RTL-MP: Toward Practical, Human-Quality Chip Planning and Macro Placement

Andrew B. Kahng[1] [2],

Ravi Varadarajan[2] **and Zhiang Wang**[2]

[1]Department of Computer Science and Engineering, University of California, San Diego

[2]Department of Electrical and Computer Engineering, University of California, San Diego

Outline

- Background and Motivation
- Main Contributions
- Our Methodology
- Experimental Setup and Results
- Conclusion and Future work

Background and Motivation

- **Macro placement is crucial to SoC design closure**
  - Sets the direction for all of place-and-route
  - Basis of architecture/RTL exploration, and design planning
- **Machine learning accelerators make macro placement more challenging**
  - Autogenerated complex RTL structure with long, inscrutable autogenerated module names
  - Hundreds of macros (see following TABLA designs as examples)

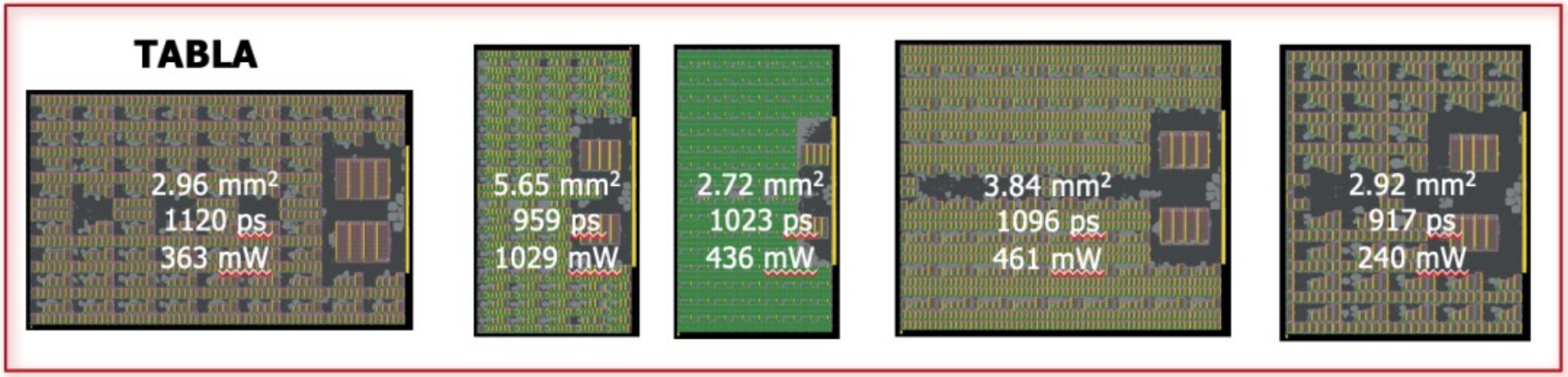
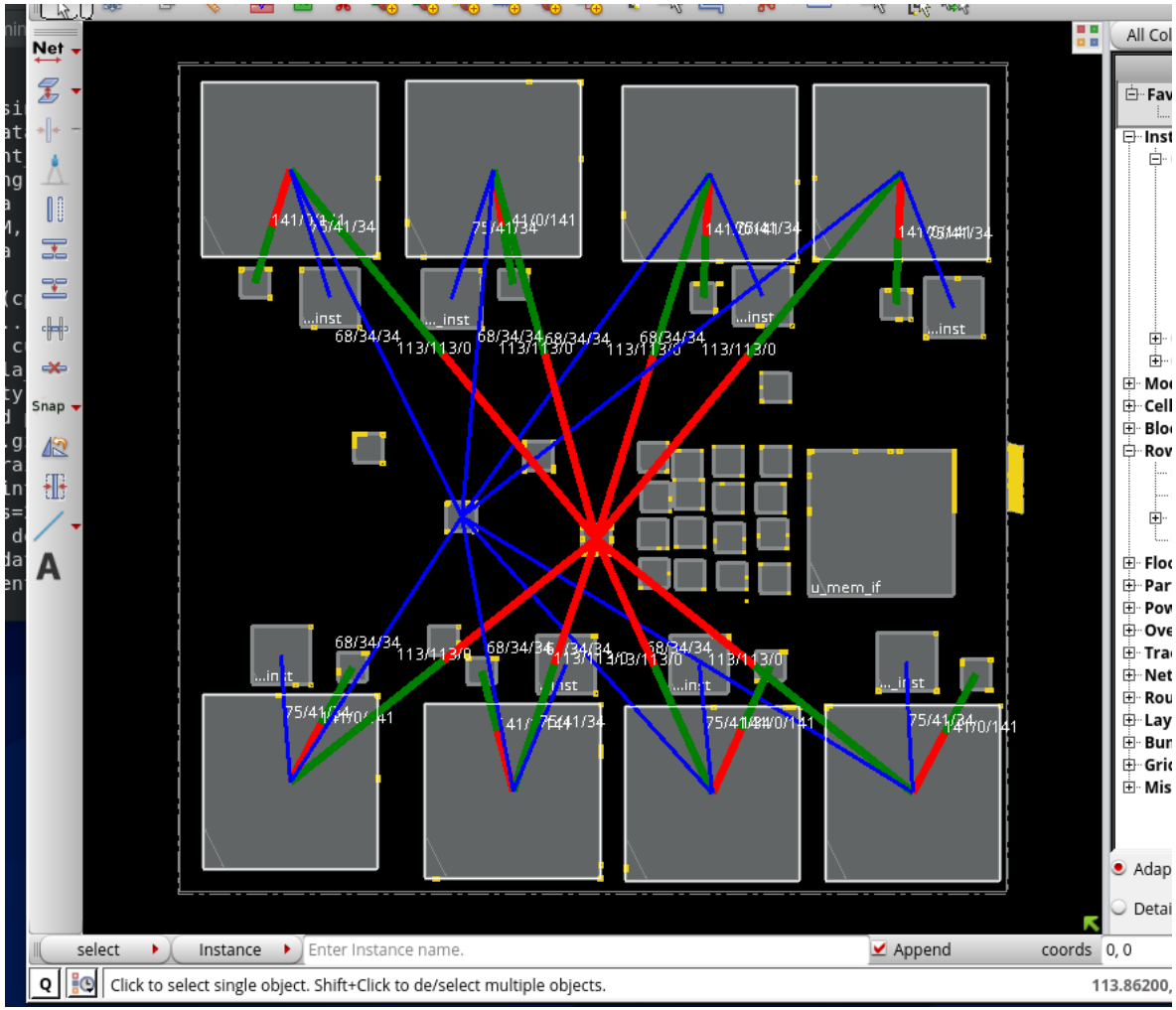


