Reconfigurable Systems for Dynamic Workloads: Design Space Exploration and Resource Management

Presenter: Jiahao Lin

Department of Electrical and Computer Engineering

University of Wisconsin - Madison

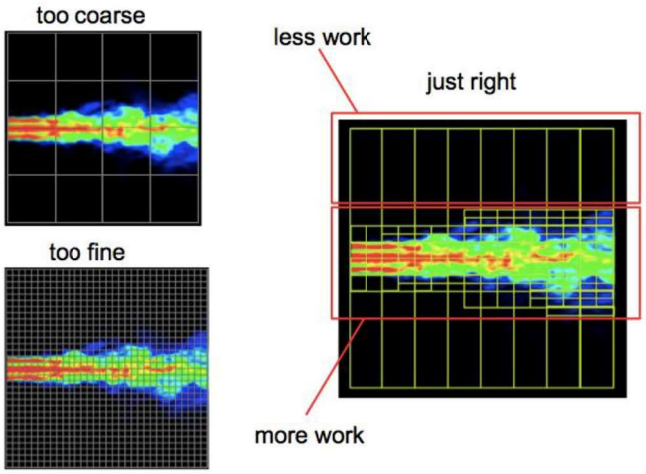
Committee Members: Ali Akoglu, Joshua San Miguel, Tsung-Wei Huang and Umit Y. Ogras (Ph.D. Advisor)

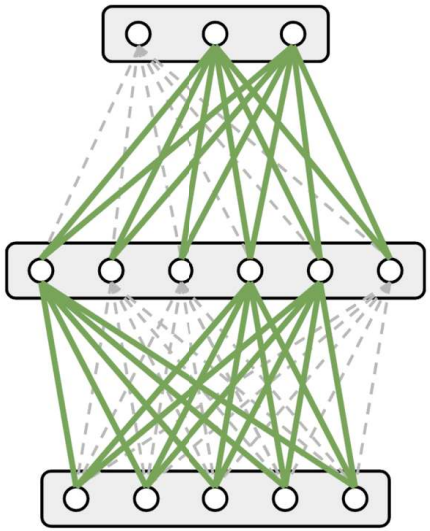
Agenda

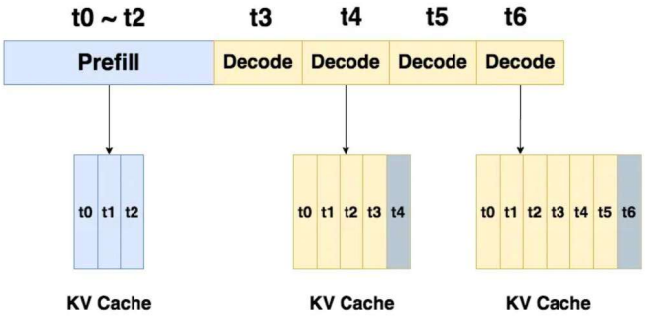
- **Motivation: Reconfigurable Systems and Dynamic Workloads**
- **Preliminary Work**
  - An Overview for Real-Time Spectrum Sensing in Modern RF Autonomy Systems
  - Run-time Dynamic Scheduler and Design-Time Optimization on CGRAs
- **Proposed Work**
  - A Heterogeneous Acceleration Platform for Hybrid LLM Service
  - Resource Management of Heterogeneous Antenna Array
- **Timeline and List of Publications**
- **Conclusions**

Motivation: Run-time Dynamic Workloads

- **Workloads with run-time dynamic changes**
  - Video/image processing: adaptive encoding
  - Sparsity of activations in neural networks: zero-value bypass
  - LLM Services' context length and Prefilling/Decoding: chat/summary





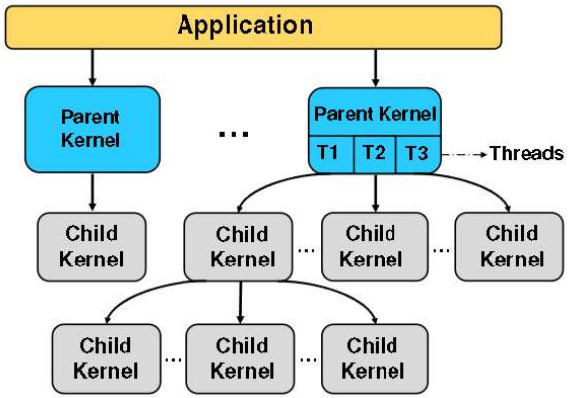


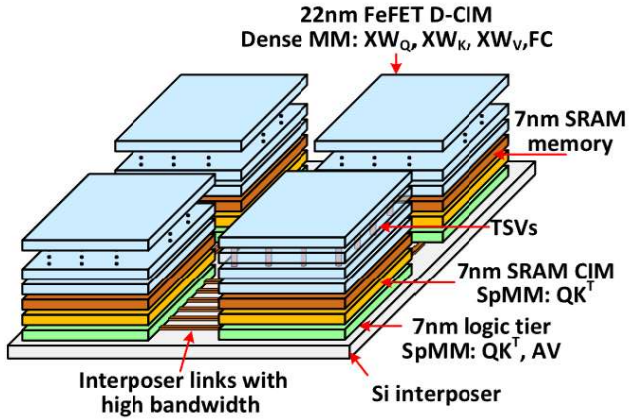
Growing numbers of dynamic workloads can benefit from run-time, workload-aware acceleration.

Img Sources:
[1] Adaptive Parallel Computation with CUDA Dynamic Parallelism | NVIDIA Technical Blog
[2] From Dropout to Winner — Advancing Neural Network Sparsity | by Mr. For Example | Medium

Motivation: Run-time Dynamic Workloads

- **How do we leverage these dynamic properties?**
  - GPU: CUDA dynamic parallelism
  - CPU: task-based parallelism
  - Heterogeneous Integration: Sparse and Dense MM



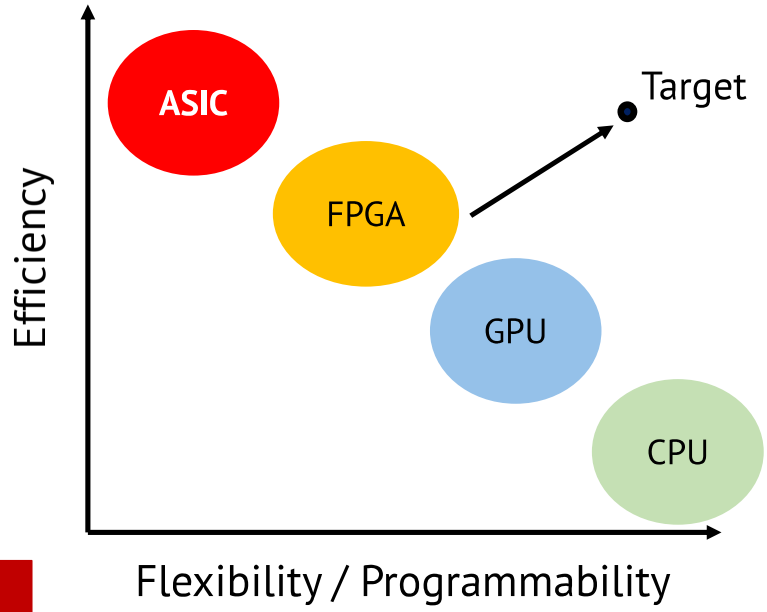


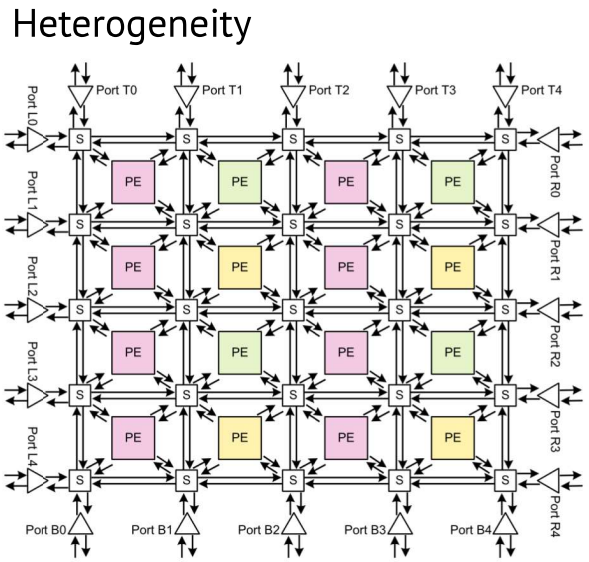
Dynamic scheduling and Heterogeneous integration are utilized to accelerate dynamic workloads

Img Sources:
[1] Tang, Xulong, et al. "Controlled kernel launch for dynamic parallelism in GPUs." 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2017.
[2] Luo, Yandong, and Shimeng Yu. "H3d-transformer: A heterogeneous 3d (h3d) computing platform for transformer model acceleration on edge devices." ACM Transactions on Design Automation of Electronic Systems 29.3 (2024): 1-19.

