AI-driven Memory Exploration in RTL Architect

Efficient Physical-aware PPA Enhancement

Shu Rong Lee
MediaTek

Agenda

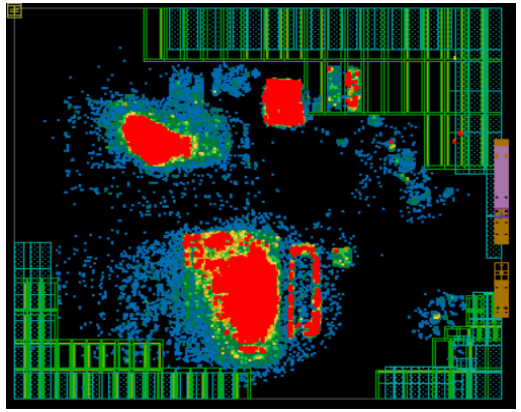
- Motivation
- **R**einforcement **L**earning **M**emory exploration **M**ethodology (RLMM)
- RLMM Experimental Results
- Summary

Motivation – Physical Awareness

Considering physical implementation at the RTL design stage

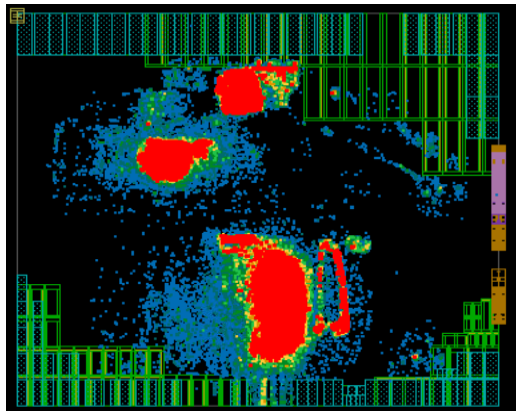
- RTL designer decides the type of memories and creates a behavioral model for simulation and verification
  - Insufficient for synthesis

- Memory configuration has a big impact on PPA and needs to be physically-aware
  - Existing solutions consider only each individual memory, not physical implementation with all memory wrappers
  - The best configuration for each memory wrapper based on weight does not often correspond to the best for physical implementation
  - The number of combinations of memory configurations for physical implementation is huge – it is impossible to explore all of them physically