TABLA designs are produced in an open-source project [VeriGOOD-ML]

Background and Motivation

- **How do human physical design experts create high-quality macro placements?**



Example floorplan analysis of TABLA designs

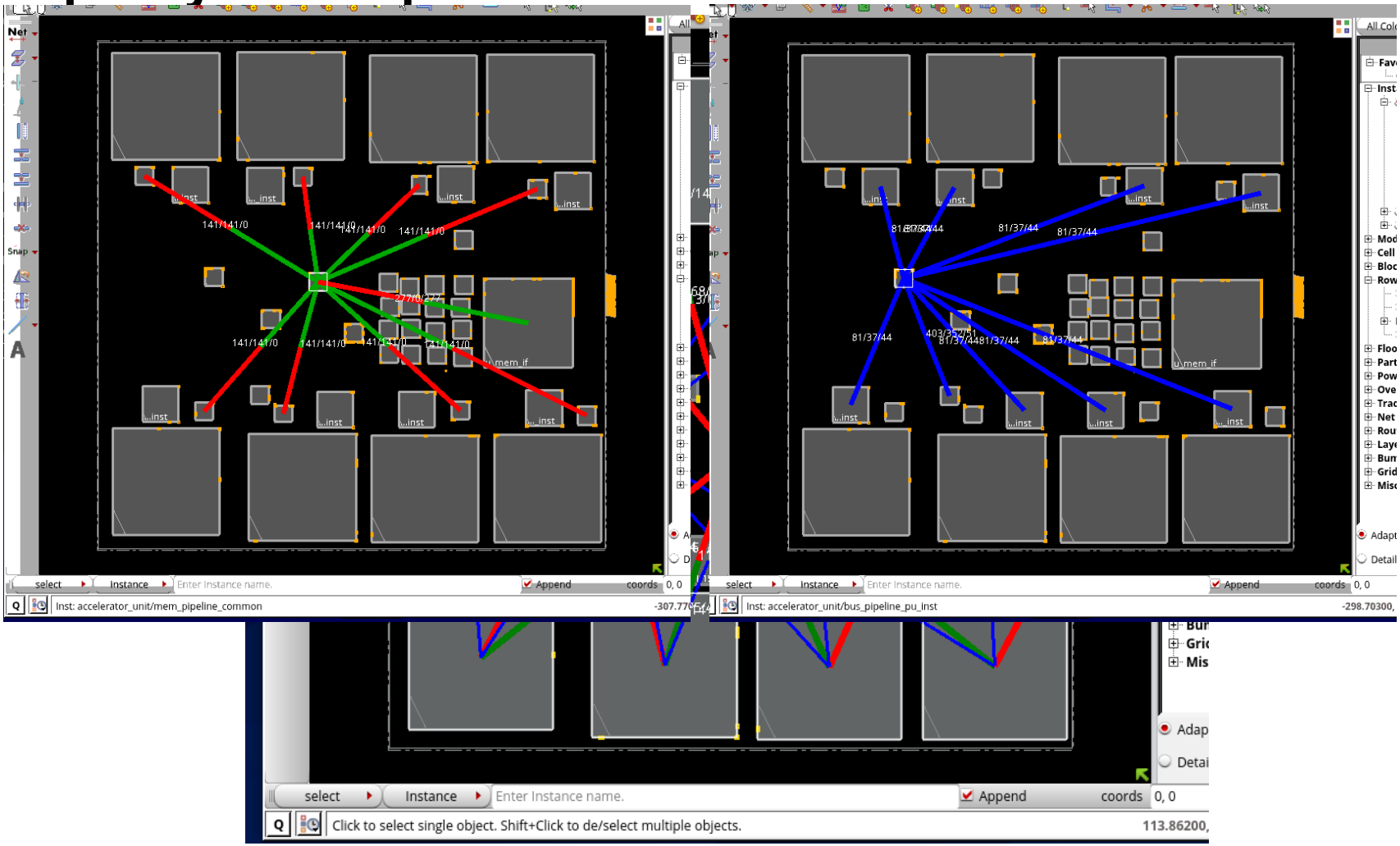
Background and Motivation

- **How do human physical design experts create high-quality macro placements?**

Example floorplan analysis of TABLA designs

Background and Motivation

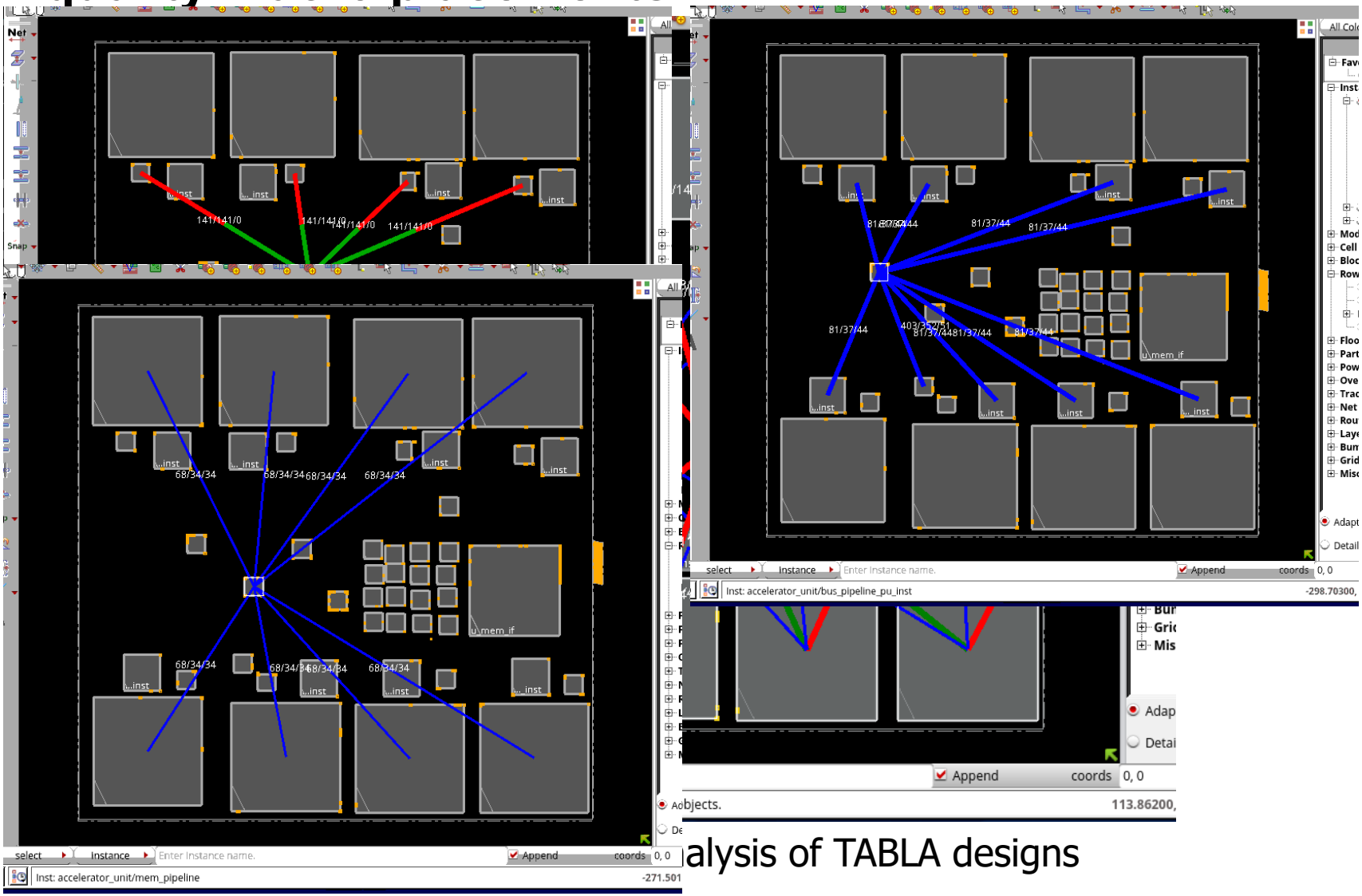
- **How do human physical design experts create high-quality macro placements?**



