A Hierarchical Dataflow-Driven Heterogeneous Architecture for Wireless Baseband Processing

Limin Jiang, Yi Shi, Yintao Liu, Qingyu Deng, Siyi Xu, Yihao Shen, Fangfang Ye, Shan Cao, and Zhiyuan Jiang

School of Communication and Information Engineering,
Shanghai University, China

Outline

ACE-Lab

Background

- 5G expansion drives demand for energy-efficient, open-source hardware to replace proprietary solutions.

- Implementing hardware for data-intensive wireless baseband processing (WBP) poses major challenges.

OPEN AIR INTERFACE

Open5GS

SRS SOFTWARE RADIO SYSTEMS

OAIBOX™

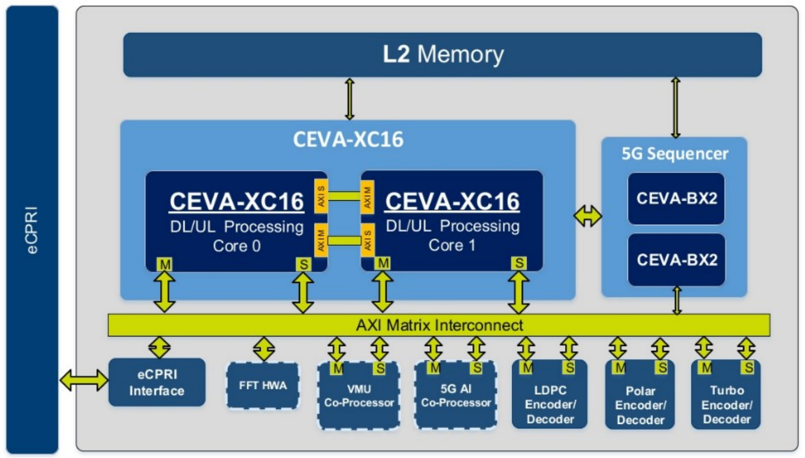
free5GC

Background

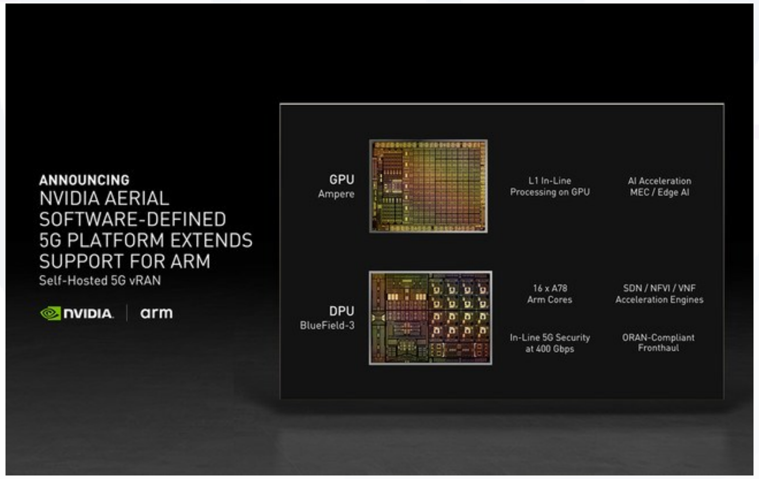
ACE-Lab

WBP: How?

- ☐ DSP:
  - ✓VLIW boosts ILP
  - ✓High control overhead limits scalability

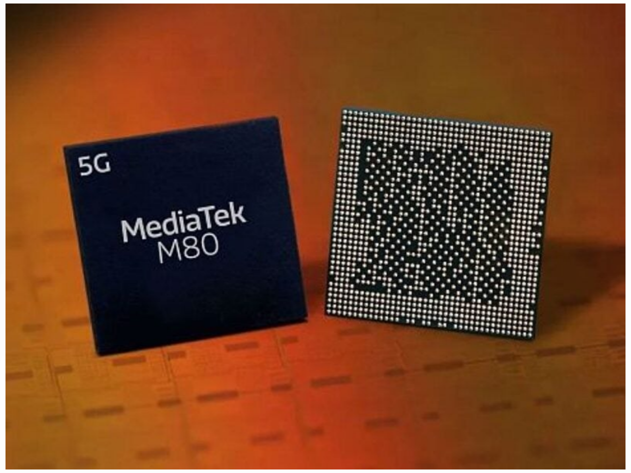


- ☐ X86 Server/GPGPU:
  - ✓ Massive computing capability
  - ✓High energy consumption