Memory config 1

GRC overflow: 14.83%
Total power: 27.4 mW



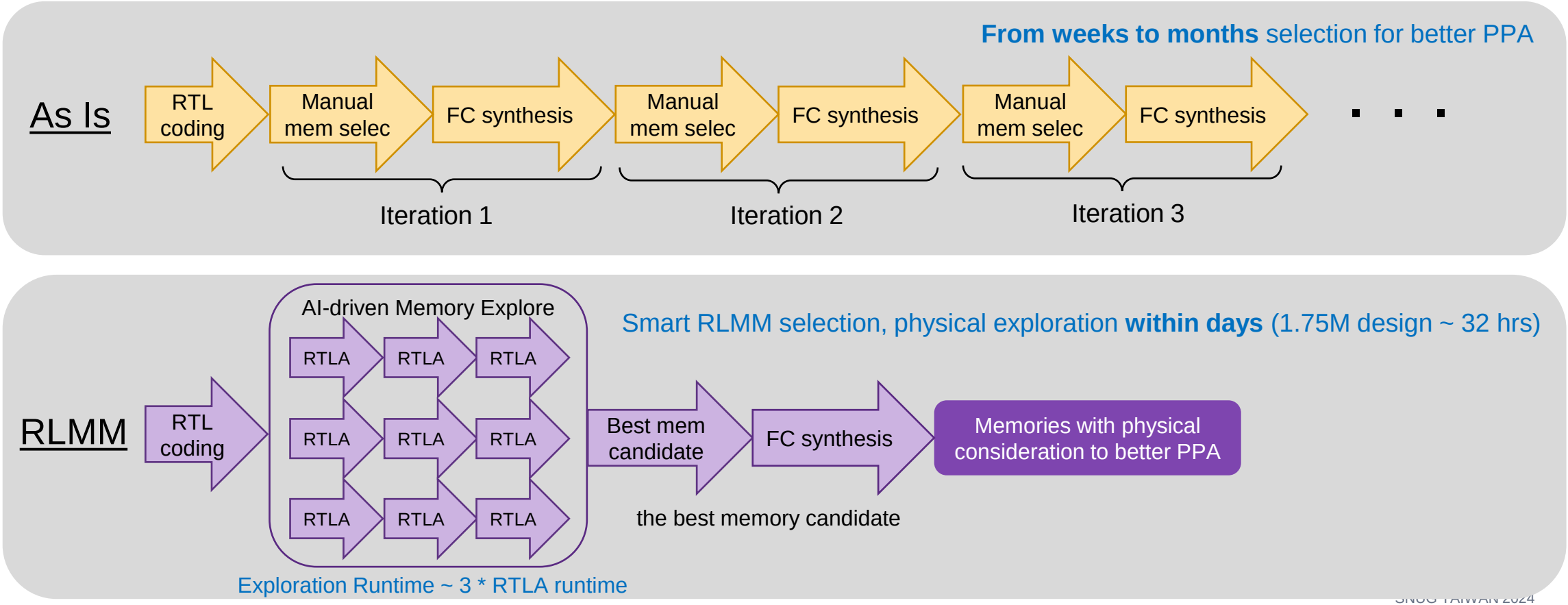
Memory config 2

GRC overflow: 17.35%
Total power: 26.7 mW

Motivation – Feedforward Guidance

Reduce project schedule with an automated flow before synthesis

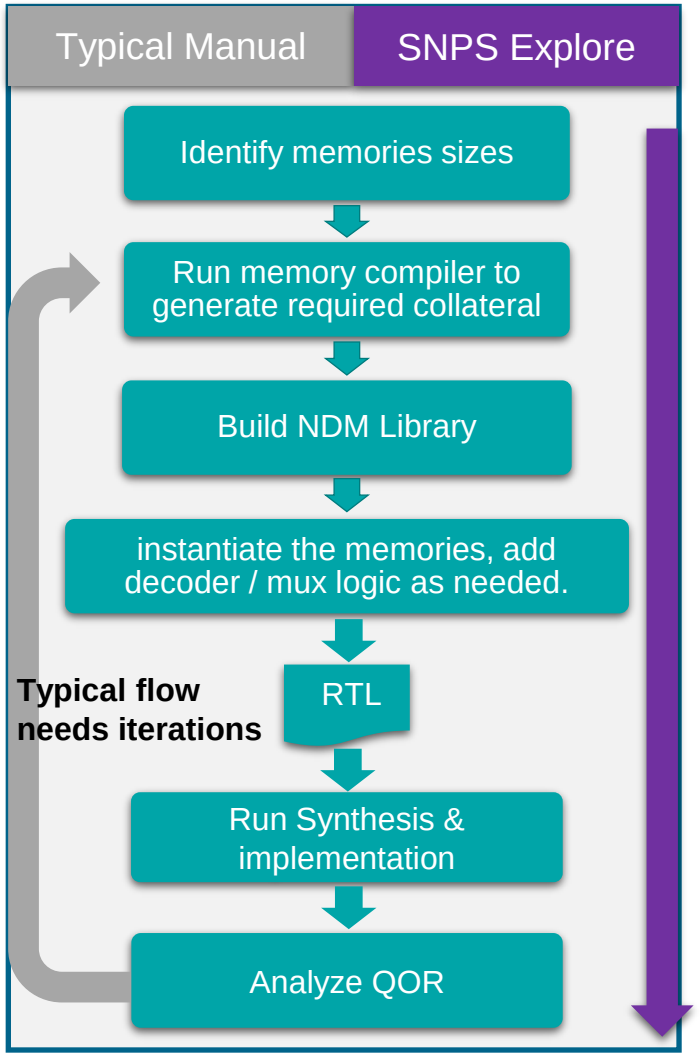
- Manual selection needs iterative work to check physical PPA
- Leverage RTL Architect's quick runtime to do early design space exploration



RTL Architect Memory Exploration Methodology

AI-based Reinforcement Learning

- **Challenges**
  - The number of combinations of memory configurations for physical implementation is huge – it is impossible to explore all of them physically
- **Reinforcement Learning (RL) reduces the iterations to the first best result**
  - Reinforcement Learning (RL) is the science of decision-making -- it is about learning the optimal behavior in an environment to obtain the maximum reward
- **Key Features / Advantage**
  - Native Synopsys Memory Compiler support
  - NDM support to feedforward to Fusion Compiler
  - Memory splitting, with control logic inserted
  - Powerful filtering capabilities and AI memory selection controls



RLMM Key Steps

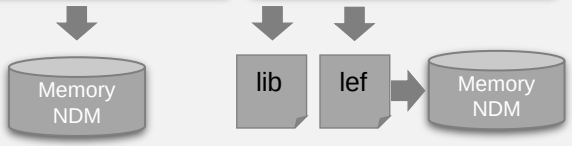
Identify and Generate

RTL is read with behavioral memory wrappers and the user identifies the required memory instances for exploration