Example floorplan analysis of TABLA designs

Background and Motivation

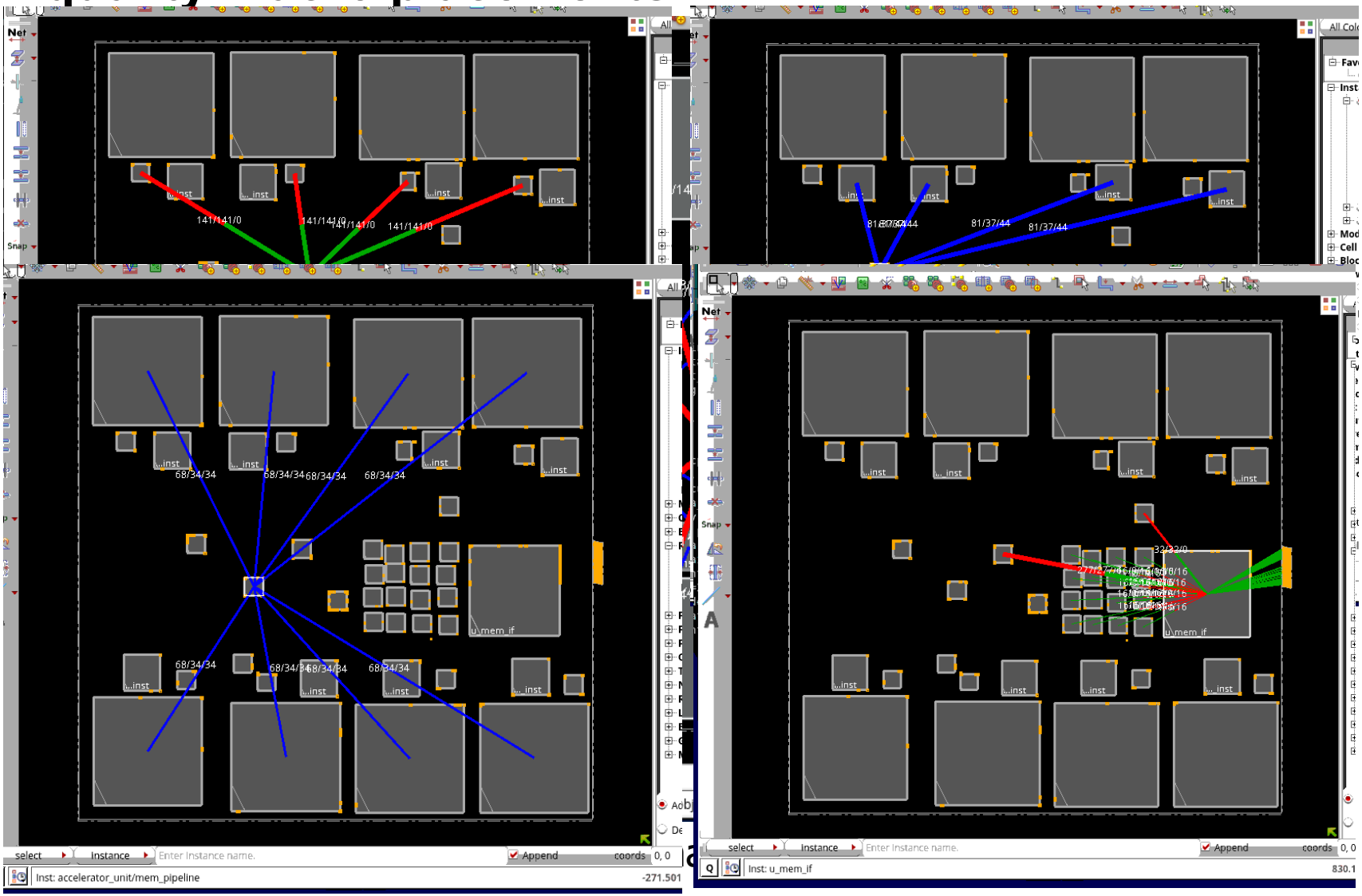
- **How do human physical design experts create high-quality macro placements?**



