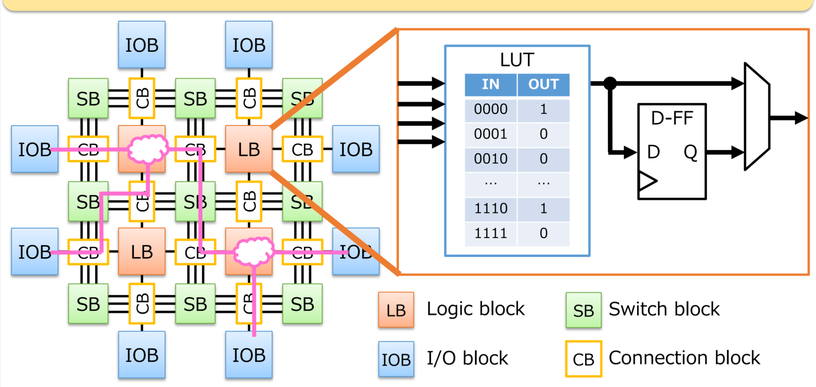
FPGA-based Domain-Specific Accelerator and Application

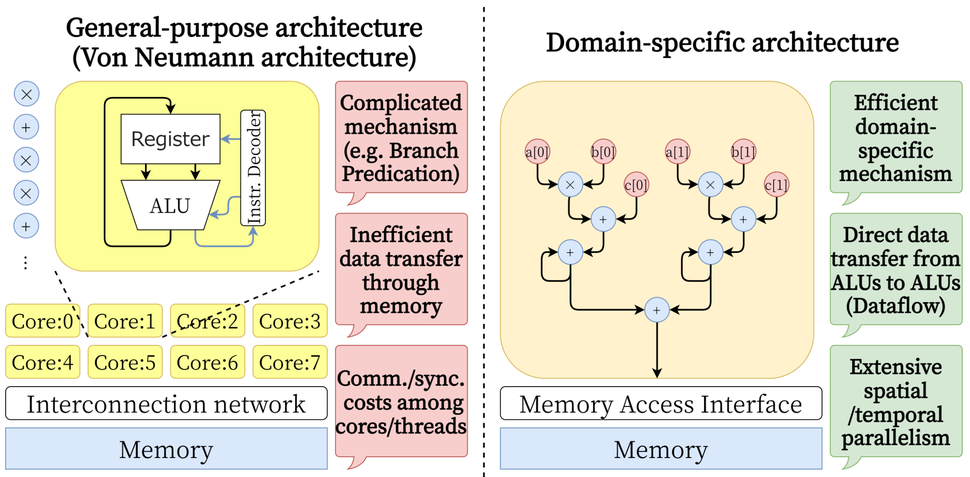
Ryota Miyagi, Lab#8 Nakamura/Takase Computing System Laboratory, IPC, IST, UTokyo

What is FPGA?



- A ***field-programmable gate array (FPGA)*** is a type of programmable integrated circuit.
- Configuring logic and its connection in an FPGA embodies flexible computing as software performs.

What is Domain-Specific Architecture?



- A ***domain-specific architecture (DSA)*** tailored to a given application domain provides superior performance and power efficiency (e.g., Google TPU, Microsoft Catapult)

Core Idea

To use FPGA-based Domain-specific architecture to cherry-pick

- the flexibility and productivity of software and
- the performance and power efficiency of hardware

and accelerate various intensive computations.
(e.g., image processing, robotics, or machine learning)

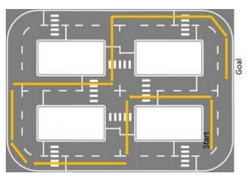
Accelerating Object Detection in Autonomous Mobile Robot

Abstract

- Autonomous mobile robots (AMR) require high-speed, real-time, reliable decision-making under strict power requirements.
- GPU-based: high power consumption 😳
- Cloud-based: communication latency 😳
- FPGA-based: optimal trade-off, balancing modest computational capabilities with low latency and power consumption ⚖

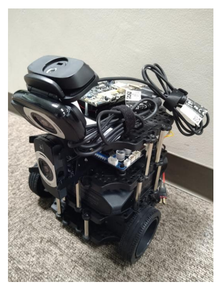
FPT FPGA Design Competition

Objective: Advance FPGA-based AMR with real-time image recognition and decision-making capabilities.
Challenges: Develop an FPGA-based AMR to navigate miniature roads following given routes and complete tasks
Tasks: Recognize and pause at traffic lights and dolls, and identify and avoid obstacles



Zytlebot

- AMR that we develop for the competition
  Zytlebot = Zynq + Turtlebot3
- **Zynq**: Xilinx SoC family that integrates an ARM-based processor and an FPGA
- **Turtlebot**: ROS Compatible open-source AMR platform with easy customization
- Processor-FPGA Co-Design

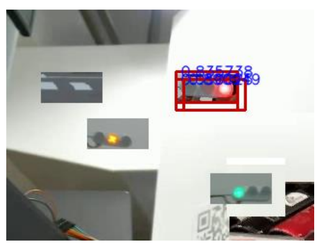


Traffic Light Detection

Input: 240*320pix BGR
Output: red light detected
Linear SVM

- Feature: BGR, HSV, HOG
- 32*64pix window,
- 8pix stride

😇 Each input has 120 windows for inference
=> How do we accelerate it???

