAMs (320KB total). IDBs store input data for the sub-graph. Each SGT has four Gibbs samplers (Fig. 4) that perform 2-color chromatic Gibbs sampling on the sub-graphs in the LGC. Each Gibbs sampler contains 4x16KB SRAMs (64KB total) for the smoothness cost table, which is pre-computed and loaded prior to inference (Fig. 4), to impose label smoothness amongst neighboring pixels. Total costs for all possible labels are computed and converted to Boltzmann probability distributions.

Measurement Results

PGMA accelerates any task formulated as a 2D MRF. Here, we consider four examples: (1) image restoration fills in damaged areas in an image, using the *House* (256x256, 256 labels) benchmark [4], (2) stereo matching creates a 3D depth map from a stereo camera image-pair, using the *tsukuba* (384x288, 16 labels) benchmark [5], (3) image segmentation is a binary foreground/background segmentation task using an airplane image (500x333, 2 labels) from VOC2012, (4) sound source separation of audio mixtures from two sources, using spectrograms (513x125, 2 labels) as 2D MRFs [6].

Increasing the number of Gibbs sampling iterations consumes more energy but further minimizes the cost function, improving perceptual quality (Fig. 5). On the four tasks, sequential, CGS, and AGS algorithms all converge to similar solutions, with AGS being the slowest (Fig. 6). However, increased parallelism and memory savings of AGS outweigh this disadvantage.

Fig. 7 compares PGMA to three commercial hardware platforms: Nvidia Jetson TX2 A57 CPU, TX2 GPU, and Xilinx Zync ZCU102 [2]. PGMA is faster than A57 and GPU of TX2. While [2] is faster, PGMA reduces the memory footprint from 4 to 1.46 MB (2.7x reduction) and achieves more than 2x throughput with only 8 samplers compared to 24 at up to 247x lower energy across the four tasks (Fig. 8). Adding more SGTs can further reduce runtime of PGMA at higher silicon area cost.

In the absence of published silicon results, we compare PGMA measurements to that of a mobile CPU and an eFPGA in the same 16nm technology [3]. The CPU version on A53 is sequential (multi-threading yields at most a 10x improvement running on a server-class CPU). Fig. 10 summarizes measured throughput (Mega-Samples/s), further scaled with per-unit power and area, across three operating voltages: V_{MIN}=0.6V, V_{NOM}=0.8V, and V_{MAX}=0.9V. PGMA and eFPGA achieve comparable speedup versus the A53. However, at 0.8V, PGMA achieves sub-nJ/sample operating energy, a 1965x and 6.3x energy reduction compared to A53 and eFPGA, respectively. Per-mm^2 reduction is greater at 4162x and 9.2x, respectively. Fig. 9 shows the annotated die photos and summary of the chip.

Acknowledgements

This work is supported in part by DARPA CRAFT, JUMP ADA, Intel and Arm.

References

[1] Zhang *et al.*, *ISCA*, 2018.
[2] Ko *et al.*,*FPL*, 2019.
[3] Whatmough *et al.*, *VLSI*, 2019.
[4] Besag, *J R STAT SOC B* 1986.
[5] Scharstein *et al.*, *IJCV*, 2002.
[6] Ko *et al.*, *JETC*, 2018.

978-1-7281-9942-9/20/$31.00 ©2020 IEEE

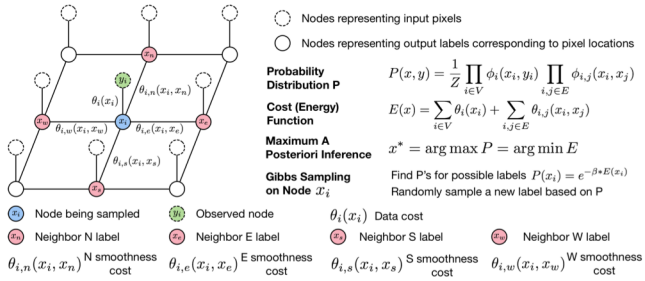


Fig. 1: Bayesian MRF model and Gibbs sampling inference.