alysis of TABLA designs

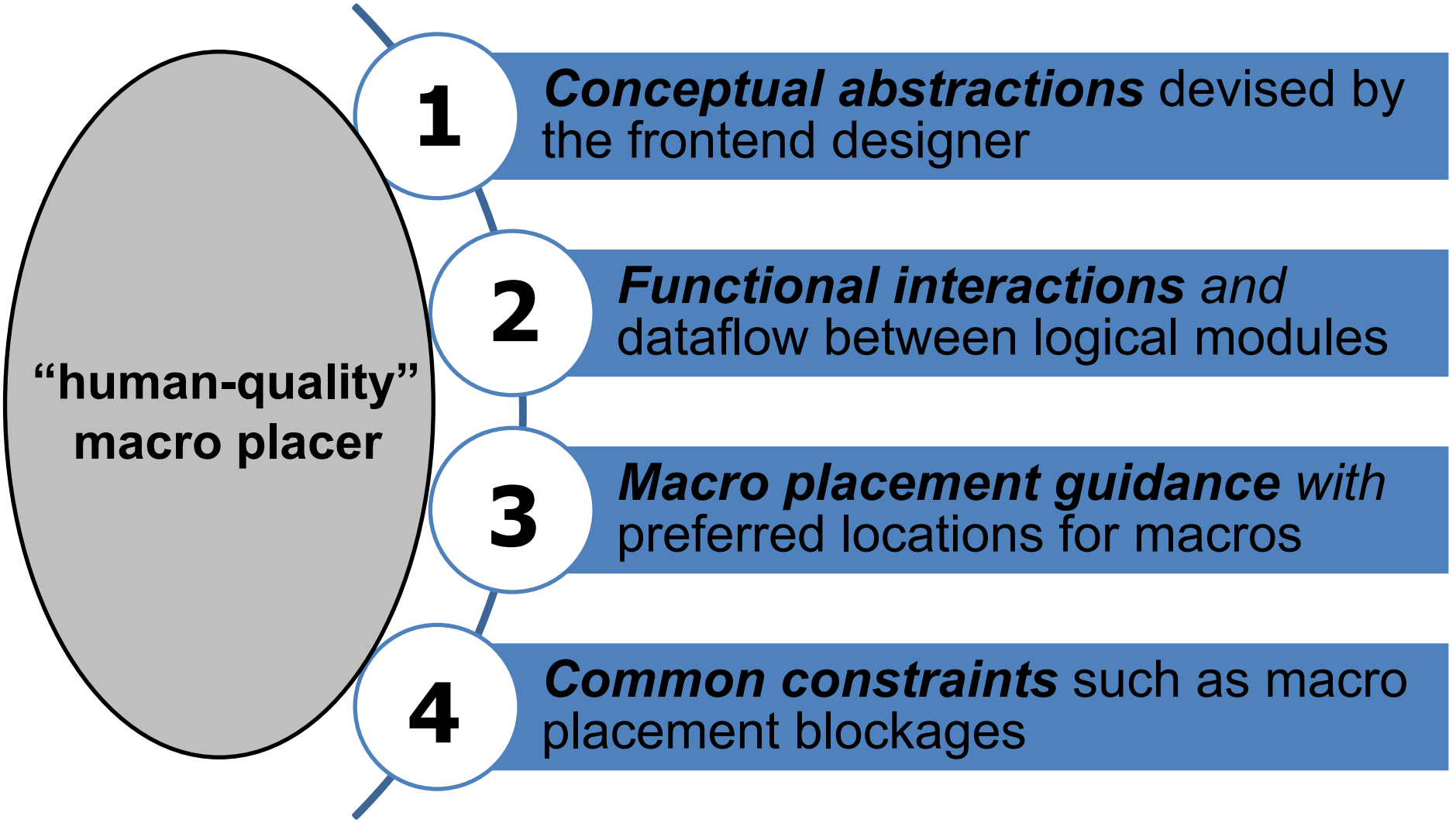
Background and Motivation

- **How do human physical design experts create high-quality macro placements?**



Background and Motivation

- **A "human-quality" macro placer must correctly comprehend**



Outline

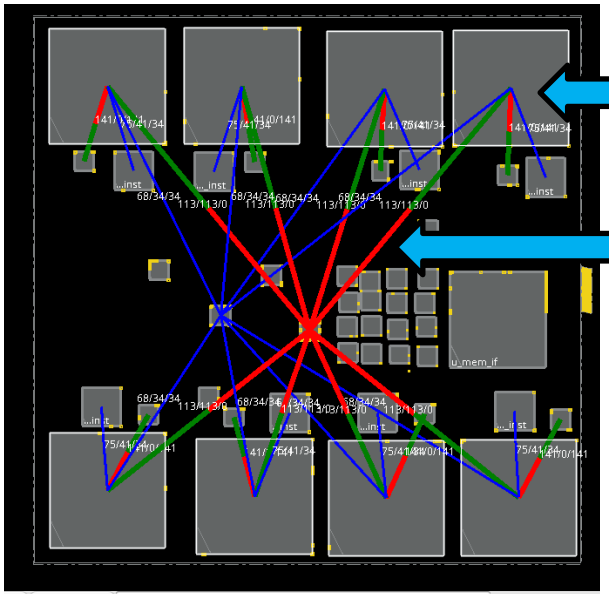
- Background and Motivation
- Main Contributions
- Our Methodology
- Experimental Setup and Results
- Conclusion and Future work

Main Contributions

- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**

RTL-MP

- *Conceptual abstractions devised by the frontend designer*
  - Through converting the structural netlist representation of the RTL design into a clustered netlist

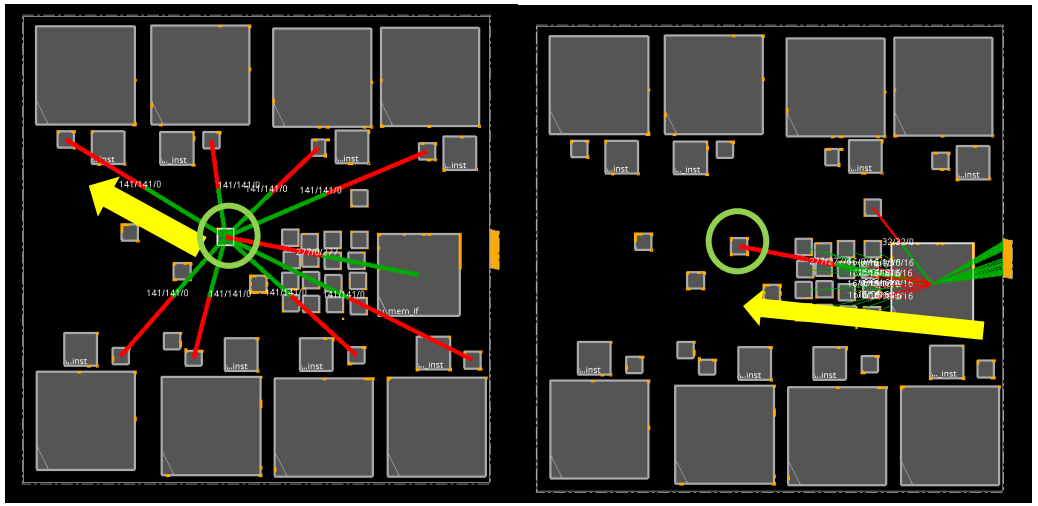


Main Contributions

- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**

RTL-MP

- *Functional interactions*
  - *Through analyzing the dataflow between logical modules (including Input-output pins)*