ANNOUNCING
NVIDIA AERIAL SOFTWARE-DEFINED 5G PLATFORM EXTENDS SUPPORT FOR ARM
Self-Hosted 5G vRAN

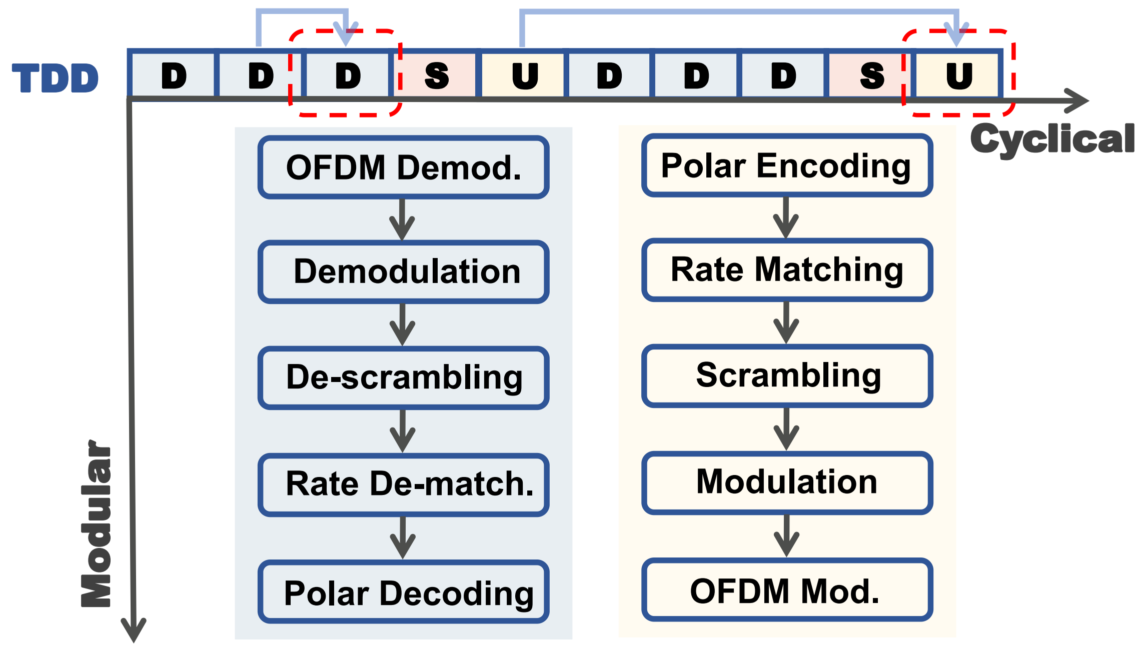
NVIDIA | arm
GPU
Ampere
L1 In-Line Processing on GPU
AI Acceleration MEC / Edge AI
DPU
BlueField-3
16 x A78 Arm Cores
SDN / NFVI / VNF Acceleration Engines
In-Line 5G Security at 400 Gbps
ORAN-Compliant Fronthaul

- ☐ ASIC:
  - ✓Best PPA
  - ✓Long time to market



Motivation

- Two characteristics of WBP:
  - **Modular:** TDD frame structure separates uplink and downlink into **data-independent**, **successive** modules.
  - **Cyclical:** Signal generation or decoding based on communication protocols is on a *subframe* (**periodic**) basis.



WBP is decoupled and predictable.

Contribution

- A **cache-free manycore** architecture is proposed to increase energy and area efficiency without compromising performance due to the predictable data processing nature of WBP.
- We develop a ***pack-and-ship*** data dispatch system to enable the tiles to operate in a **bundled access and execution** style, which can drastically reduce the cost of data movement.
- A **hierarchical dataflow** task scheduling scheme is designed and two strategies, namely multi-threading and lazy-deletion, are proposed to fully utilize the hardware resources.

Related Works

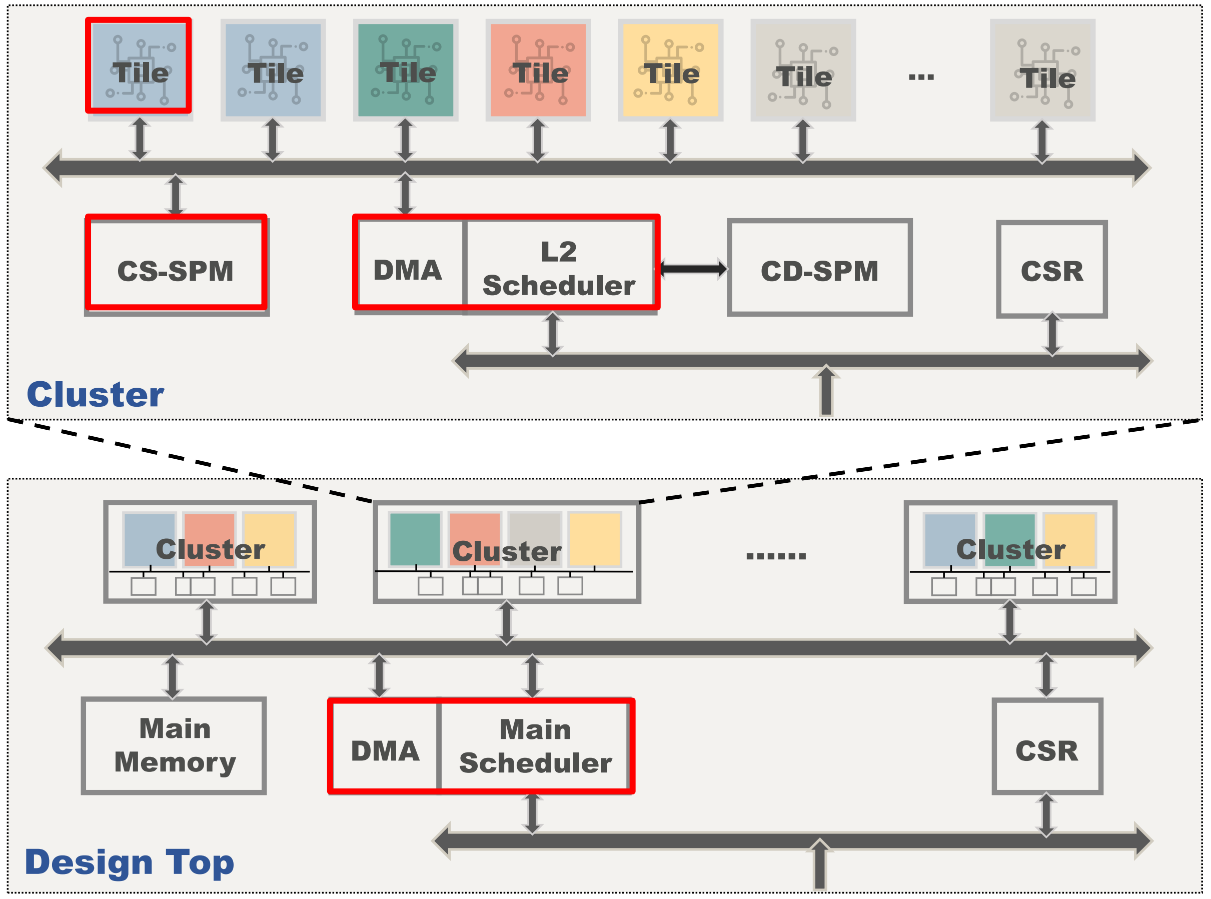
■ Various works have been presented in academia seeking a way towards manycore parallel computing for WBP.