Memories are selected automatically, with support for user-weighting for Area / Performance / Power trade-offs

For 3d-party Memory Compiler flows, the user needs to provide memory NDM as input

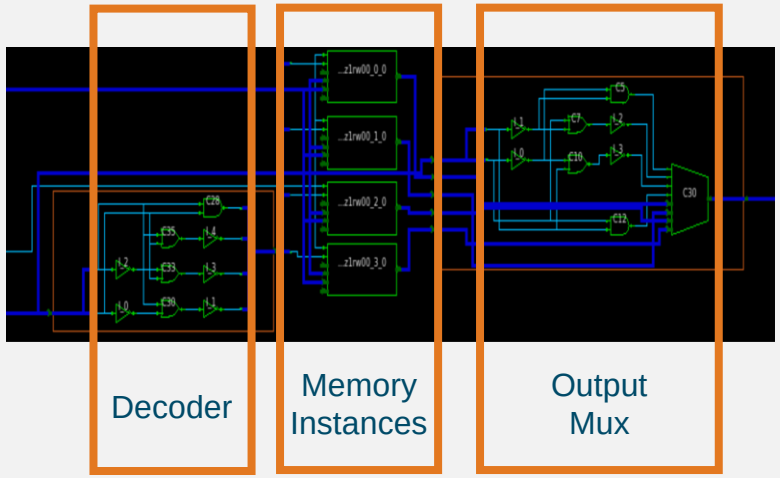
Synopsys Memory Compiler will automatically run and create the NDM



Select and Instantiate

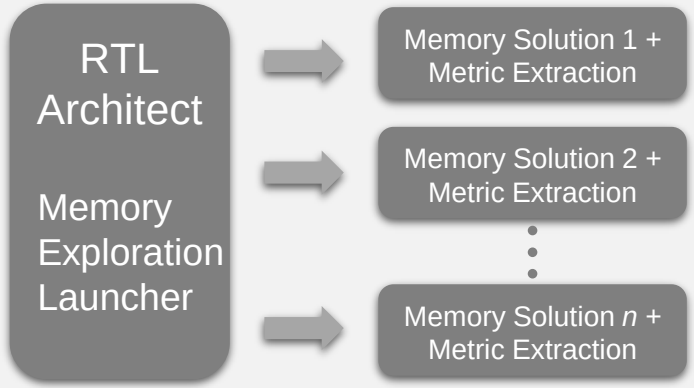
Top memory candidate solutions are selected (with user control over the number of runs)

Memories are automatically instantiated, with required control logic (where needed)