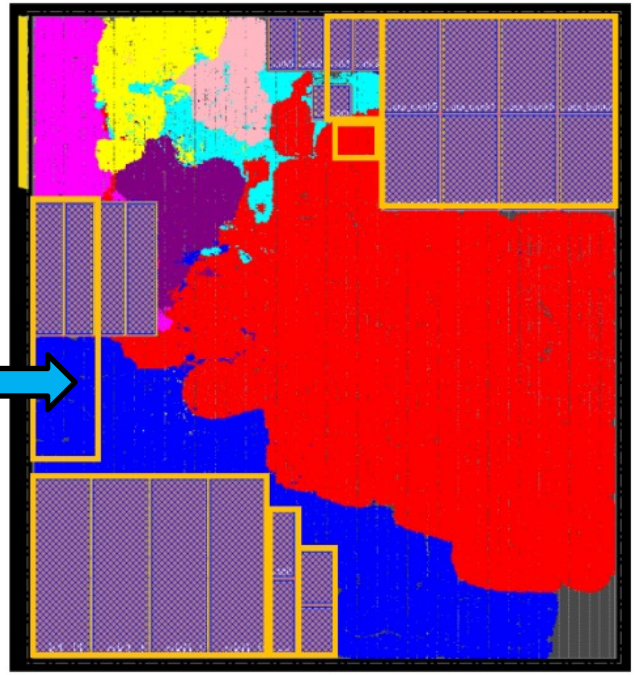
Main Contributions

- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**

RTL-MP

- *Macro placement guidance*
  - *Through pushing macros to their preferred locations*