Motivation: Reconfigurable System

- **Reconfigurable System**
  - Run-time reconfigurability to support dynamic scheduling
  - ASIC-like performance and power efficiency
  - Global/Local Heterogeneity to facilitate diverse functionalities





Reconfigurable System has great potential to accelerate dynamic workloads

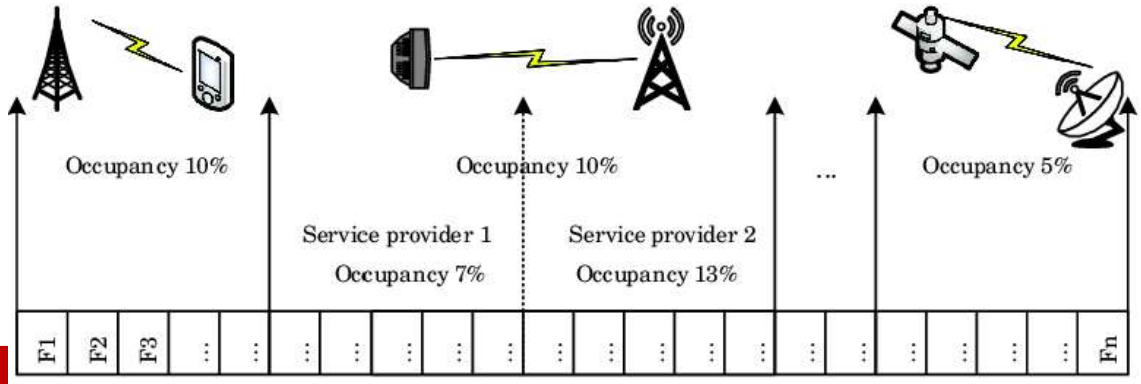
Agenda

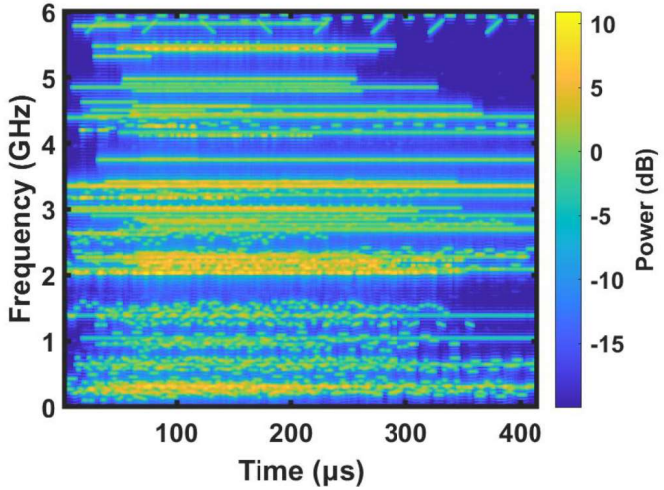
- Motivation: Reconfigurable Systems and Dynamic Workloads
- **Preliminary Work**
  - An Overview for Real-Time Spectrum Sensing in Modern RF Autonomy Systems*
  - Run-time Dynamic Scheduler and Design-Time Optimization on CGRAs
- Proposed Work
  - A Heterogeneous Acceleration Platform for Hybrid LLM Service
  - Resource Management of Heterogeneous Antenna Array
- Timeline and List of Publications
- Conclusions

*__Jiahao Lin__, et al. "An Overview of Challenges and Requirements for Real-Time Spectrum Sensing in Modern RF Autonomy Systems." IEEE Design & Test (2025).

Real-time Wideband Spectrum Sensing (SS)

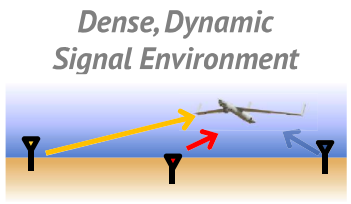
- **Radio cognition is fundamental for wireless communication**
  - A growing demands: Internet-of-Things (IoT), live stream, and unmanned vehicles
- **The real-time wideband SS is the basic component of radio cognition**
  - To help fully utilize the spectrum resources and enables rapid interaction
- **The wideband SS is highly dynamic at run-time, because of:**
  - Unknown spectrum environment (combination of sparse and active bands)
  - Different sample rates of sub-bands
  - Diverse transmitter algorithm

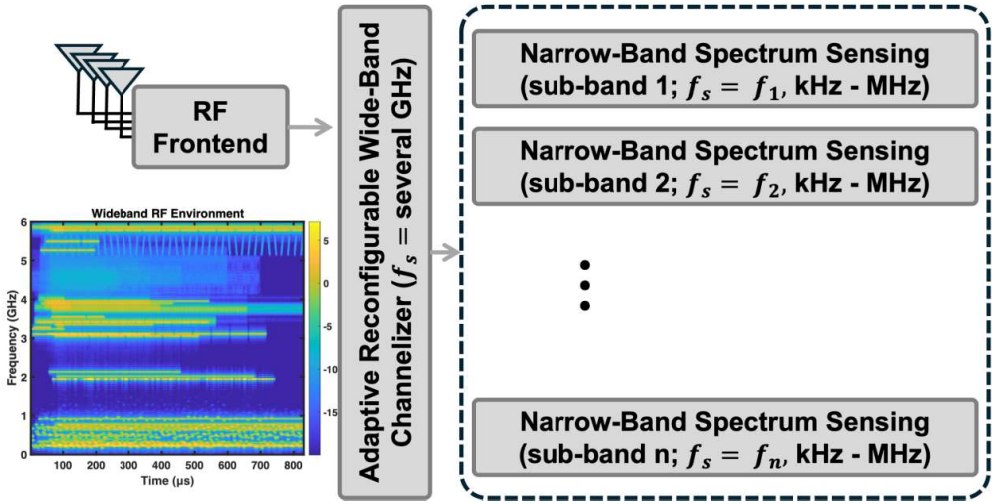




New Challenges

- **Congested frequency bands usage**
  - Require analysis on wider band and more analysis to distinguish
- **Increasing complexity of software/AI-defined transmitters**
  - More complicated and diverse analysis algorithms
- **Highly dynamic and diverse wideband spectrum environments**





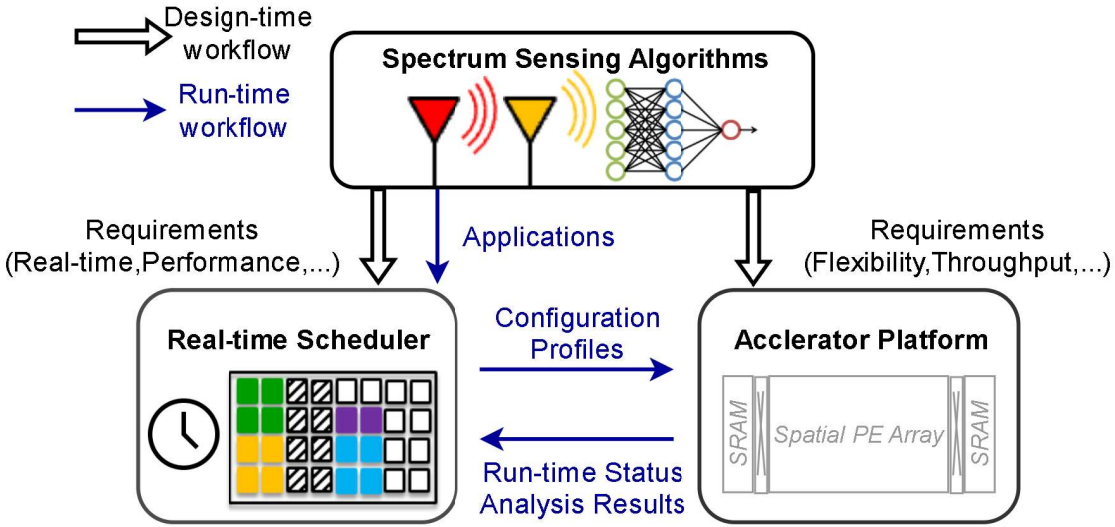
New challenges of SS demand more computation, flexibility and sensor capabilities

System Requirements

- **Reconfigurable hardware needs fast switching at waveform timescales:**
  - Adapt its computation kernels at the speed of changing waveforms
- **Real-time adaptation for resource management:**
  - Dynamically schedule processing resources based on the sensed signal environment
- **High throughput:**
  - Handle dense and wide-band spectrum
- **Broad application scope:**
  - Diverse modulation types and transmitters

Solution: The System Composition

- **What exactly does the system require?**
  - A high-level overview: radio frequency sensors, computation platform and real-time resource management module
  - Sensors collect samples, accelerator do analysis, and resource management allocate computation kernels and configure antenna arrays



Sensors+Analysis Requirements → Runtime resource management +HW/SW Design

Investigation on Existing Approaches

Accelerator Platforms

Name	Flexibility	Reconfig. Speed	Throughput Optimization
DAP [55]	PE	less than $1\mu s$	systolic connection
FluxSPU [56]	PE	Not Specified	push/pop mechanism
UDSP [57]	PE	Not Specified	multi-layer switchbox
TIA [58]	Tile	Not Specified	triggered instructions
Arena [59]	Group	8 cycles ideally	ring network
FlexArch [60]	Tile	Not Specified	continuation passing
Amber [61]	DPR	in ms scale	streaming and reused buffer

Scheduling Approaches

Name	Programmability	Real-time	Performance Enhancement
Diagonal [63]	C/C++	No	fast compilation
MTDE [19]	C/C++	Yes	bypass reconfiguration
Work Steal [60]	C/C++	Yes	greedy scheduling
SparseAdapt [20]	C/C++	Yes	phase change detect
PPA [64]	C/C++	Yes	virtualized scheduling
DML [65]	C/C++	No	ILP
CEDR [66]	DAG Model	Yes	HEFT scheduler
IRIS [67]	API Model	Yes	locality aware scheduler
SURF [68]	API Model	Yes	adaptive task mapping

No single method meets all requirements of next-gen spectrum sensing.

Holistic approaches need to be developed.