	Work	Core Heterogeneity	Scalability	DLP	TLP	HW/SW Co-design
GPPs	Sora	☹	☺	☺	☺	☹
Sys-level Analyses	TeraPool	☹	☺	☺	-	☺
	SPECTRUM	☹	☺	☹	☺	☹
NoCs	MACRON	☺	☺	☺	-	☺
	MAGALI	☺	☺	☹	-	☹
ASIP	DXT501	☺	☹	☺	☺	☺
	Ours	☺	☺	☺	☺	☺

System Design: Architecture

ACE-Lab

- **Tile:** RV32IM core with customized vector extension & local scratchpad memory.
- **L2 DMA:** Orchestrating Tiles via a scalar scheduler.
- **CS-SPM:** Swap space for cluster.
- **Main scheduler:** Managing high-level scheduling; Directing the main DMA engine to transfer data between main memory and the clusters.



System Design: Pack-n-Ship

ACE-Lab

■ NUMA Approach:

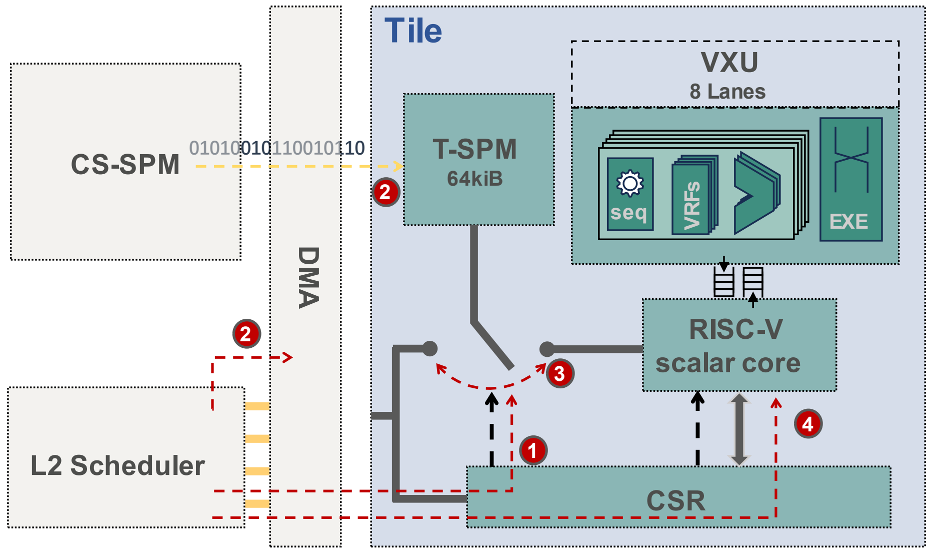
- ☐ Each tile and cluster has its own SPM, accessed by *outside* DMA.
- ☐ Eliminating fragment memory access.

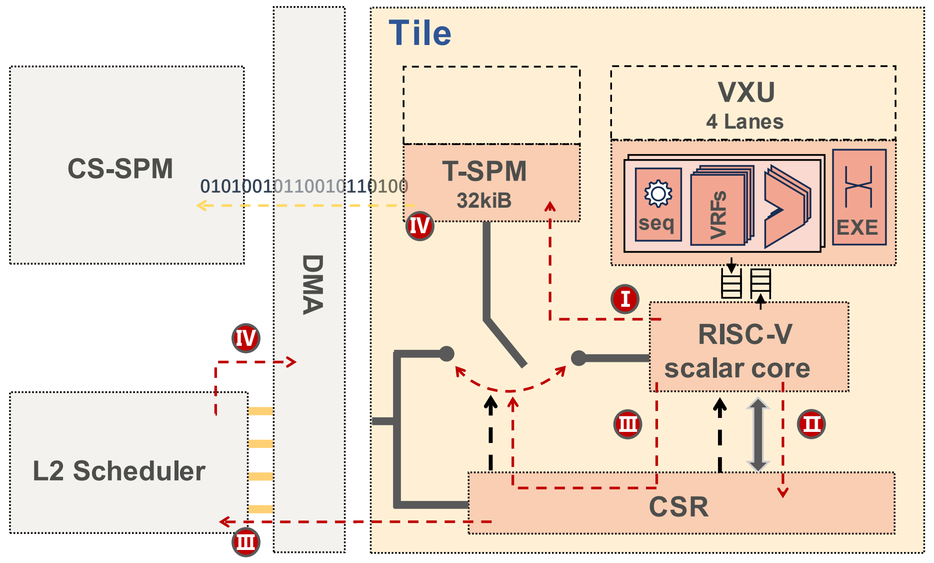
■ Before Execution

- ☐ Alter T-SPM direction by atomic instructions.
- ☐ DMA moves data from CS-SPM to T-SPM.
- ☐ Change back T-SPM direction to the RV core.
- ☐ De-assert core reset.

■ After Execution

- ☐ Store results in T-SPM.
- ☐ Notify attributes of return value to CSRs.
- ☐ Alter T-SPM direction & Issue an interrupt.
- ☐ DMA retrieve data back to CS-SPM.