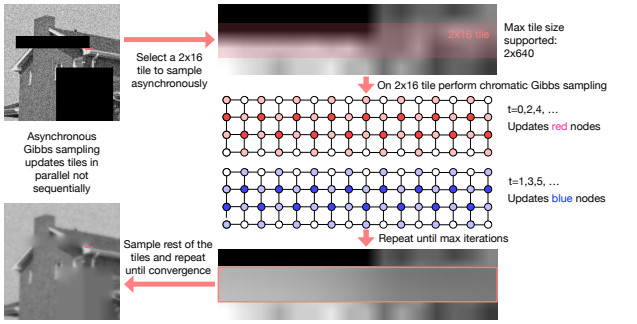


Fig. 2: Unsupervised pixel-labeling task using an efficient tiled asynchronous chromatic Gibbs sampling schedule.

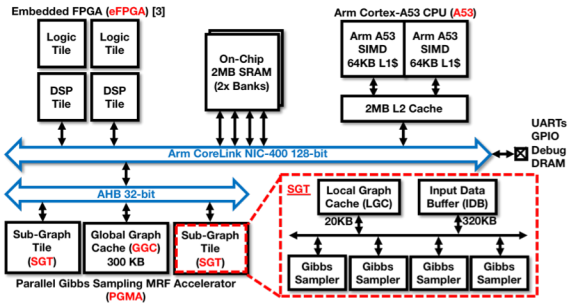


Fig. 3: 16nm SoC with a dual-core Arm Cortex-A53 (2MB L2$), embedded FPGA (eFPGA), and PGMA with 300KB global graph cache (GGC) and sub-graph tile (SGT) cores.

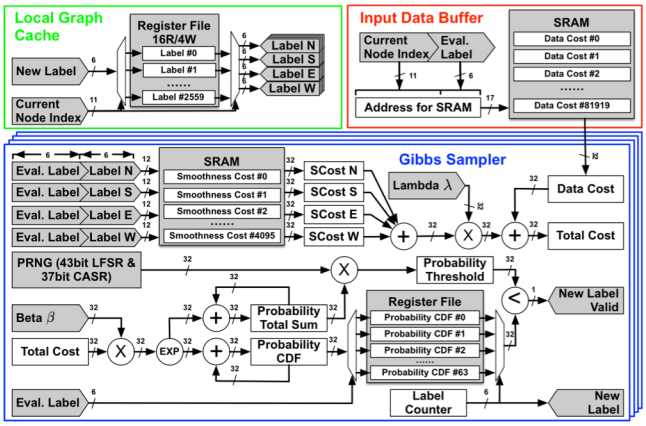


Fig. 4: SGT core contains 320KB input data buffer (IDB), 20KB local graph cache (LGC) and four Gibbs sampler datapaths.

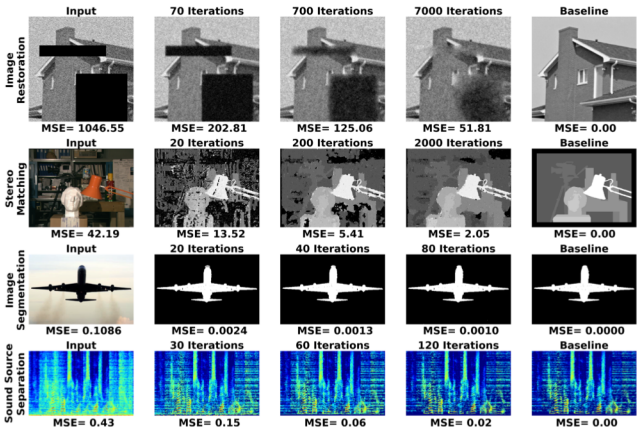


Fig. 5: Unsupervised machine perception tasks: image restoration, stereo matching, image segmentation and sound source separation.