macro guidance



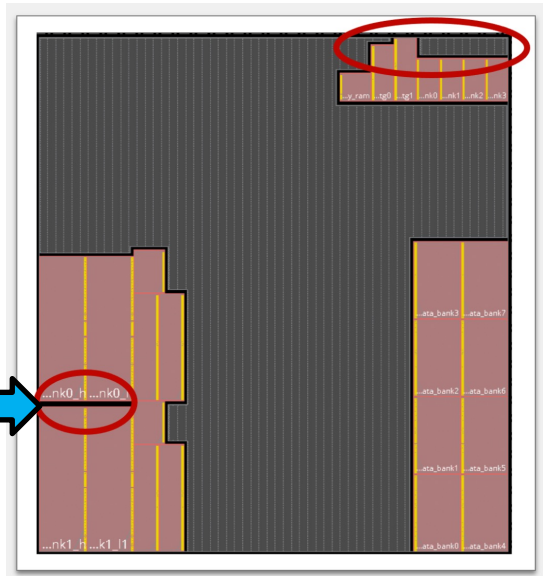
Main Contributions

- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**

RTL-MP

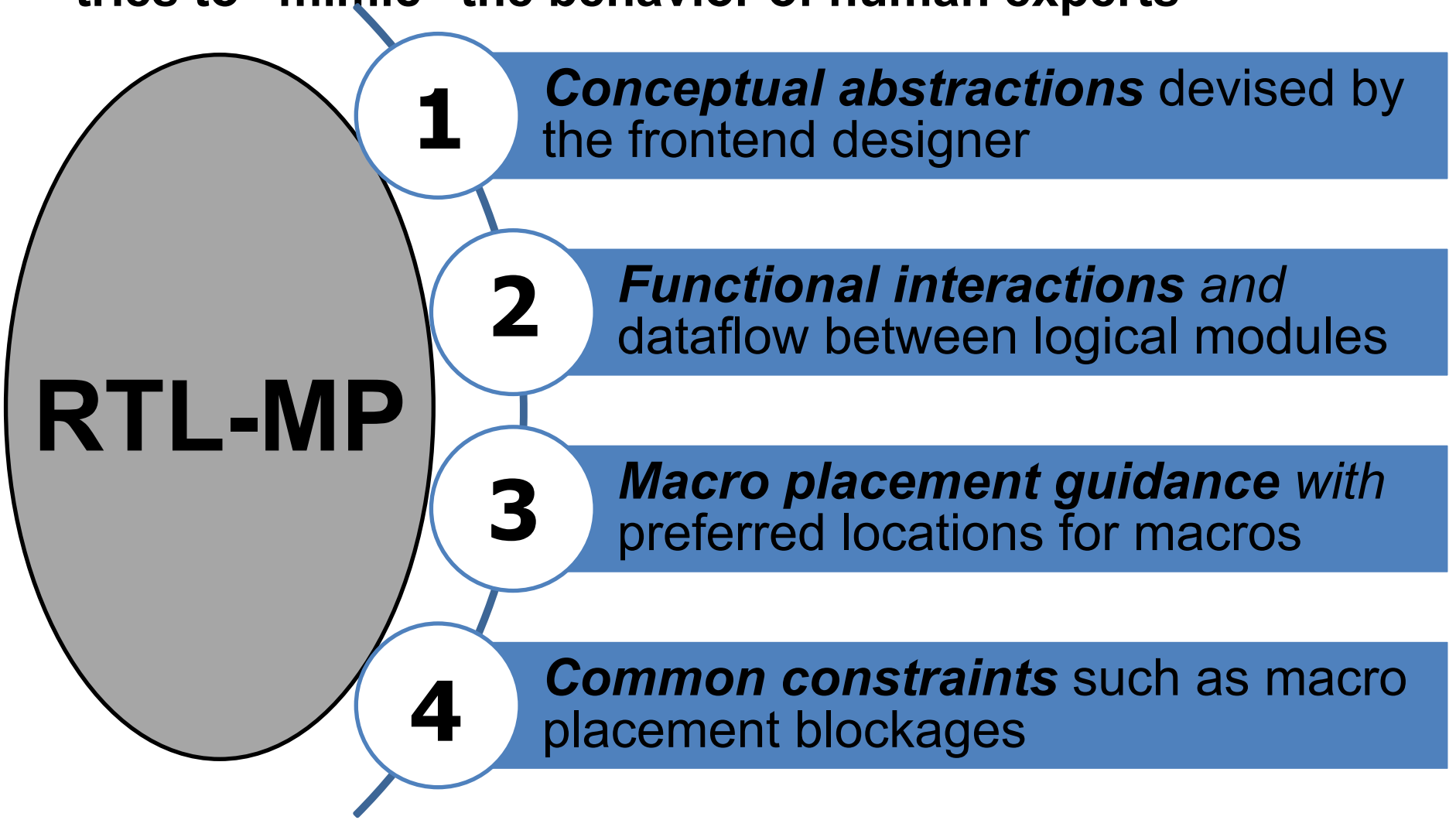
- *Common constraints*
  - *Fixed macro constraints*