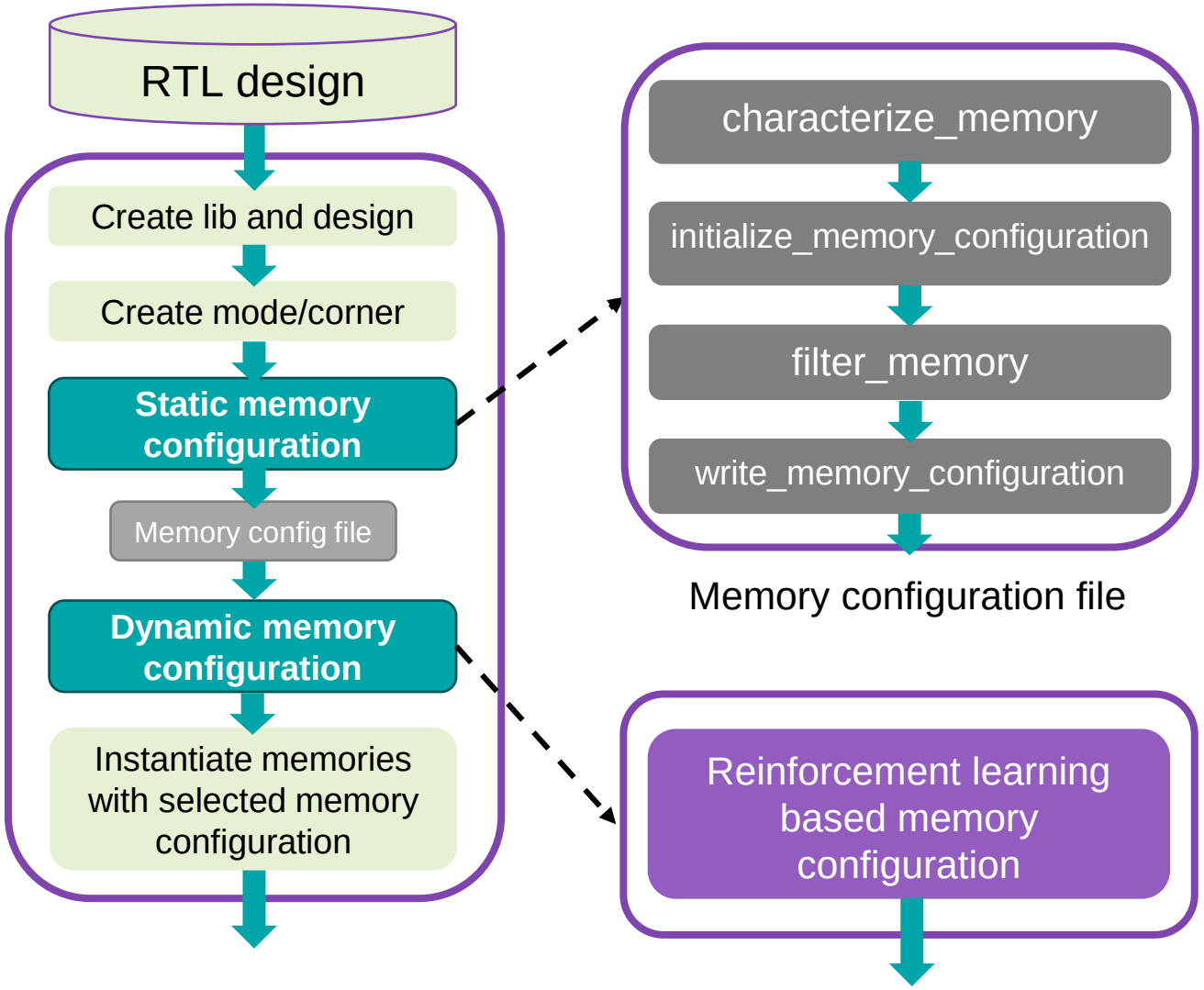
Explore and Analyze

Exploration Runs Launched in Parallel



Runs can be analyzed / compared

RLMM Static and Dynamic Selection



Static: choose top *n* memories from library characterization based on weighting

Collect data from all memories in reference libs

Derive module interface and instance sections of memory configuration

Find **top *n* memories** for each wrapper, update memory config with the result of filtered memories

Auto configuration file generation
Output characterized configuration data

Dynamic: perform reinforcement-learning physically-aware selection

Apply reinforcement learning (RL) to find the best memory configuration

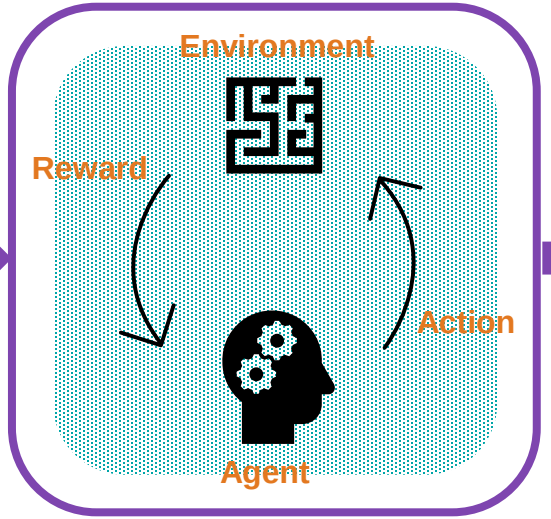
RLMM Dynamic Workflow

- Based on config file, hook up memory connections and generate the corresponding control logic
- With reinforcement learning, the exploration search space is effectively reduced

Memory configuration file generated automatically from static selection

```
"instance_section": {
    "u0_3-pd_switchable-rf0-wrapper_snps_": {
        "memories": [
            "DWC_SRM8TDP_256x32",
            "DWC_SRM8TDP_1024x8",
            "DWC_SRM8TDP_64x32"
        ],
    },
    "u_m-u_m_pd-dsu0-x0-mem0-wrapper_snps_": {
        "memories": [
            "DWC_SRM8TDP_64x32",
            "DWC_SRM8TDP_256x32",
            "DWC_SRM8TDP_1024x8"
        ],
    },
    "u0_0-pd_switchable-cmem0-dtags0_0": {
      ...,
        "memories": [
            "DWC_SRM8TDP_32x39",
            "DWC_SRM8TDP_32x22"
        ],
    },
    "u0_0-pd_switchable-cmem0-dtags1_0": {
        "memories": [
            "DWC_SRM8TDP_32x22",
            "DWC_SRM8TDP_32x39"
        ],
    }
```