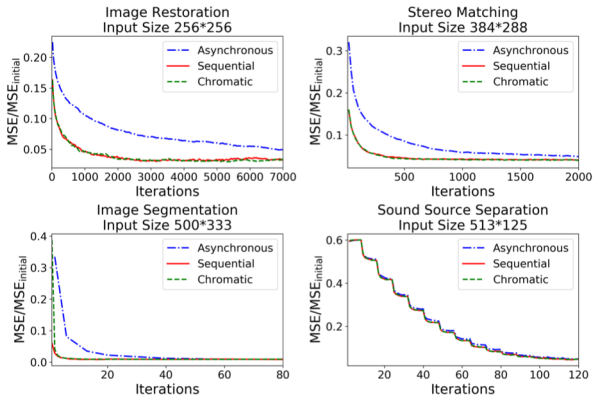


Fig. 6: Reduction of MSE vs. sampling iterations on four workloads.

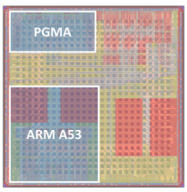
Hardware	Freq. (MHz)	On-Chip Memory	Algorithm Type
Nvidia Jetson TX2 - A57	2000	2 MB	Sequential
Nvidia Jetson TX2 - GPU	1300	2 MB	CGS[a]
Xilinx Zynq UltraScale+ MPSoC ZCU102	150	4 MB	CGS[a]
PGMA (this work)	450	1.46 MB	CGS[a] + AGS[b]

[a] CGS – Chromatic Gibbs sampling [b] AGS – Asynchronous Gibbs sampling

Fig. 7: Comparison on four hardware platforms with frequency, on-chip memory size and supported Gibbs sampling algorithms

	TX2 A57 CPU			TX2 GPU			Zynq ZCU102 FPGA [2]			PGMA		
Workload	Secs	MS/s	MS/s/W	Secs	MS/s	MS/s/W	Secs	MS/s	MS/s/W	Secs	MS/s	MS/s/W
Restore	399.3	0.15	0.14	-	-	-	2.12	24.72	5.46	9.97	44.56	1132.26
Stereo	68.37	0.55	0.51	-	-	-	0.63	55.95	12.36	1.88	109.38	2779.17
Segment	1.33	2.5	2.34	0.09	38.26	100.16	0.04	88.59	19.57	0.02	189.98	4826.99
Sourcesep	2.75	2.45	2.29	0.13	49.83	130.44	0.07	88.59	19.57	0.04	189.98	4826.99

Fig. 8: Comparing performance of four workloads on different platforms to achieve 95% MSE reduction



Process Technology	TSMC 16nm FFC FinFET
Die Size (PGMA size)	5mm x 5mm (2.3mm x 1.3mm)
PGMA Gate Count	~ 2 million
Total SRAM	1.46 MB
Supply Voltage	0.8 V (0.4 – 1 V operational)
Fmax	451.2 MHz
Power @ Fmax	39.3 mW
Model	Markov Random Field
Inference	Gibbs sampling (MCMC)

Fig. 9: Die photos (left) and 16nm test chip summary table (right).

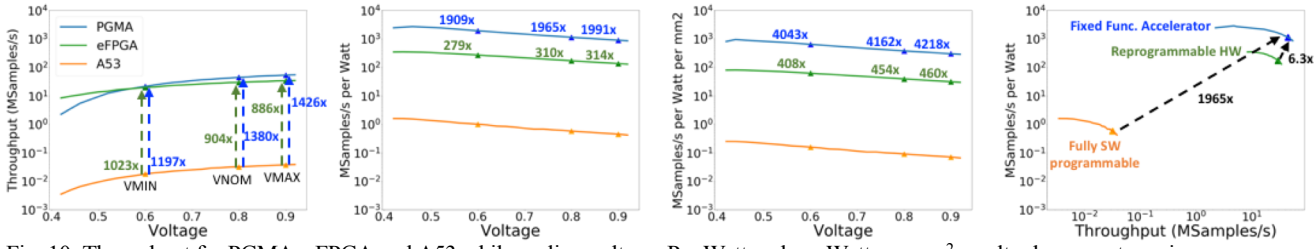


Fig. 10: Throughput for PGMA, eFPGA and A53 while scaling voltage. Per Watt and per Watt per mm^2 results show greater gains.