System Design: Heterogeneous Configuration

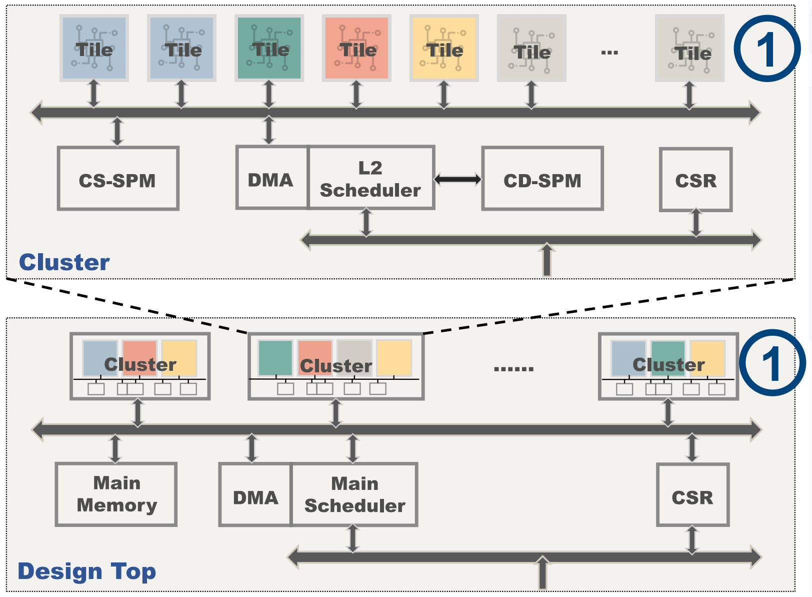
ACE-Lab

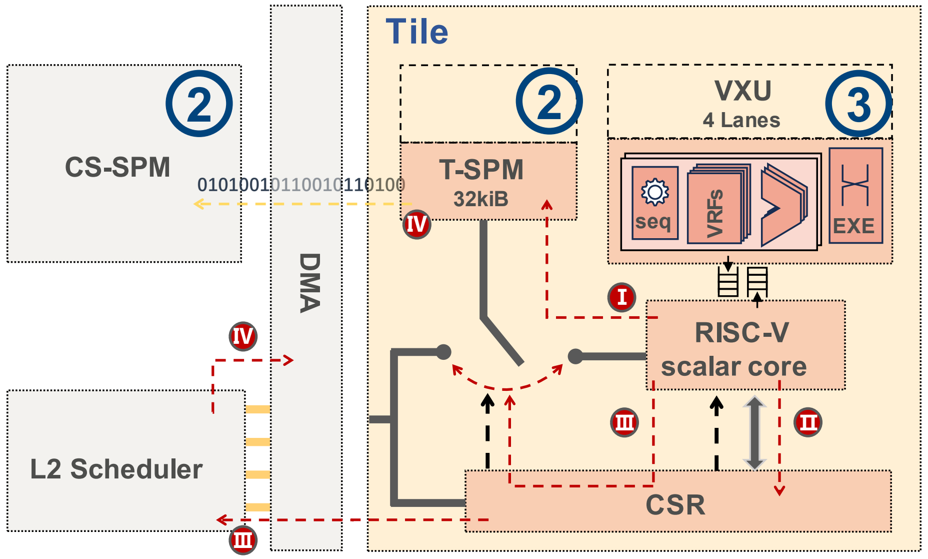
■ Configurable dimensions for WBP

① **# of clusters & tiles:** Enhancing thread- & task-level parallelism of processing Tx & Rx in consecutive time slots – dependent on protocol throughput.

② **SPM footprint:** Dependent on computation type: FFT, Polar decoding; Multi-threading capability.

③ **# of lanes and VRFs:** Enhancing DLP capabilities.