Reinforcement Learning memory configuration



An intelligent agent takes actions in a dynamic environment to maximize the cumulative reward

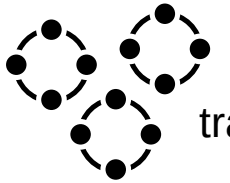
Memory candidates for the best physical PPA

```
Result #5:
{u_mdvlpsys-mdvlp_dvfs_top-u_sram_dm0 TGELSCRIJMANSB2KX32B256M8SW1A 1x1}
{u_mdvlpsys-mdvlp_dvfs_top-u_sram_pm1 TGEFSCRIJMANSB4KX26B128M8SW0A 1x1}
{u_mdvlpsys-mdvlp_dvfs_top-u_sram_dm1 TGEFSCRIJMANSB2KX32B128M8SW1A 1x1}
{u_mdvlpsys-mdvlp_dvfs_top-u_sram_pm0 TGEFSSCRIJYMANSA1KX26B128M4SW0A 1x1}
```

PPA report of selected memory configuration

Summary Table Comparator

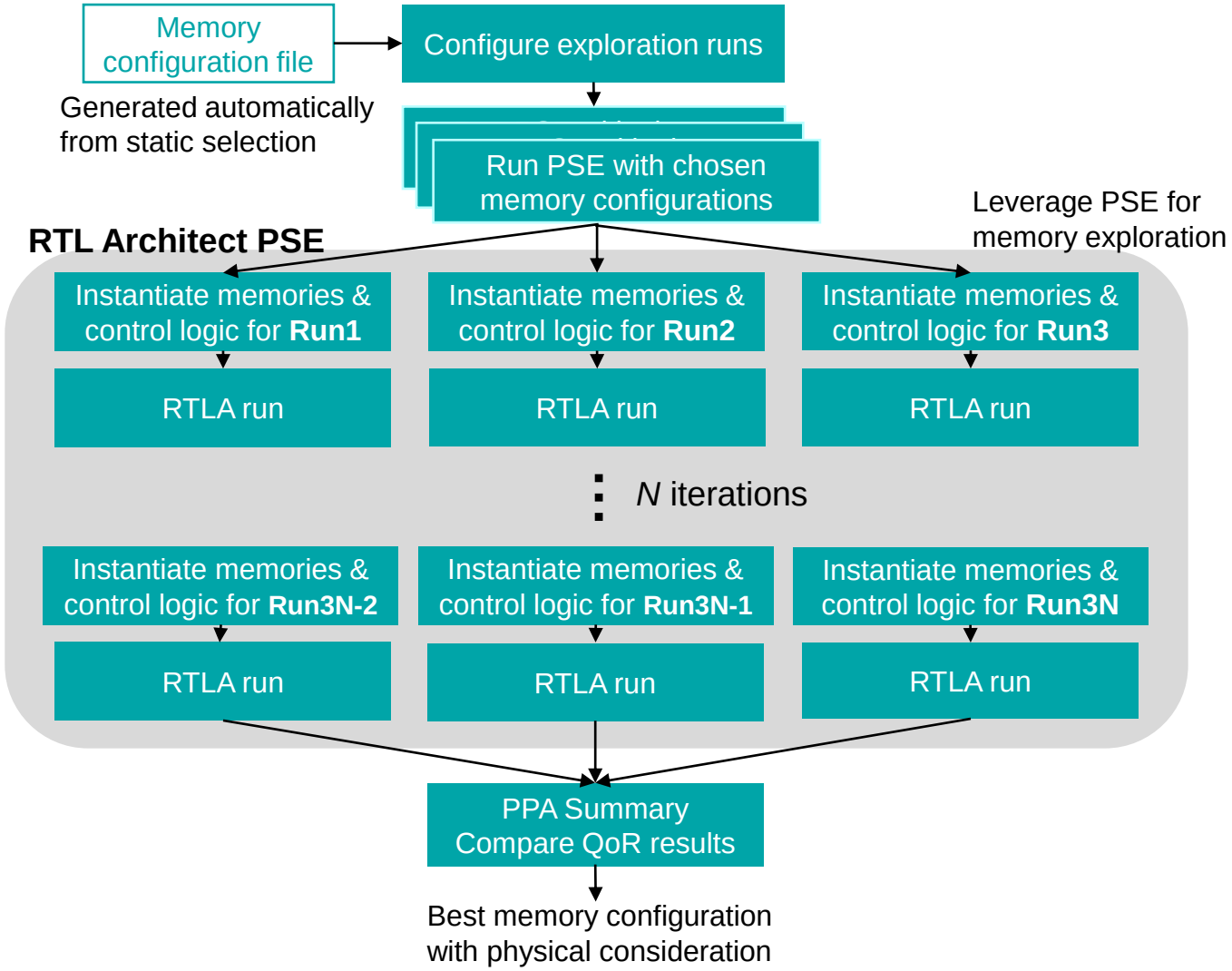
Run	Timing						Instance Count			
	WNS	TNS	NVE	Trans Cost	Trans Violation Num	Cap Violation Num	Standard	Register	Repeaters	ICG's
MEM1_4	0.04	0.00	0	0.00	0	0	1709	26	147	1
MEM1_3	0.04	0.00	0	0.00	0	0	1651	19	140	1