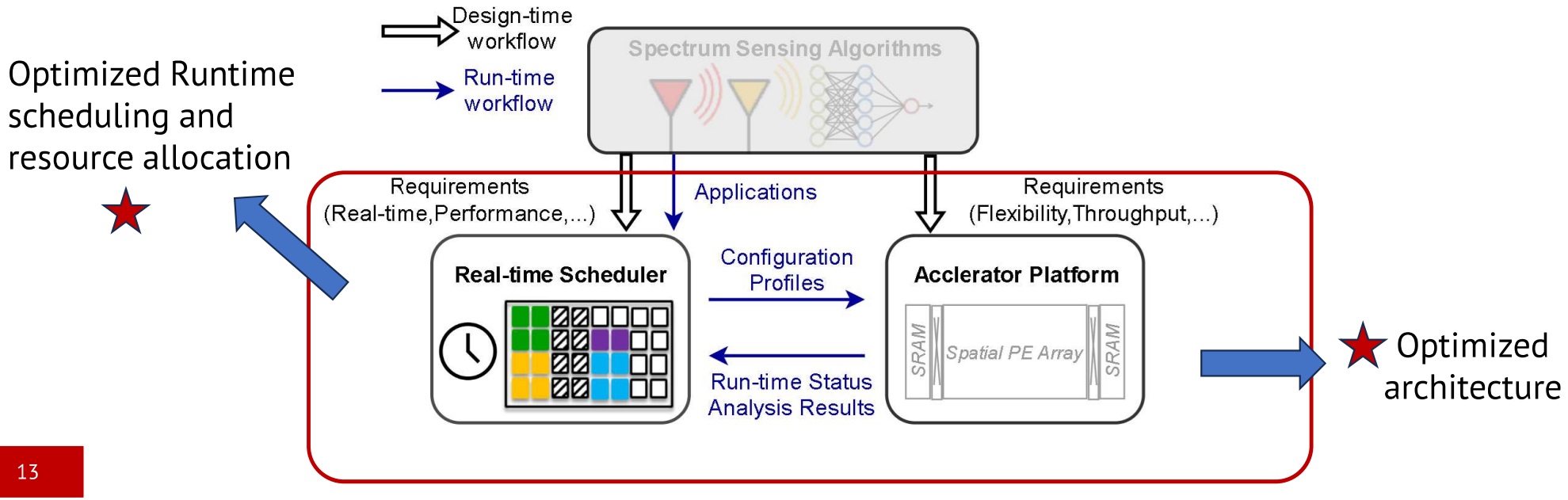
Agenda

- **Motivation: Reconfigurable Systems and Dynamic Workloads**
- **Preliminary Work**
  - Dynamic Systems for Real-Time Spectrum Sensing in Modern RF Autonomy
  - Run-time Dynamic Scheduler and Design-Time Optimization on CGRAs*
- **Proposed Work**
  - A Heterogeneous Acceleration Platform for Hybrid LLM Service
  - Resource Management of Heterogeneous Antenna Array
- **Timeline and List of Publications**
- **Conclusions**

***Jiahao Lin,** et al, "CADAS: Communication-Aware Dynamic Scheduler on CGRAs for Large-volume and Real-time Processing" ACM Transactions on Embedded Computing Systems (2025). (2nd round of revision)

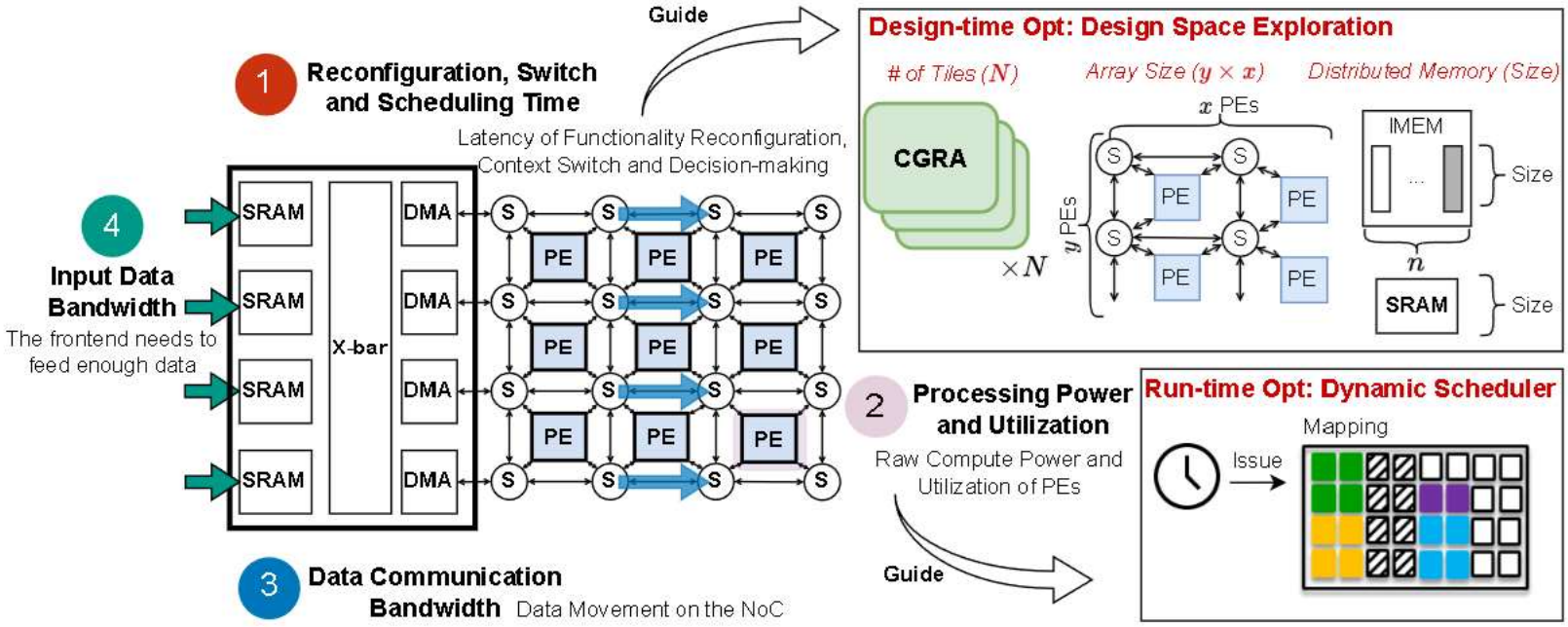
Design-time and Run-time Optimization

- **For a holistic solution, it is crucial to**
  - Give real-time decision on how to schedule computation kernels on hardware
  - Make design-time optimization on the hardware architecture to adapt to the characteristics of workloads



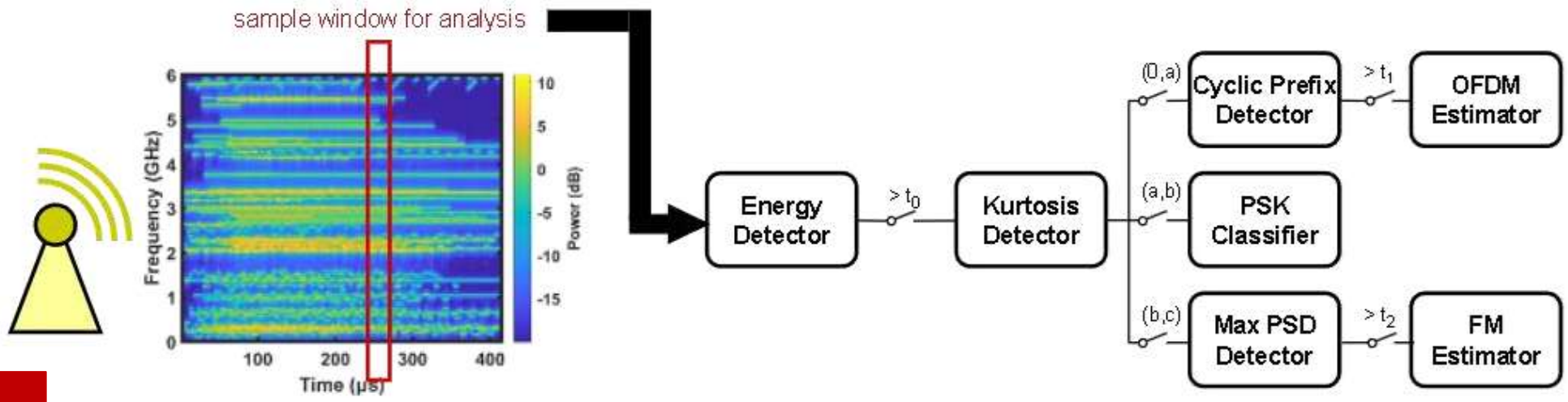
Design-time and Run-time Optimization

- **CADAS: Communication-Aware Dynamic Scheduler on CGRAs for Large-volume and Real-time Processing**
  - Design-time: Explore the global architectural features (e.g. scale, PE/memory organization)
  - Run-time: A dynamic scheduler places the incoming tasks to the available PEs