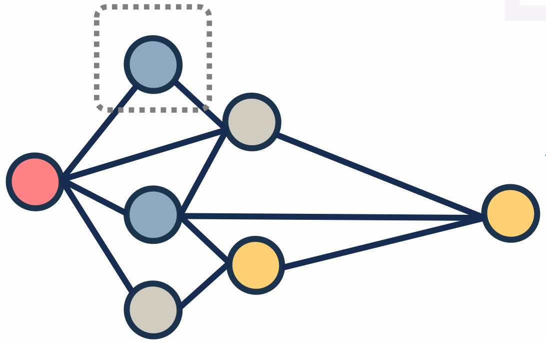
Execution Model: Dataflow Model

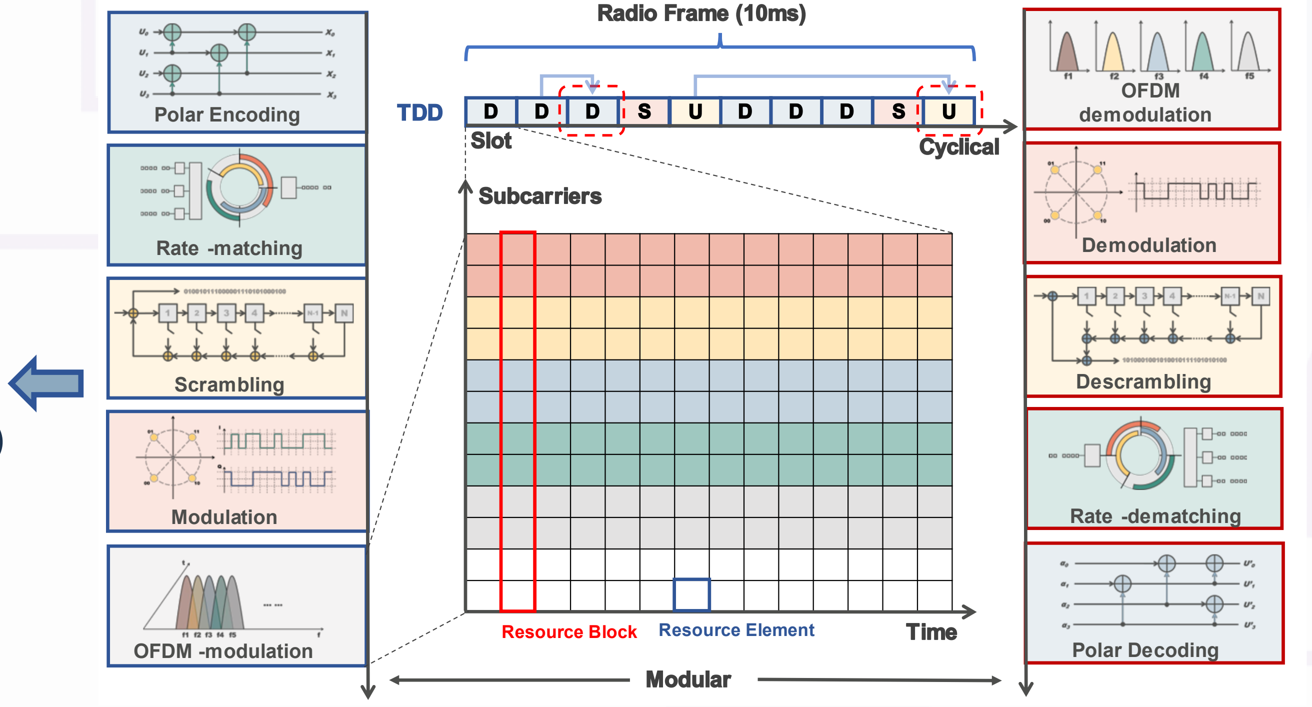
ACE-Lab

WBP interpreted as DAGs

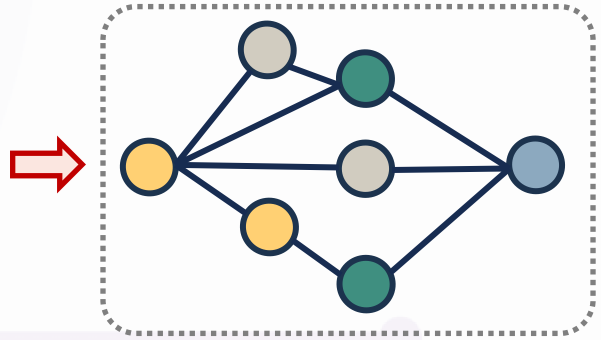
- A subsequent module is activated only when all preceding tasks are complete.
- WBP follows a consistent flow over time, enhancing data locality as the DAG information is unlikely to be reconfigured on the hardware.
- Less scheduler-bounded

Task: A module running on a tile.





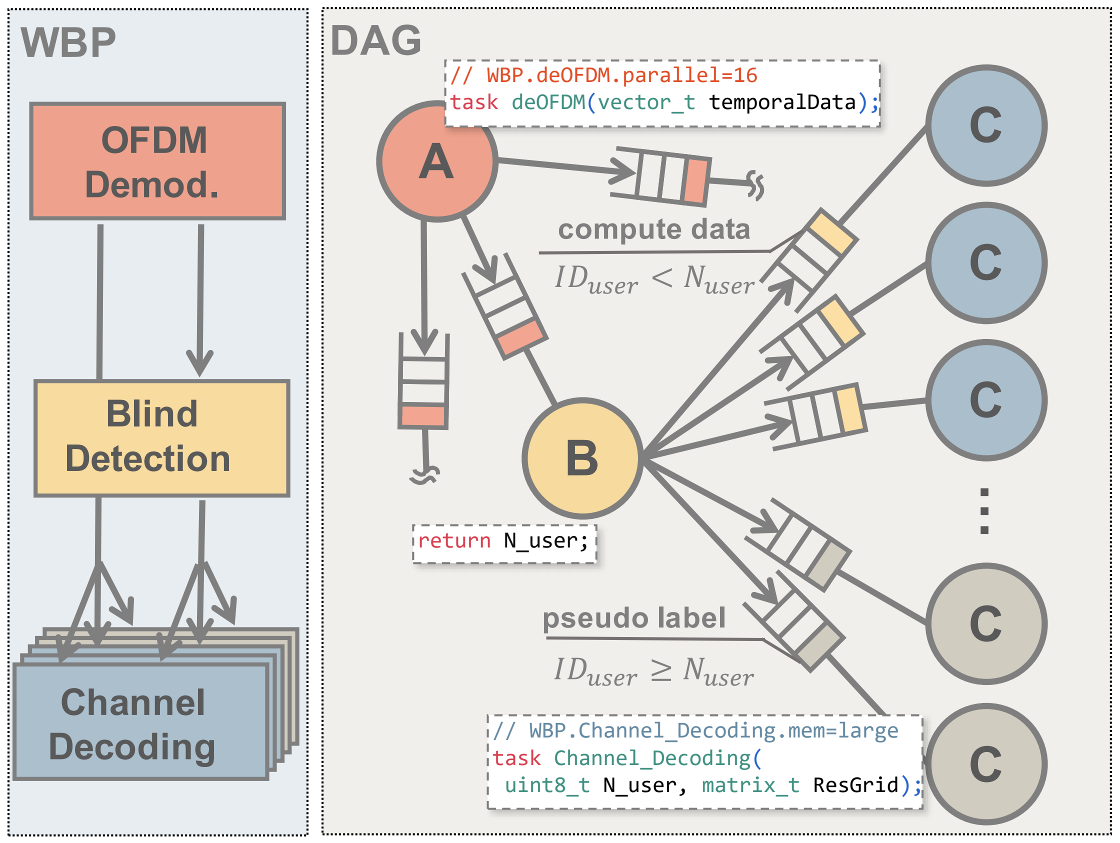
Thread: Several related tasks, representing a complete transmit or receive processing flow.



Execution Model: Dataflow Model

ACE-Lab

- Attributes guide the scheduler in selecting the most suitable tile for deployment.
- Runtime adjustment of DAGs
  - Tasks can be dismissed on-the-fly once the worst-case DAG is determined.
  - If the blind detection task detects fewer users, the computational burden can be reduced.



```
// WBP.deOFDM.parallel=16
task deOFDM(vector_t temporalData);
```

```
return N_user;
```