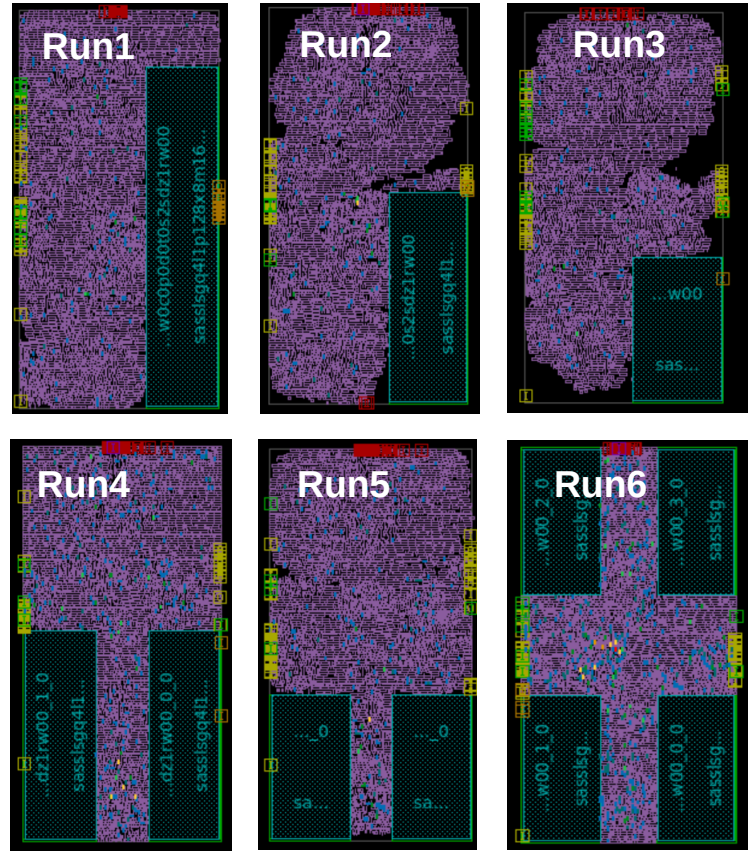
trained ML model

Explore with Parallel Space Exploration

Leverage PSE for parameter sweeping

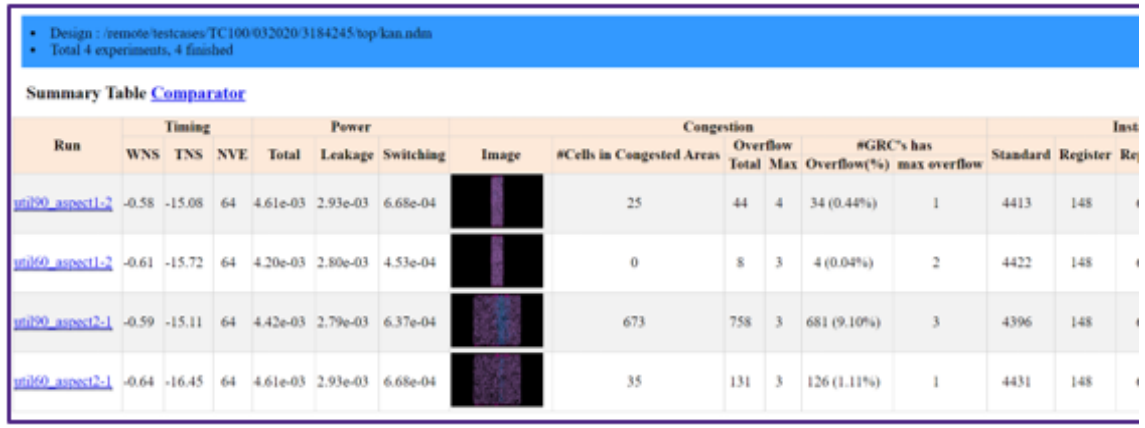


- PSE explores memory configurations and placement
- Find the best PPA among multiple objectives – overall consideration of **congestion/power/timing/area**