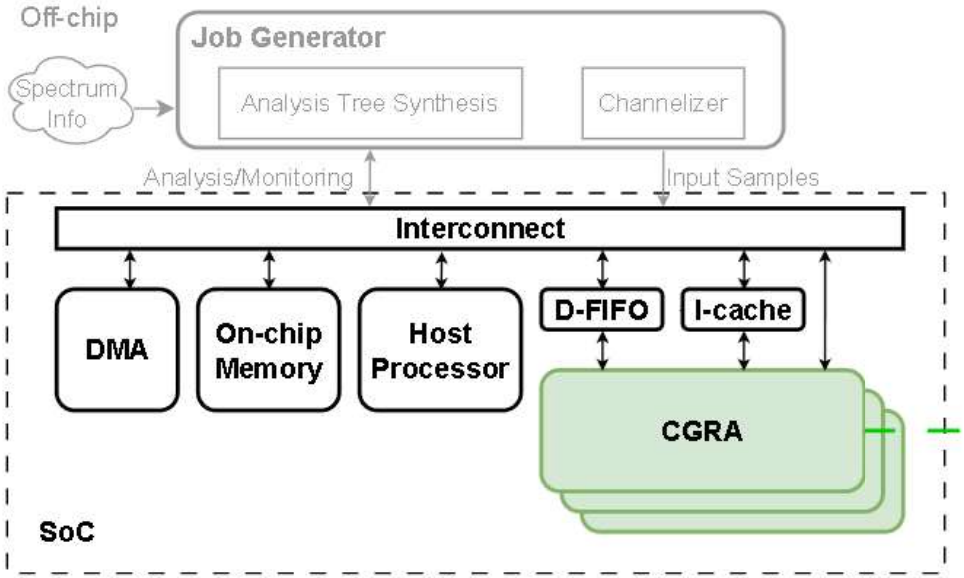


Background: Workloads Description

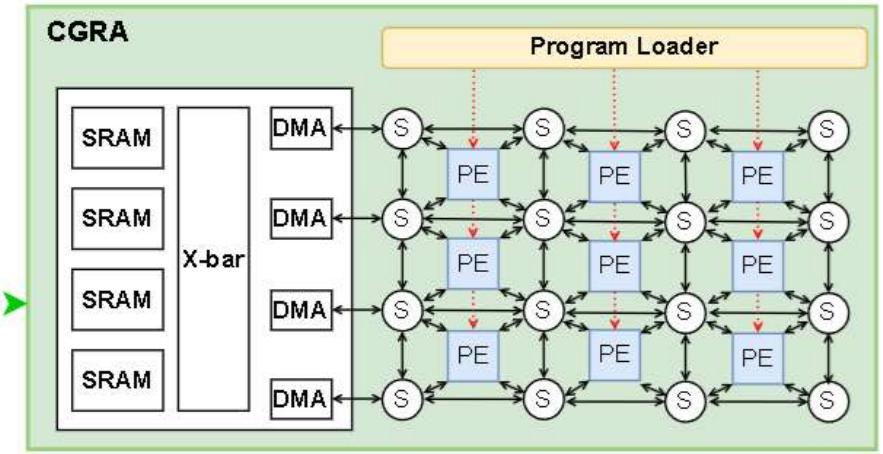
- **A spectrum sensing's analysis example is performed here**
  - Each node corresponds to a processing kernel, and edges capture the conditional dependencies between kernels
  - Signals arrive dynamically in a window-based fashion
  - Generally, interesting samples require deeper processing in the tree which involves more complicated processing kernels.



Background: Target Hardware Architecture



(a) Hardware System Overview



(b) CGRA 3x3 Array Architecture

A sample SoC that integrates the real-time scheduler into the host processor, and uses CGRAs as an accelerator platform

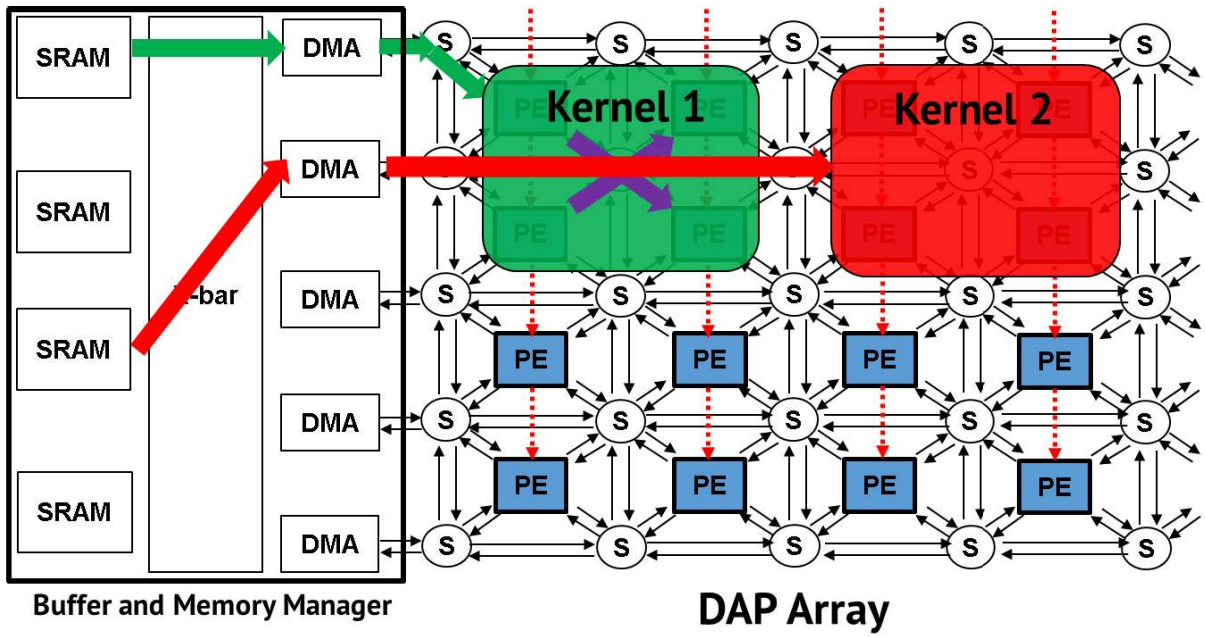
An example CGRA architecture designed for SS

- We explore the global architecture (e.g., size), dynamic placement, and routing

Background: Kernel Placement

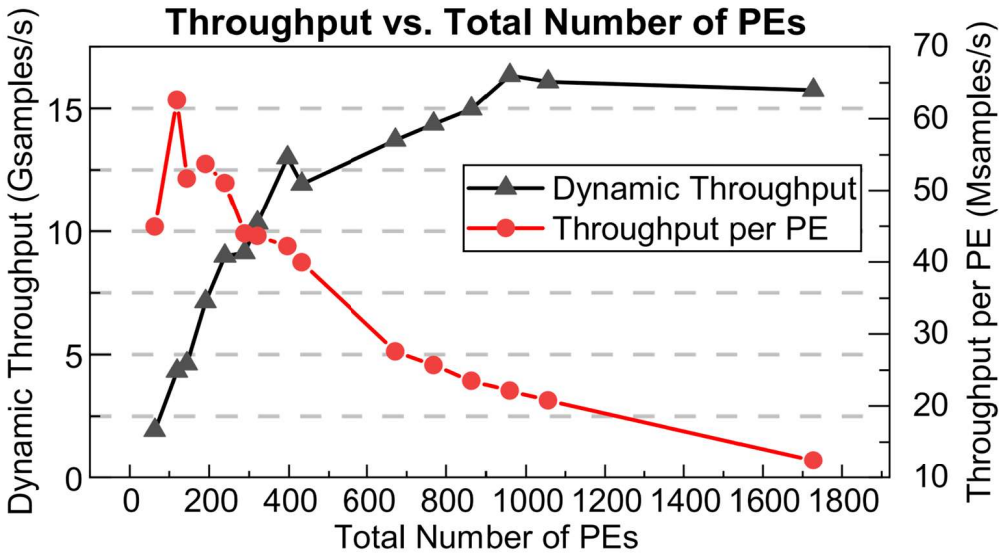
- **Map pre-compiled computation kernel onto hardware** (available due to domain knowledge)
  - Input samples come from either from DMA and SRAM or the output of previous kernels
  - Each row has an DMA engine and DMA engines are connected to SRAMs using a X-bar

- **We place the kernels to the available PEs and assign the communication resources**
  - Set the routing path
  - Set the switches & X-bar



Design-time Optimization

- **Experiment**
  - Sweep the number of PEs in a single (monolithic) PE-Array
  - Place, route, and simulate the resulting system under different SS scenarios

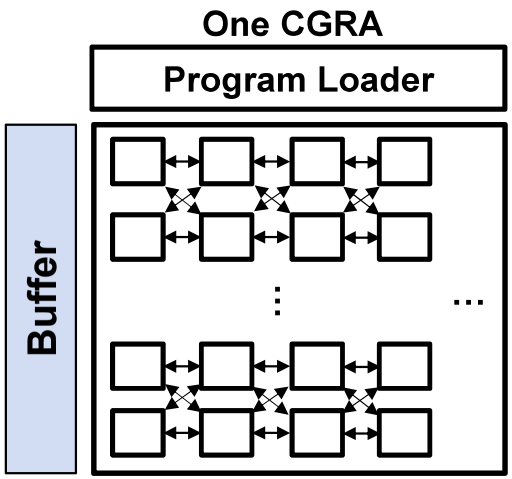


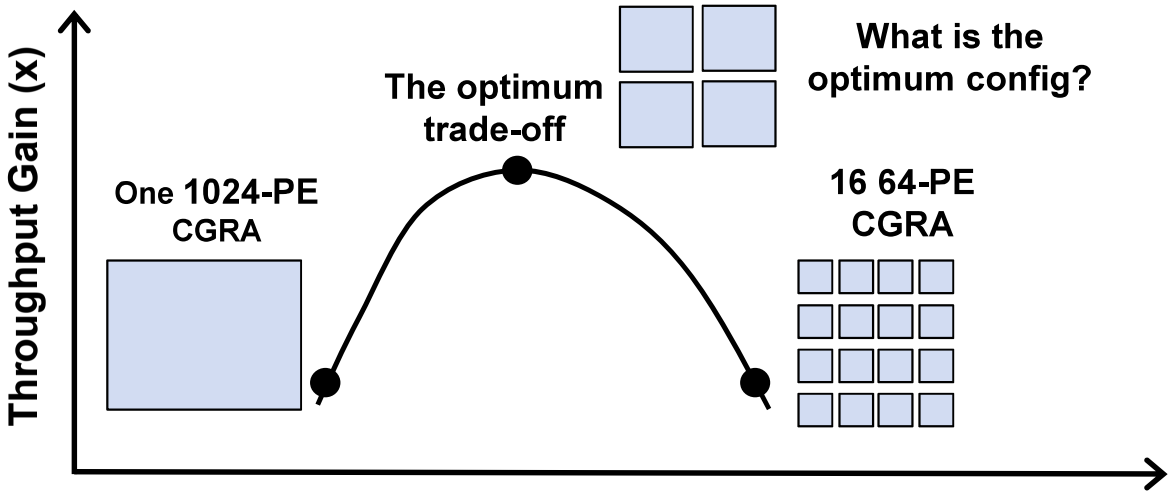
- **Observations**
  - Total throughput saturates despite adding more PEs
  - Throughput per PE reaches a peak and diminishes (due to comm/memory bottlenecks)