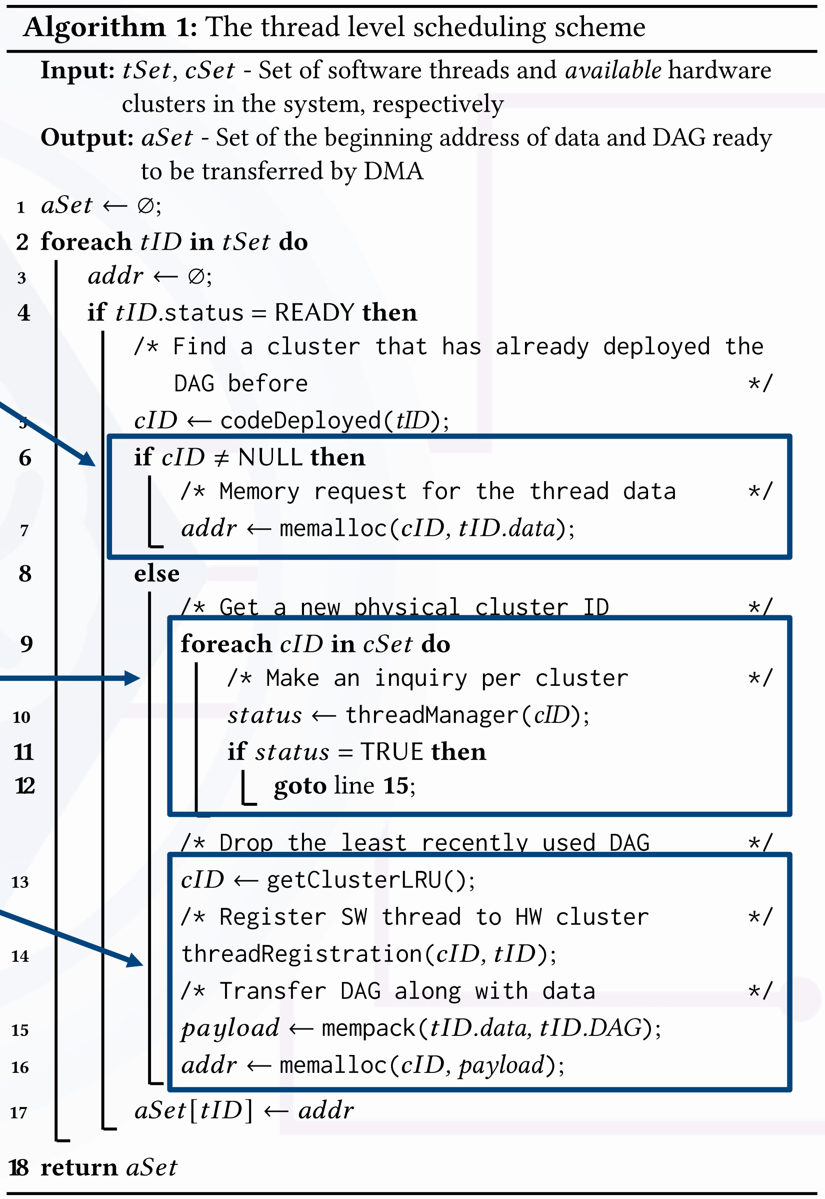
```
// WBP.Channel_Decoding.mem=large
task Channel_Decoding(
  uint8_t N_user, matrix_t ResGrid);
```

Execution Model: Multi-Level Scheduling

ACE-Lab

Thread-Level Lazy-Deletion

- Does not immediately free up DAG memory. Checks whether the DAG has already been deployed to a cluster and only transfers the data for the next thread issue.
- Checks the validity for running multiple threads within a single cluster.
- Drop the least recent used DAG when all available computing resources are occupied.



Algorithm 1: The thread level scheduling scheme

Input: *tSet*, *cSet* - Set of software threads and *available* hardware clusters in the system, respectively

Output: *aSet* - Set of the beginning address of data and DAG ready to be transferred by DMA

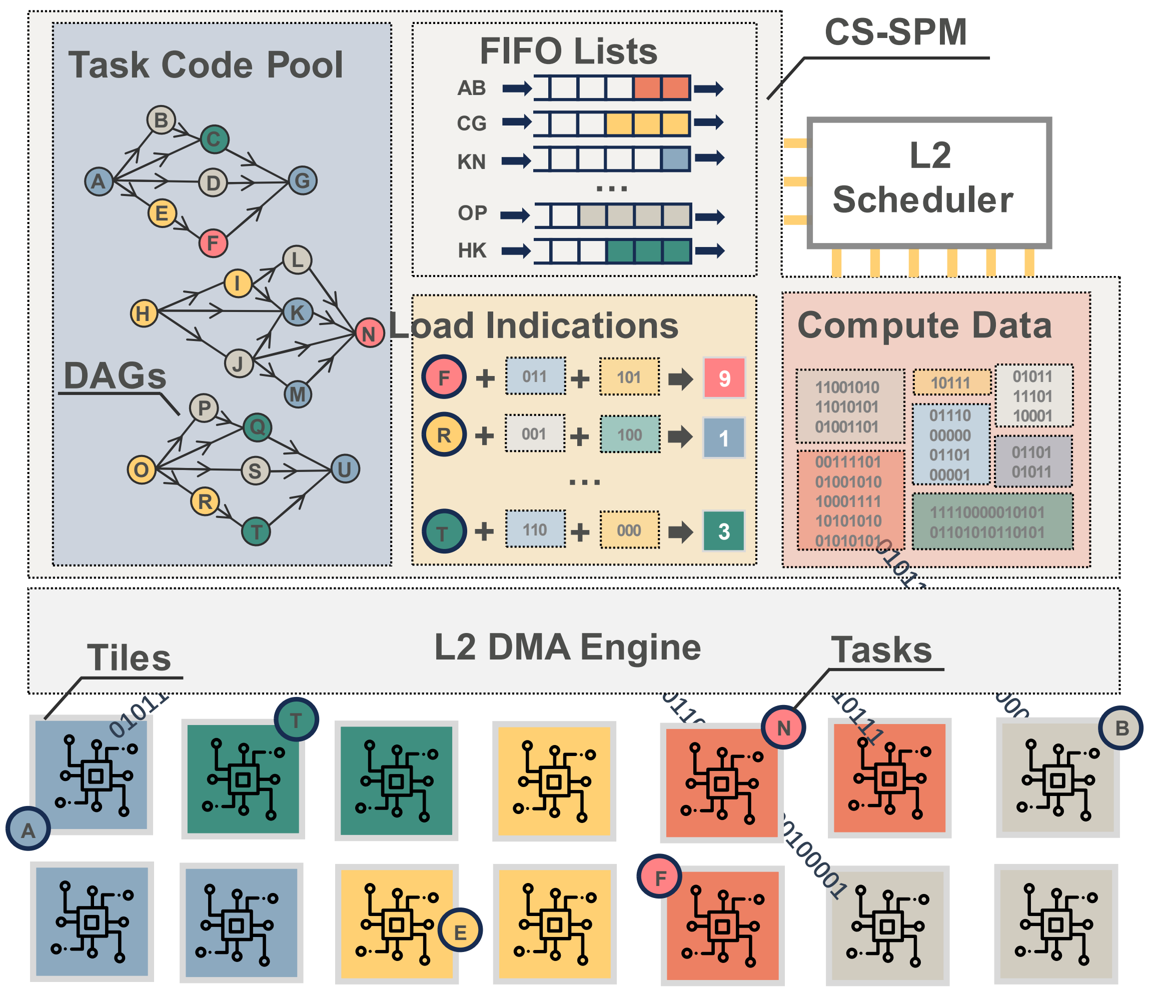
```
1  aSet ← ∅;
2  foreach tID in tSet do
3      addr ← ∅;
4      if tID.status = READY then
           /* Find a cluster that has already deployed the
              DAG before                                        */
5          cID ← codeDeployed(tID);
6          if cID ≠ NULL then
               /* Memory request for the thread data            */
7              addr ← memalloc(cID, tID.data);
8          else
               /* Get a new physical cluster ID                 */
9              foreach cID in cSet do
                   /* Make an inquiry per cluster               */
10                 status ← threadManager(cID);
11                 if status = TRUE then
12                     goto line 15;
               /* Drop the least recently used DAG              */
13             cID ← getClusterLRU();
               /* Register SW thread to HW cluster              */
14             threadRegistration(cID, tID);
               /* Transfer DAG along with data                  */
15             payload ← mempack(tID.data, tID.DAG);
16             addr ← memalloc(cID, payload);
17         aSet[tID] ← addr
18 return aSet
```