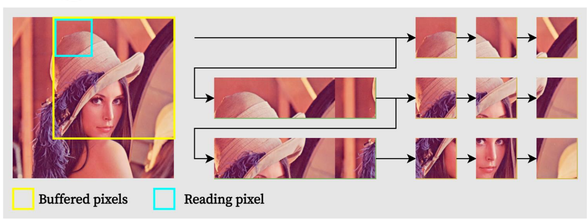


Pipelined sliding window

Dataflow computation for examining a small portion of an image at a time.
In each cycle,

- Sequential fixed-width memory access
- Buffers cover a "window" of the image

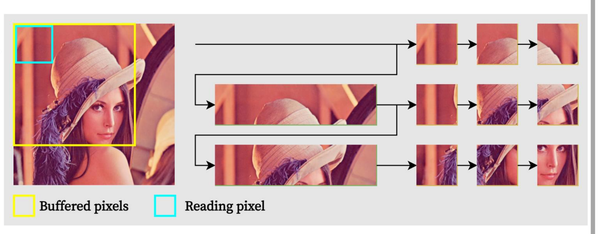
Inference for a given window is done in one cycle
=> 😍 200x faster inference than software



In the case of a 4x4 pixel input, a 3x3 pixel window, and reading one pixel in each cycle

Next cycle

GIF here



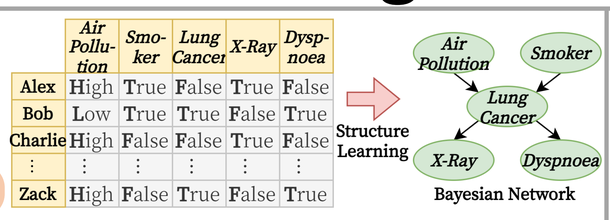
Publications

[1] Ryota Miyagi, Sho Kinoshita, Masashi Oda, Naofumi Takagi, Hideki Takase, "Zytlebot: FPGA Integrated ROS-Based Autonomous Mobile Robot," ICFPT 2021, Auckland, Dec 2021.

Acknowledgments

This work is supported by JST PRESTO JPMJPR18M8.

Accelerating Score-based Structure Learning of Bayesian Networks



Abstract

- A ***Bayesian network*** is a probabilistic model that encodes conditional independence among random variables into a DAG.
- It facilitates decision-making by accurate / robust inference (e.g., cancer screening, anomaly detection in IoT devices, etc.)
- Learning a DAG from data (***structure learning***) is computationally intensive, and speeding up the process is a major concern.

Contribution

Accelerating the bottleneck of score-based structure learning of identifying promising parent variable sets for each variable by
(A) proposing a strategy to shift the problem to a form suitable for parallelization and
(B) designing an FPGA-based domain-specific accelerator for the proposed strategy.

Score-based structure learning

The score is a product of the local scores (LS):

$$p(\boldsymbol{S} \mid G) = \prod_{X \in \boldsymbol{V}} \mathrm{LS}(X, \boldsymbol{Pa_X})$$

where, $\boldsymbol{Pa_X}$ is parent set of X.

😍 Eliminating useless parent sets in advance significantly reduces the search space of DAG.
😇 The local score computation is now the bottleneck. => How do we accelerate it???

(A) Motivation and Strategy

Local scores depend on numerous **counting queries** (e.g. How many records have *Smoker*=*True*, *LungCancer*=False *and others*=*).

- **Baseline**: scanning samples for each query
- **ADtree**: sophisticated tree data structure