Analyzing the scalability and performance as a function of system size are critical

Design-time Optimization

- **Analyze architectural decisions**





Drawbacks:

- Data movement overhead between PEs
- Complexity of placement and scheduling algorithms
- One program launcher shared by too many PEs
- Limited data bandwidth (one row shared by many PEs)

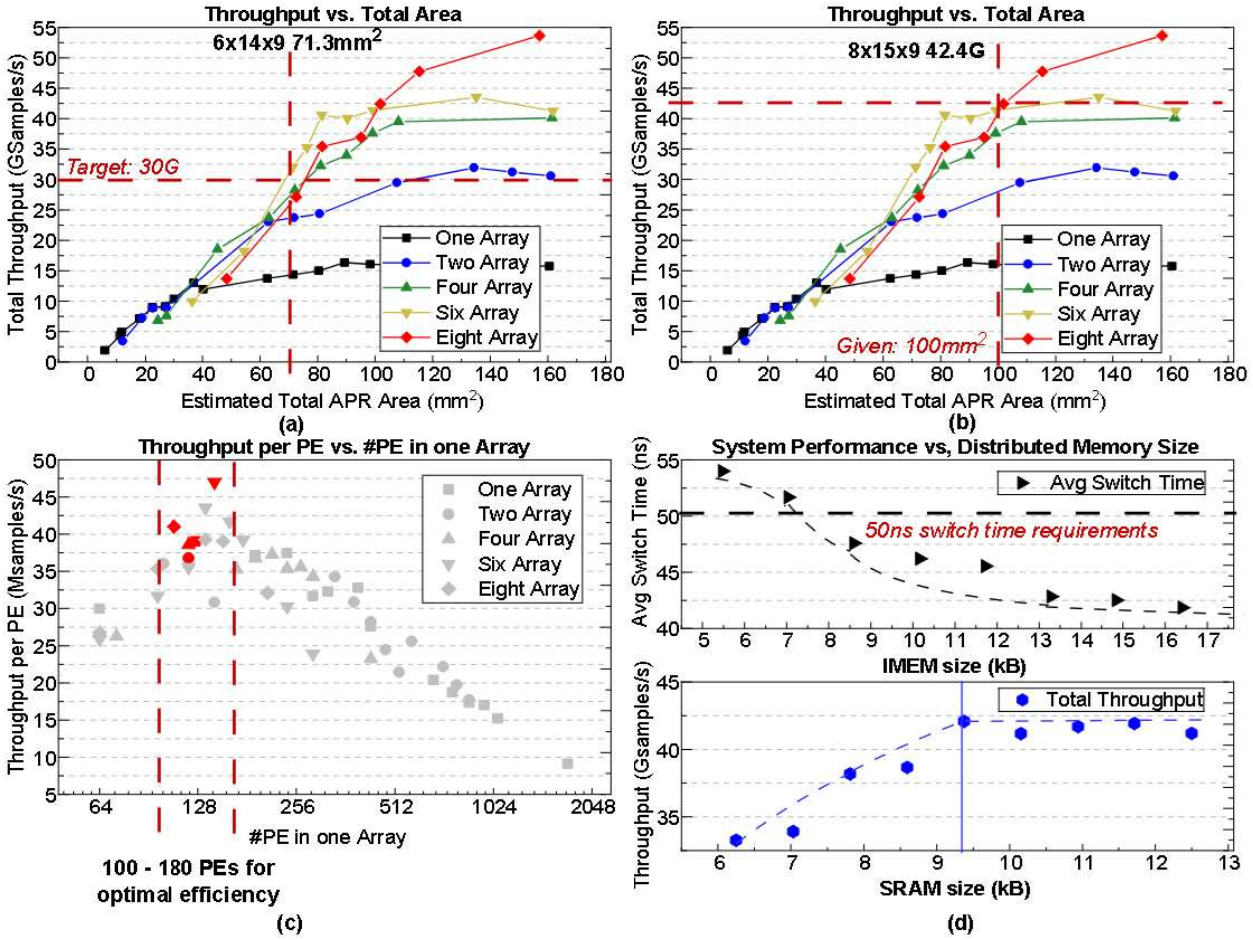
Drawbacks:

- Low utilization (fragmentation)
- Inter-PE array communication
- Synchronization overhead between individual program loader

Design-Time Optimization

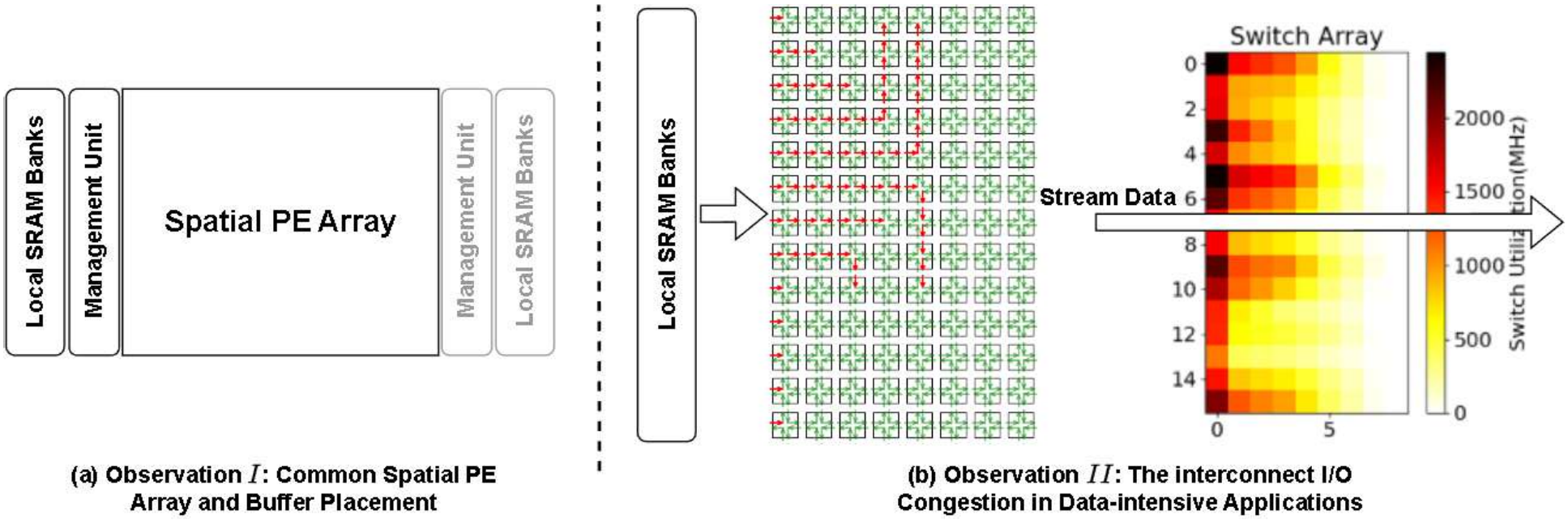
- Throughput vs. Area
- Throughput per PE
- Distributed memory's impact

The DSE finds optimized hardware configuration for the given benchmarks



Run-time Optimization

- **Observations: During the runtime, the bottleneck can be**
  - Kernel switch time and NoC communication bandwidth

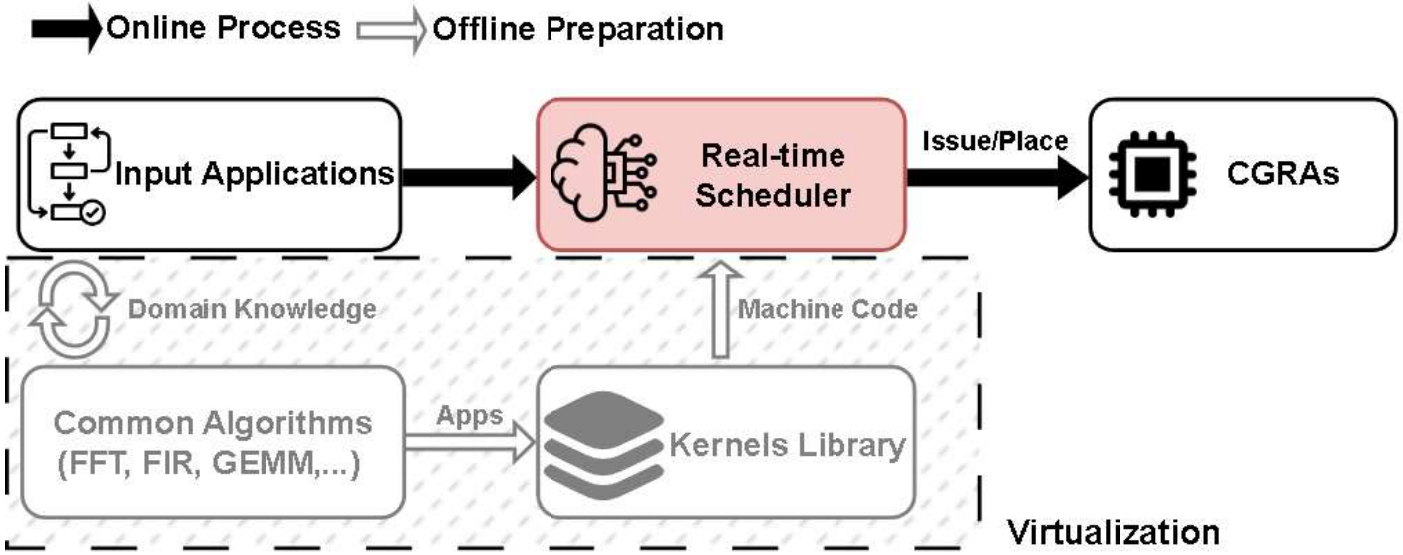


(a) Observation I: Common Spatial PE Array and Buffer Placement

(b) Observation II: The interconnect I/O Congestion in Data-intensive Applications