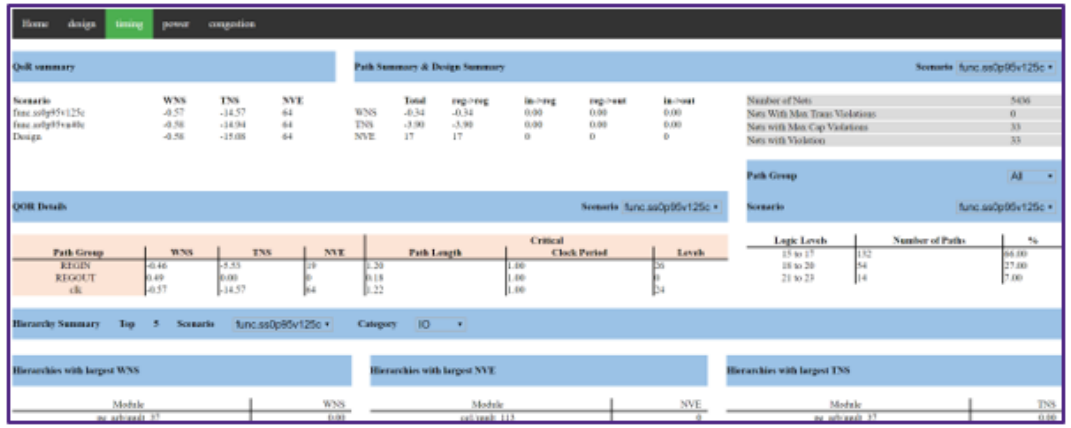


Parallel Space Exploration PPA Summary

- PSE collects PPA Summary for each memory configuration in HTML format



Run	Timing			Power			Congestion							Inst
	WNS	TNS	NVE	Total	Leakage	Switching	Image	#Cells in Congested Areas	Overflow Total	Overflow Max	#GRC's has Overflow(%)	max overflow	Standard	Register
util90_aspect1-2	-0.58	-15.08	64	4.61e-03	2.93e-03	6.68e-04		25	44	4	34 (0.44%)	1	4413	148
util60_aspect1-2	-0.61	-15.72	64	4.20e-03	2.80e-03	4.53e-04		0	8	3	4 (0.04%)	2	4422	148
util90_aspect2-1	-0.59	-15.11	64	4.42e-03	2.79e-03	6.37e-04		673	758	3	681 (9.10%)	3	4396	148
util60_aspect2-1	-0.64	-16.45	64	4.61e-03	2.93e-03	6.68e-04		35	131	3	126 (1.11%)	1	4431	148



QORsum

Runs | Metrics | Info | Settings

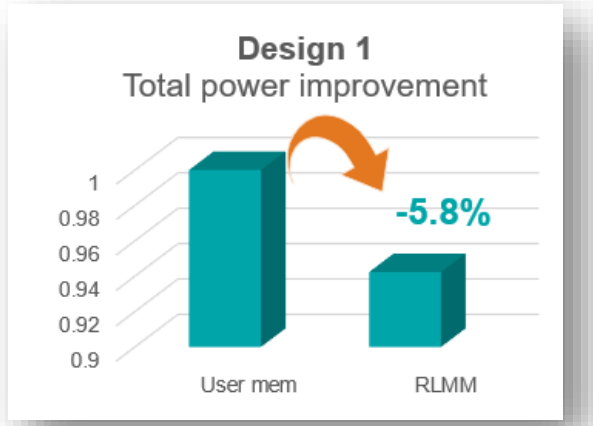
QOR Summary
Performance
App Option Details
Timing / LDRC
Timing Summary
Path Group Details
Logical DRCs