  - *Macro placement blockages*
  - *Small dead space (notch) avoidance*
  - *Pushing macros to peripheries*

notch



Main Contributions

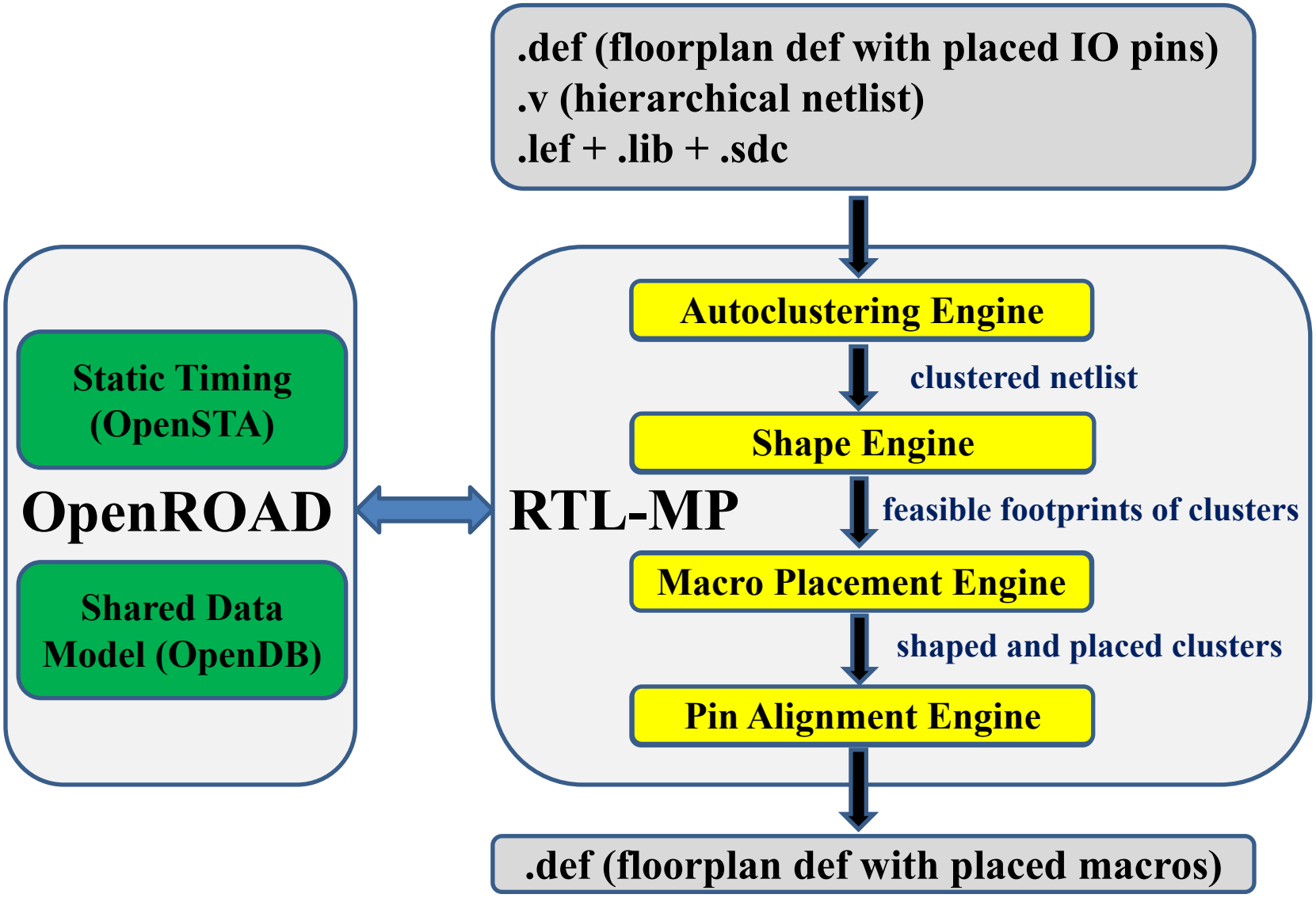
- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**