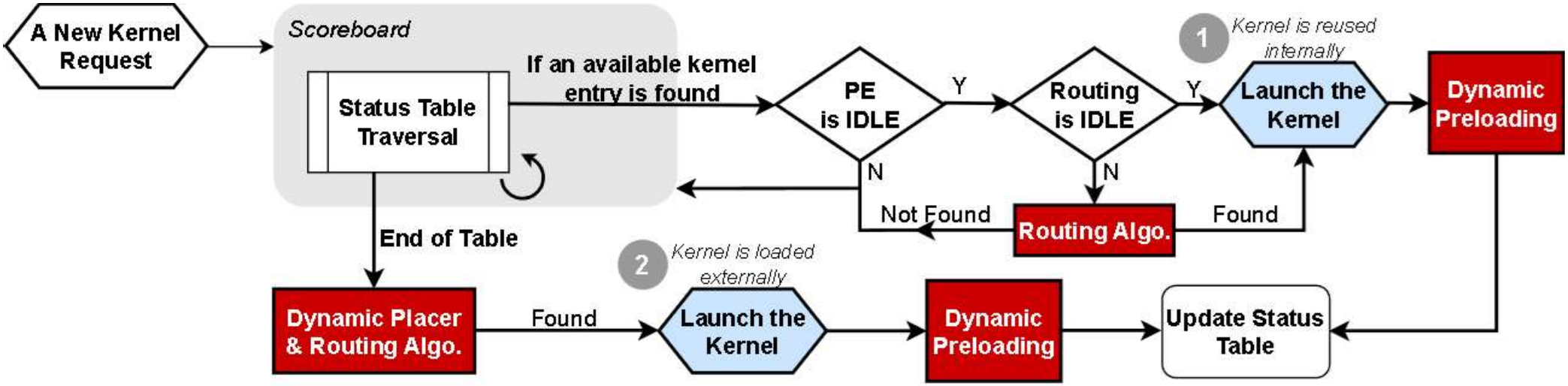
Real-Time Scheduler Goals

- Find the kernel placement and routing configurations that minimize the kernel switch time and maximize the bandwidth utilization
- Minimize the decision-making time for placement and routing configurations



Run-time Optimization

- **Kernel placement and routing path scheduling workflow**
  - Scoreboard: If a matching kernel from another job is under-utilized and its instructions are already loaded into the PEs, the scheduler reuses it to avoid context switch time
  - Dynamic placer: If no suitable kernel is found in the scoreboard, the OCS triggers dynamic placement and routing to allocate a new location



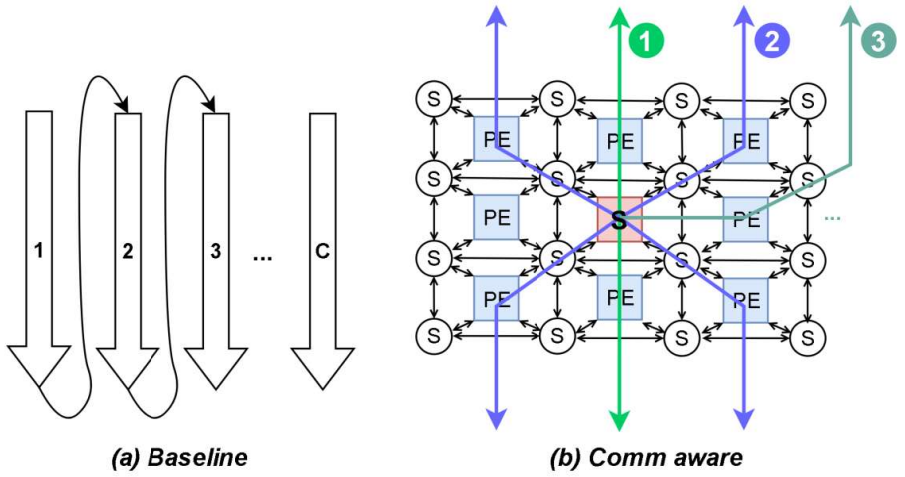
Run-time Optimization

- **We model the bandwidth utilization with kernels' placement and routing path**
  - Cost function as a function of the communication latency between data source & destination
  - Minimizing the cost function maximizes the communication bandwidth utilization
  - This cost function drives both the dynamic placer and scoreboard traversal, enabling the system to select either a new placement or a reused kernel with the lowest cost

$$B_{\mathrm{IO}}(k,r) = \frac{D_{\mathrm{sample}}}{t_{\mathrm{data}}(k) \cdot N_{\mathrm{links}}(r) \cdot L_w f}$$

$$\mathcal{P}_k^{\star} = \underset{\mathcal{P}_k \in \mathcal{F}_p(k)}{\arg\max}\ B_{\mathrm{IO}}(k,r) = \underset{\mathcal{P}_k \in \mathcal{F}_p(k)}{\arg\min}\ t_{\mathrm{data}}(k),$$

$$Cost(k) = |L_k(x) - L_{\mathrm{src}}(x)|$$

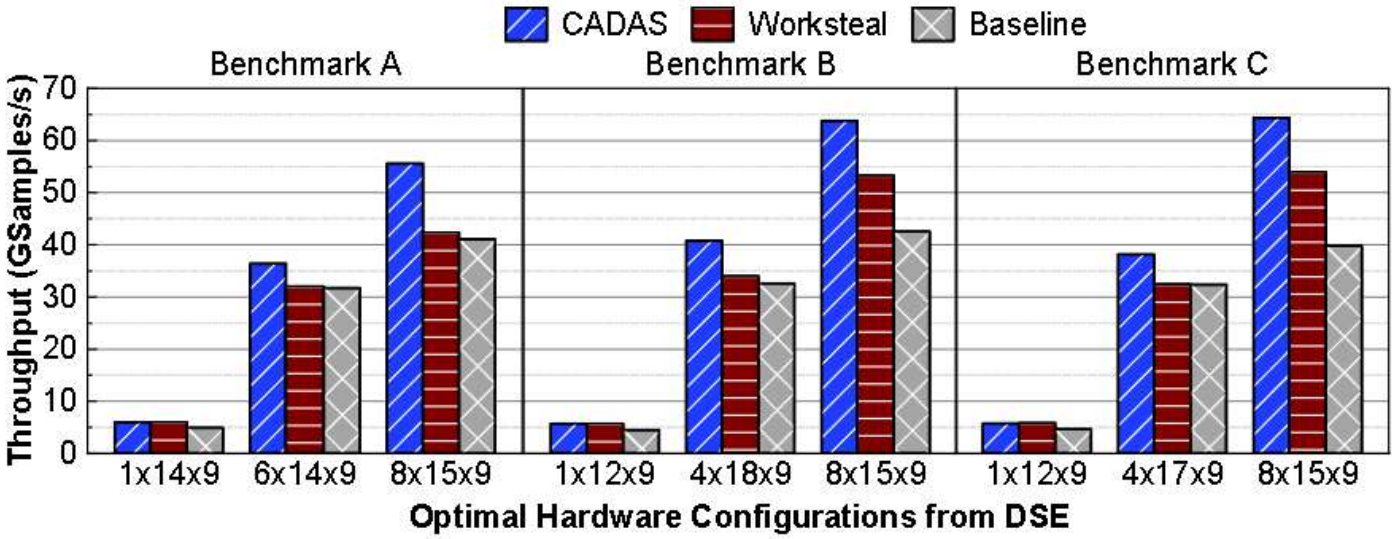


(a) Baseline *(b) Comm aware*

Run-time Optimization – Ablation Study

- **HAMMER (Baseline) [1]:** Synthesizes a domain-specific set of pre-characterized kernels offline, and uses a hash table to identify the kernel id and configurations
- **Work-steal[2]:** Preloads successor kernels into unutilized PEs, so that the scheduler can use scoreboard to utilize them

Up to 1.6x and 1.3x gain potention over the adapted baseline and work-steal methods



[1] Qilin Si and Benjamin Carrion Schaefer. 2025. HAMMER: Hardware-aware Runtime Program Execution Accelerationthrough runtime reconfigurable CGRAs. In Proceedings of the 30th Asia and South Pacific Design Automation Conference.272–278

[2] Tao Chen, Shreesha Srinath, Christopher Batten, and G Edward Suh. 2018. An architectural framework for acceleratingdynamic parallel algorithms on reconfigurable hardware. In 2018 51st Annual IEEE/ACM International Symposium onMicroarchitecture (MICRO). IEEE, 55–67.