Execution Model: Multi-Level Scheduling

ACE-Lab

■ Tile-Level Scheduling

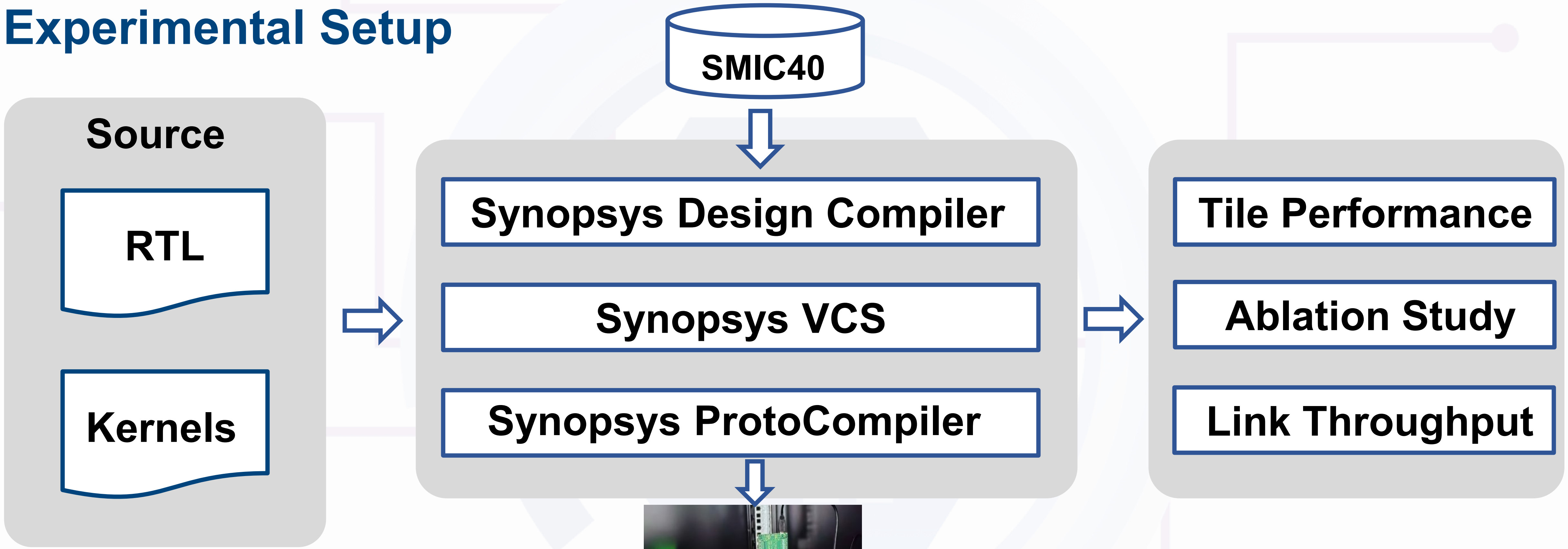
- ◻ **L2 Scheduler**: Processes the nodes (tasks) in the **task code pool** and checks their readiness through the FIFO queues.
- ◻ **FIFO Lists:** Track edges between the DAG nodes.
- ◻ **Load Indications:** Task to be processed and the preferred tile.
- ◻ **L2 DMA:** Transfers data from the **compute data** section to the heterogeneous tiles



Evaluation

ACE-Lab

■ Experimental Setup



Evaluation

ACE-Lab

Single-Tile Performance

- @ 500 MHz
- Lies between commercial hardware and ASICs
  - 2.3x in FFT & 2x in BP
  - Still suffer from Von Neumann bottleneck