👍 High data reusability
👎 Data dependence prevent its parallelization

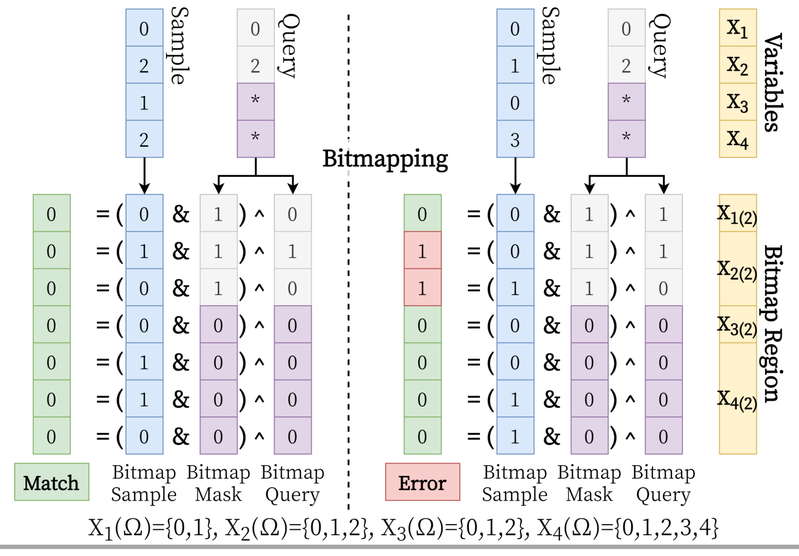
Insight:
data reusability benefit
<< parallelization benefit?

- **Our method**: scanning samples for each query with high parallelism and occupancy

👎 Lower data reusability compared to ADtree
👍 No data dependency, thus high parallelism

Bitmap representation

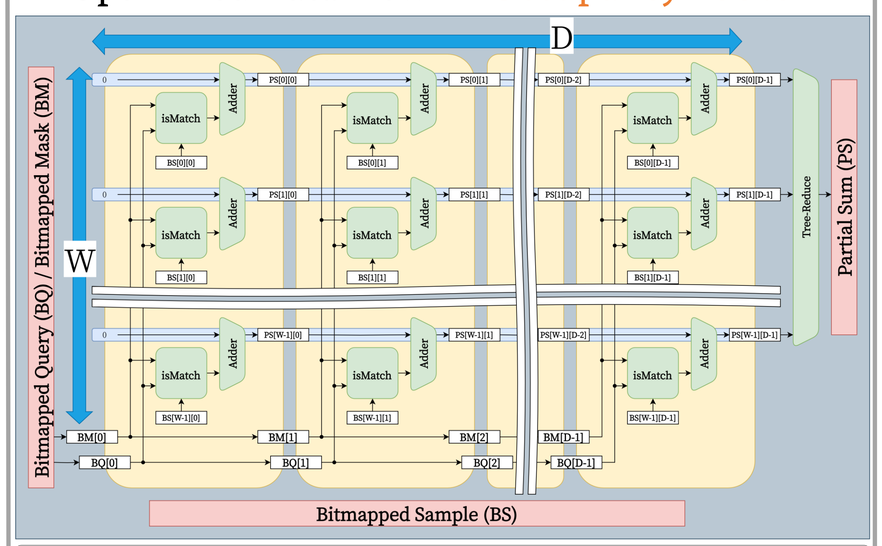
Counting queries and data are bitmapped
😍 FPGA-friendly simple logic => Parallelism ↑
😍 Flattened operation cycles => Occupancy ↑
😍 Uniform handling of problem instances



$X_1(\Omega)=\{0,1\}$, $X_2(\Omega)=\{0,1,2\}$, $X_3(\Omega)=\{0,1,2\}$, $X_4(\Omega)=\{0,1,2,3,4\}$

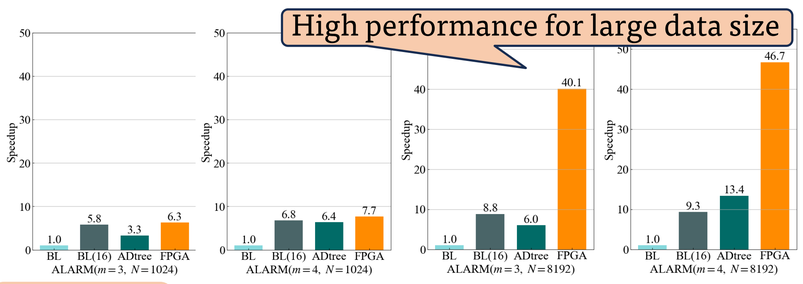
(B) Design and Evaluation

- Placing Processing Elements in parallel/series
  => Spatial / Temporal Parallelism ↑
- Optimized Data Flow => Occupancy 100%



Evaluation

We compared the local score computation time by methods {BL, BL(16), ADtree, Our Method} under the varied conditions: {maximum parent set size m, data size$N$}



Publications

[1] Ryota Miyagi, Ryota Yasudo, Kentaro Sano, Hideki Takase, "Elastic Sample Filter: An FPGA-based Accelerator for Bayesian Network Structure Learning," ICFPT 2022, Hong Kong SAR, Dec 2022.
[2] 宮城竜大, 安戸僚汰, 佐野健太郎, 高瀬英希, "ベイジアンネットワーク構造学習のための可塑性を備えたFPGAアクセラレータ," SWoPP2023, 函館, 2023年7月.

Acknowledgments

This work is supported by JST CREST JPMJCR21D2 and JST SPRING JPMJSP2108.