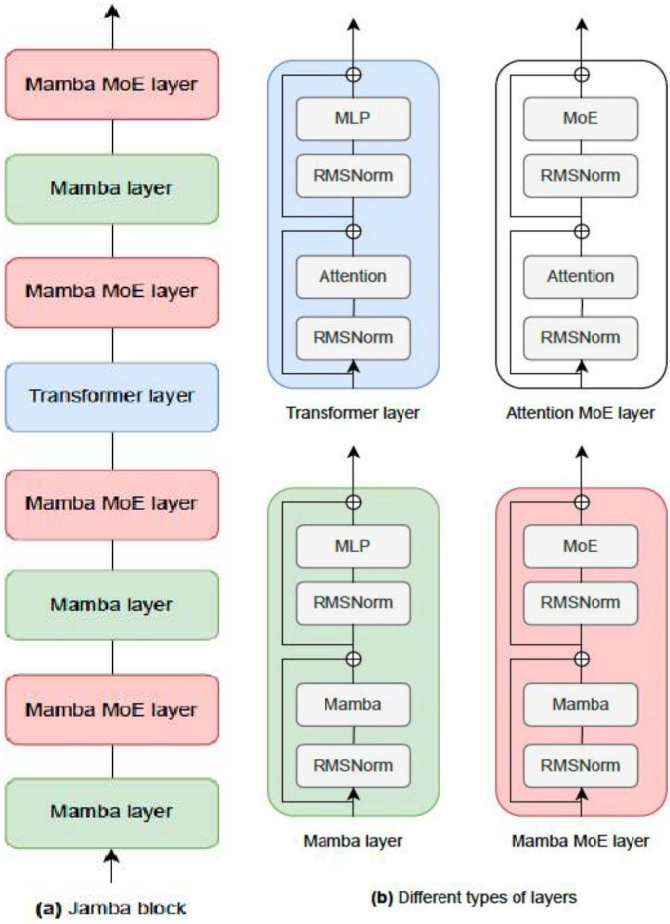
Agenda

- **Motivation: Reconfigurable Systems and Dynamic Workloads**
- **Preliminary Work**
  - Dynamic Systems for Real-Time Spectrum Sensing in Modern RF Autonomy
  - Run-time Dynamic Scheduler and Design-Time Optimization on CGRAs
- **Proposed Work**
  - A Heterogeneous Acceleration Platform for Hybrid LLM Service
  - Resource Management of Heterogeneous Antenna Array
- **Timeline and List of Publications**
- **Conclusions**

Motivation, Overview and Background

- **LLM types: Attention or Recurrency**
  - Transformer vs. SSM/RNN
- **Hybrid Solution**
  - Trade-off between Efficiency (Mem & throughput) and Capability (Score)
  - Transformer: Good Cap, Large Mem footprint(KV cache) and low throughput (Att comp)
  - Mamba: Comp efficient, high throughput, ~Cap

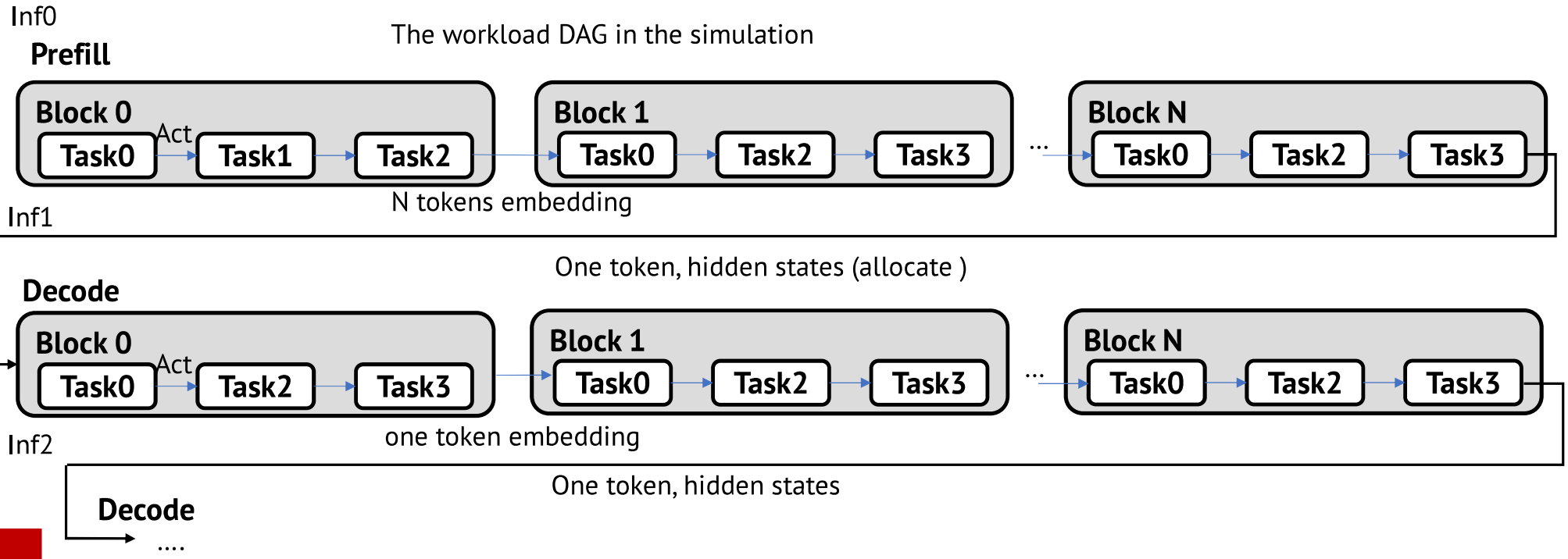
Growing trend of LLMs with a hybrid architecture[1-4] from industry



[1] Blakeman, Aaron, et al. "Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models." arXiv preprint arXiv:2504.03624 (2025).
[2] Lieber, Opher, et al. "Jamba: A hybrid transformer-mamba language model." *arXiv preprint arXiv:2403.19887* (2024).
[3] https://github.com/foundation-model-stack/bamba?tab=readme-ov-file
[4] Team, Tencent Hunyuan, et al. "Hunyuan-TurboS: Advancing Large Language Models through Mamba-Transformer Synergy and Adaptive Chain-of-Thought." arXiv preprint arXiv:2505.15431 (2025).

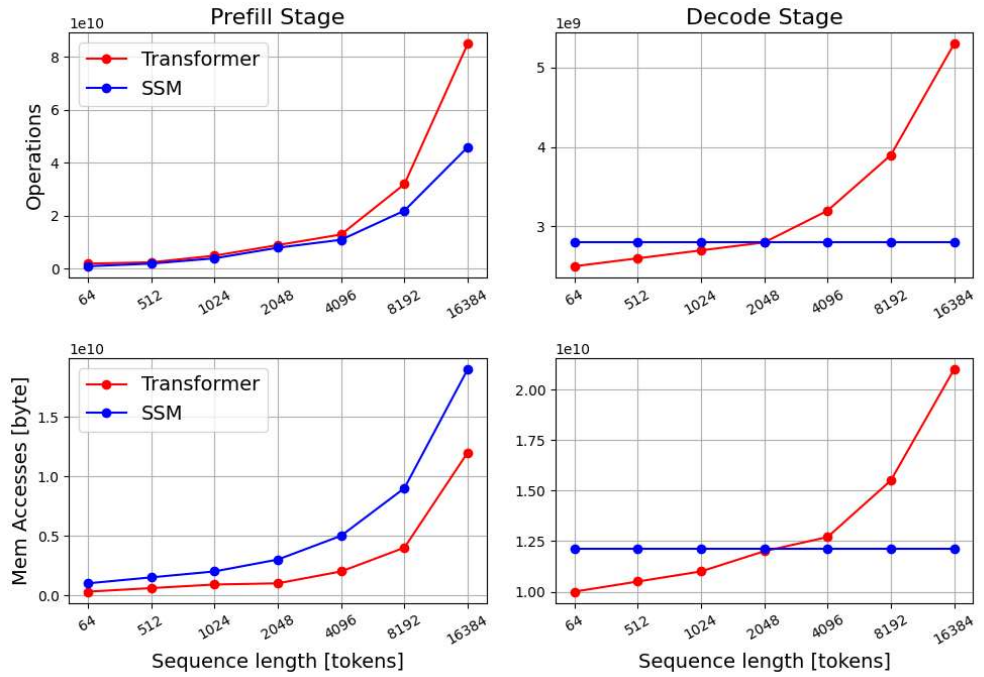
Motivation, Overview and Background (con't)

- **LLM services are runtime dynamic workloads**
  - Prefilling and decoding stages have different computation characteristics
  - The number of decoding stages is unknown when the request comes



Preliminary Profiling Results

- Profiles obtained through running OPT-2.7B and Mamba-2.8B on an A100 GPU
- Attention, Mamba, prefill and decoding phases show different characteristics



Analytic estimate of computation complexity

Phase	Attention	SSM
Prefill	$O(L^2xED)$	$O(LxEDxN)$
Decode	$O(LxED)$	$O(NxED)$

Mamba and Attention decoders show different computation and memory requirements

Profile Sources:
[1] Geens, Robin, Arne Symons, and Marian Verhelst. "Fine-Grained Fusion: The Missing Piece in Area-Efficient State Space Model Acceleration." arXiv preprint arXiv:2504.17333 (2025).