Config: A 64-lane, 50.1 GOPS VXU

Kernel	Platform	Dec. Length	Norm. Thrpt.
BP Decode	GPU	512	0.25
		1024	0.21
	ASIC	1024	15.23
	Ours	512	0.54
		1024	0.53

Kernel	Platform	FFT Length	Clock Cycles
FFT	DSP	128	588
		512	2559
		2048	11922
	HW Accel.	128	211
		512	845
		2048	3875
	Ours	128	251
		512	1122
		2048	5073

Evaluation

ACE-Lab

■ Ablation Study

- ☐ 12T vs. 3C4T-2L2S
- ☐ 6.5% power increase; 1.3x throughput
- ☐ Under-utilization in single-level arch.
- ☐ 6.4% and 9.5% gain under lazy-deletion

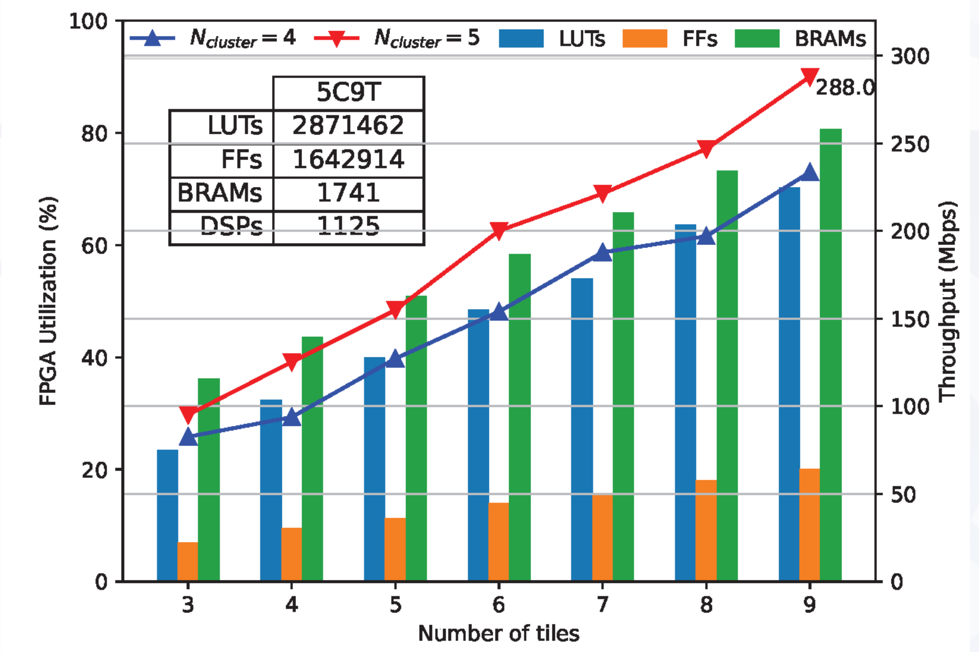
Config:
Large (L) Tile (T): w/ 64-lane VXU, 32 VRFs
Small (S) Tile: w/ 8-lane VXU, 64VRFs

Baseline + extra features	Power (W)		Throughput (Mbps)	
	12T	3C4T	Single-Level	Multi-Level
12T / 3C4T arch.	3.24	3.45	8.5	21.2
+ Multi-Threading			64.1	84.7
+ Lazy-Deletion			68.5	93.6

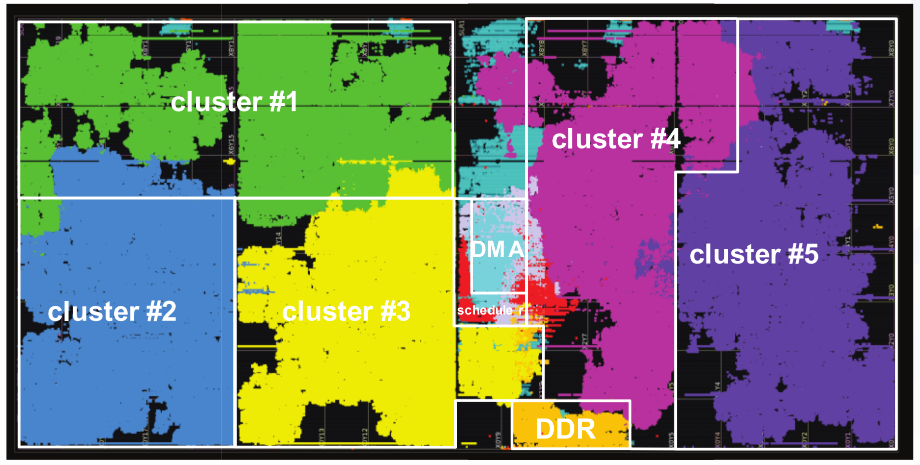
Evaluation

■ Link throughput experiment on prototype

- ☐5C9T
- ☐288Mbps



	5C9T
LUTs	2871462
FFs	1642914
BRAMs	1741
DSPs	1125



Module	Configuration
Channel Coding	Polar Codes
Rate-Matching	RV0
Scrambling	Gold Sequence
Modulation	QPSK
OFDM	128 subcarriers
Channel Estimation	Least Squares
Channel Equalization	Zero Forcing
Channel Decoding	Min-Sum BP

Conclusion

- We propose a **pack-and-ship** approach within a **cache-free** NUMA system.
  - Instructions and data are organized in bundles and delivered by schedulers to local scratchpad memory in order to **reduce data movement** costs.

- We also develop a **hierarchical dataflow** scheme along with two strategies, namely **multi-threading** and **lazy-deletion**, to exploit and allocate the hardware resources more efficiently.

- Our HW/SW co-design surpasses the existing architectures attributed to strong single-tile performance as well as flexible scalability and coarse-grained parallelism.

Q&A
Thank you for your attention!

A Hierarchical Dataflow-Driven Heterogeneous Architecture for Wireless Baseband Processing

Presenter: Limin Jiang
jianglimin@shu.edu.cn
Shanghai University