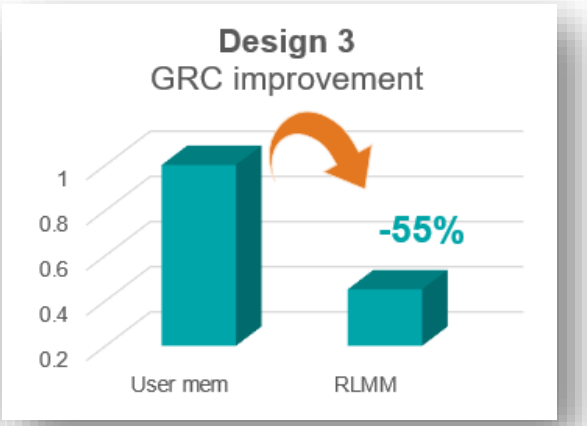
	Run	Setup						Logical DRCs		Netlist				Power				Routability	Runtime		Parameters
		WNS	TNS	NVE	r2rWNS	r2rTNS	r2rNVE	TranDRCs	CapDRCs	Util	StdCellArea	StdCells	HBufs	TotalPwr	LeakPwr	Gated%	Bits/Flop	Overflow%	TotalRun	StageRun	memory_cell
1	MEM1_1	0.110	0.0	0	0.000	0.0	0	0	0	59.0	2037	6172	0	1488200	21200	10.9	1.00	0.35	0.13	0.13	MEM1_1
2	MEM1_2	0.130	0.0	0	0.000	0.0	0	0	0	51.3	2027	6139	0	1465400	20400	10.9	1.00	0.36	0.14	0.14	MEM1_2
3	MEM1_3	0.120	0.0	0	0.000	0.0	0	0	0	48.7	2023	6173	0	1470800	20800	10.9	1.00	0.31	0.12	0.12	MEM1_3
4	MEM1_4	0.100	0.0	0	0.000	0.0	0	0	0	65.8	2028	6119	0	1472700	22700	10.9	1.00	0.94	0.11	0.11	MEM1_4
5	MEM1_5	0.100	0.0	0	0.000	0.0	0	0	0	57.7	2044	6168	0	1487800	24800	10.9	1.00	0.56	0.11	0.11	MEM1_5
6	MEM1_6	0.080	0.0	0	0.000	0.0	0	0	0	86.5	2021	6173	0	4207000	23000	12.5	1.00	1.39	0.17	0.17	MEM1_6

QoR Summary for the top 6 candidates – the User can specify the numbers of rankings

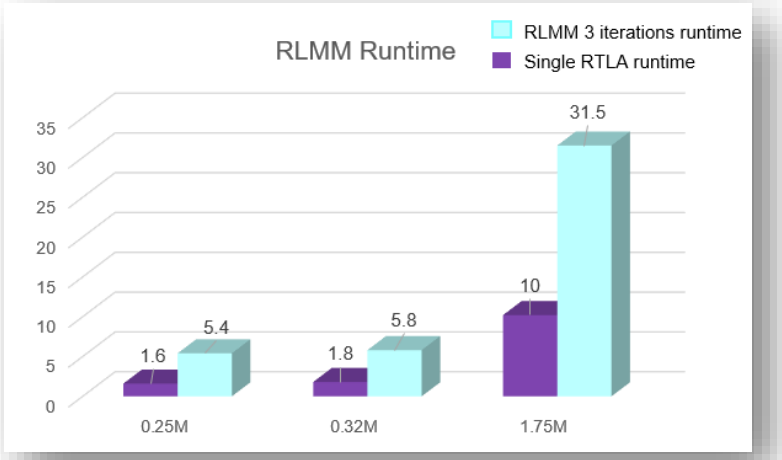
RLMM Experimental Results

- Validated in FC with AI-suggested memories
  - **Design 1** (4 wrappers) : Focus on power reduction
    - 5.8% power reduction with updated mem configuration and re-placement
  - **Design 2** (54 wrappers) : Focus on power reduction
    - 2.8% power reduction with updated mem configuration and re-placement
  - **Design 3** (135 wrappers) : Focus on congestion reduction
    - GRC down from 1.26% to 0.58%, reduced 51% track overflow with updated mem configuration and re-placement
- With RTL Architect's quick runtime, exploration TAT is reduced from months to hours





Power improved without area/timing/GRC overhead



Runtime is dependent on design size, irrelevant to number of wrappers

Summary: RTL Architect's RLMM

- Memory selection during the RTL design stage should consider physical implementation → **RLMM** provides AI-driven memory exploration for physically-aware PPA optimization
- **RLMM** facilitates exploration flow with automation instead of manual iterative loops → Memory selection TAT can be reduced from months to hours
- **RLMM** testing produced 3%-6% power improvement and 51% track overflow reduction, and the results were validated with Fusion Compiler synthesis
- **RLMM** helps with AI-based smart exploration and physical placement with quick optimization runtime at the RTL development stage


THANK YOU
Our Technology, Your Innovation™