Challenges and Proposal

Challenges to accelerate hybrid LLMs

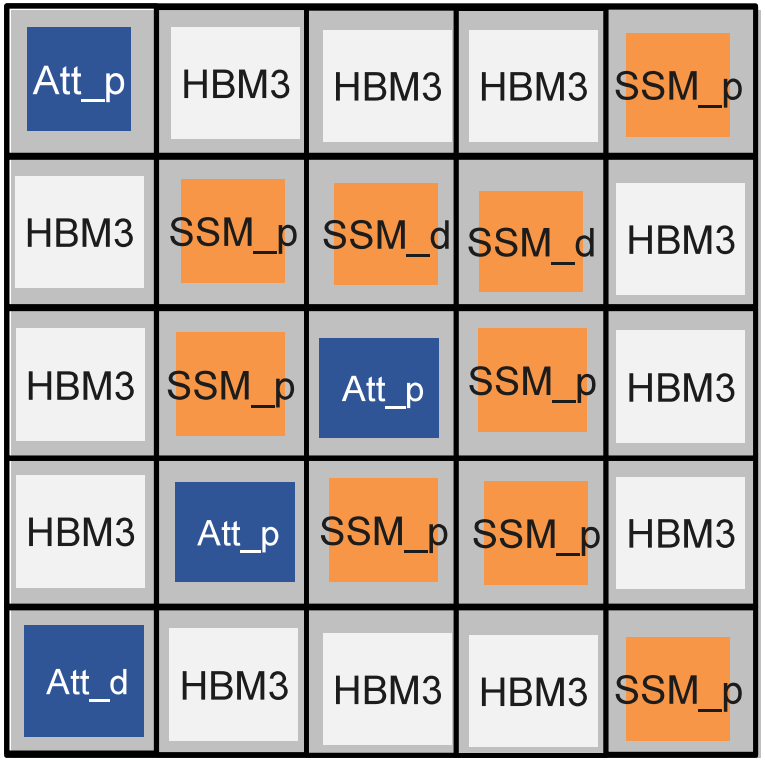
- Diverse operations across Transformer and Mamba blocks (e.g., non-linear, element-wise)
- Dynamic workload patterns in LLM services (prefilling vs. decoding)
- Large-scale system demands: memory, bandwidth, and compute

Proposal

- Chiplet-based DSE framework with heterogeneous accelerators
- Bottleneck analysis: memory bandwidth, capacity, and compute under varied chiplet placement/organization and workloads
- Runtime scheduler for request dispatch, memory management, and kernel mapping

A Demonstration of Acceleration Platform

- **Computation chiplets:**
  - Att_* and SSM_* are transformer and mamba accelerators
  - Computation and communication resource allocations adapt to prefilling or decoding phases (_p and _d)
  - Analytic models (performance, power, area) are set up based on the profiles of existing accelerators
- **Memory chiplets**
  - We use Ramulator3 to model the behaviors of HB3
- **Network on Interposer**
  - We follow the assumption in UCIE3 (e.g. link bandwidth, frequency)

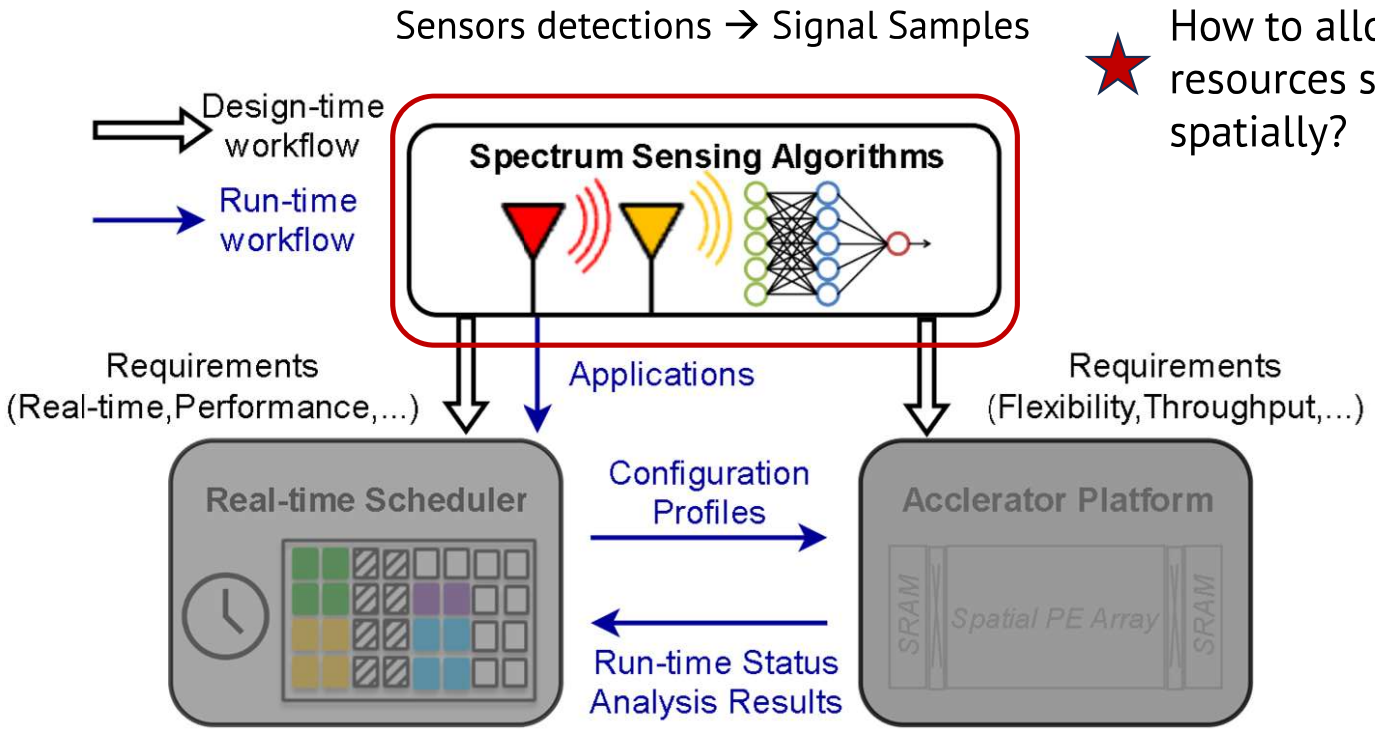


Agenda

- **Motivation: Reconfigurable Systems and Dynamic Workloads**
- **Preliminary Work**
  - Dynamic Systems for Real-Time Spectrum Sensing in Modern RF Autonomy
  - Run-time Dynamic Scheduler and Design-Time Optimization on CGRAs
- **Proposed Work**
  - A Heterogeneous Acceleration Platform for Hybrid LLM Service
  - Resource Management of Heterogeneous Antenna Array
- **Timeline and List of Publications**
- **Conclusions**

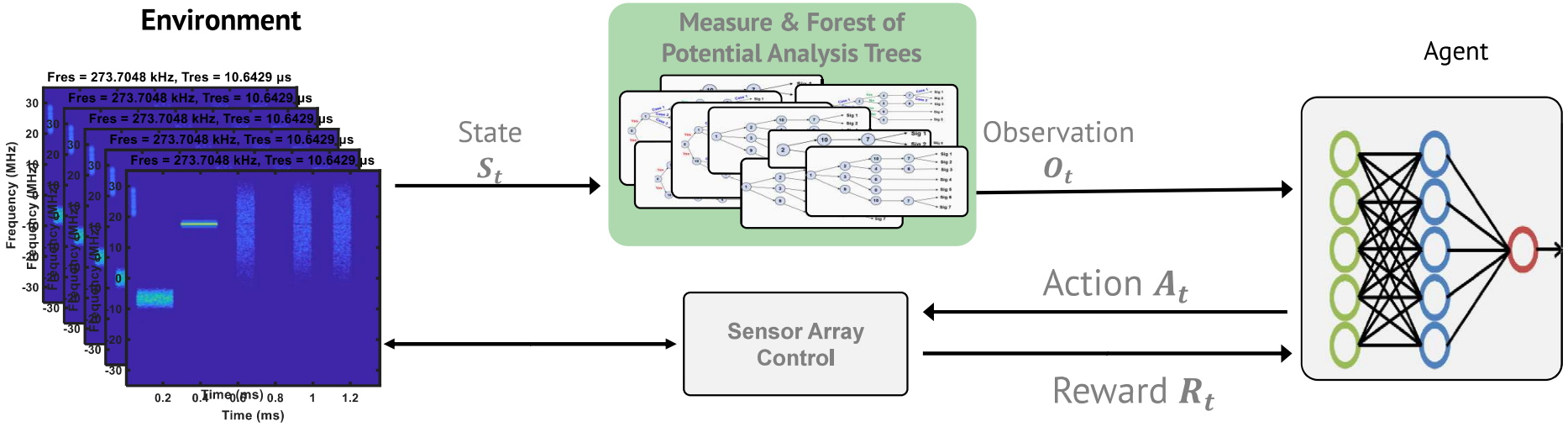
Resource Management for Antenna Arrays

- **Reminder**



Resource Management for Antenna Arrays

- The limited receive channel and antenna resources need to be managed
- We propose an intelligent resource manager (IRM) for this purpose



Resource Management for Antenna Arrays

1. **Find probabilistic upper bounds on**
   - The number of emitters that can be detected/classified
   - The score (e.g., a weighted sum of detected signals, where weight shows importance
2. **Utilize learning-based (e.g. RL, MLP, …) to predict distributions parameters of emitters (e.g. location, center frequency, …)**
3. **Use these predictions to model the detection score function**
4. **Find the actions (input) that maximize the detection score**

An intuitive whitebox process