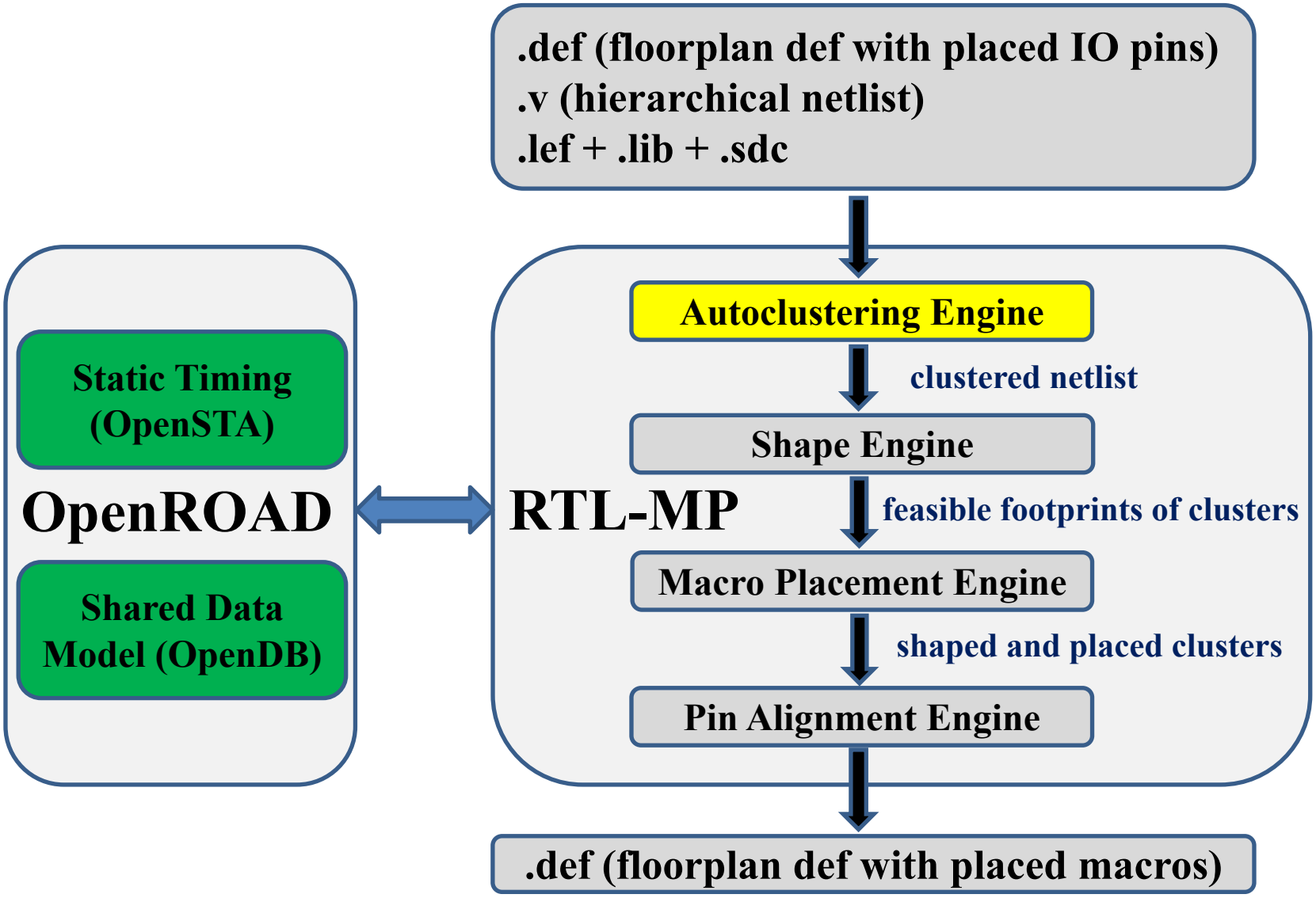
Outline

- Background and Motivation
- Main Contributions
- Our Methodology
- Experimental Setup and Results
- Conclusion and Future work

Our Methodology - RTL-MP Flow



Our Methodology - RTL-MP Flow

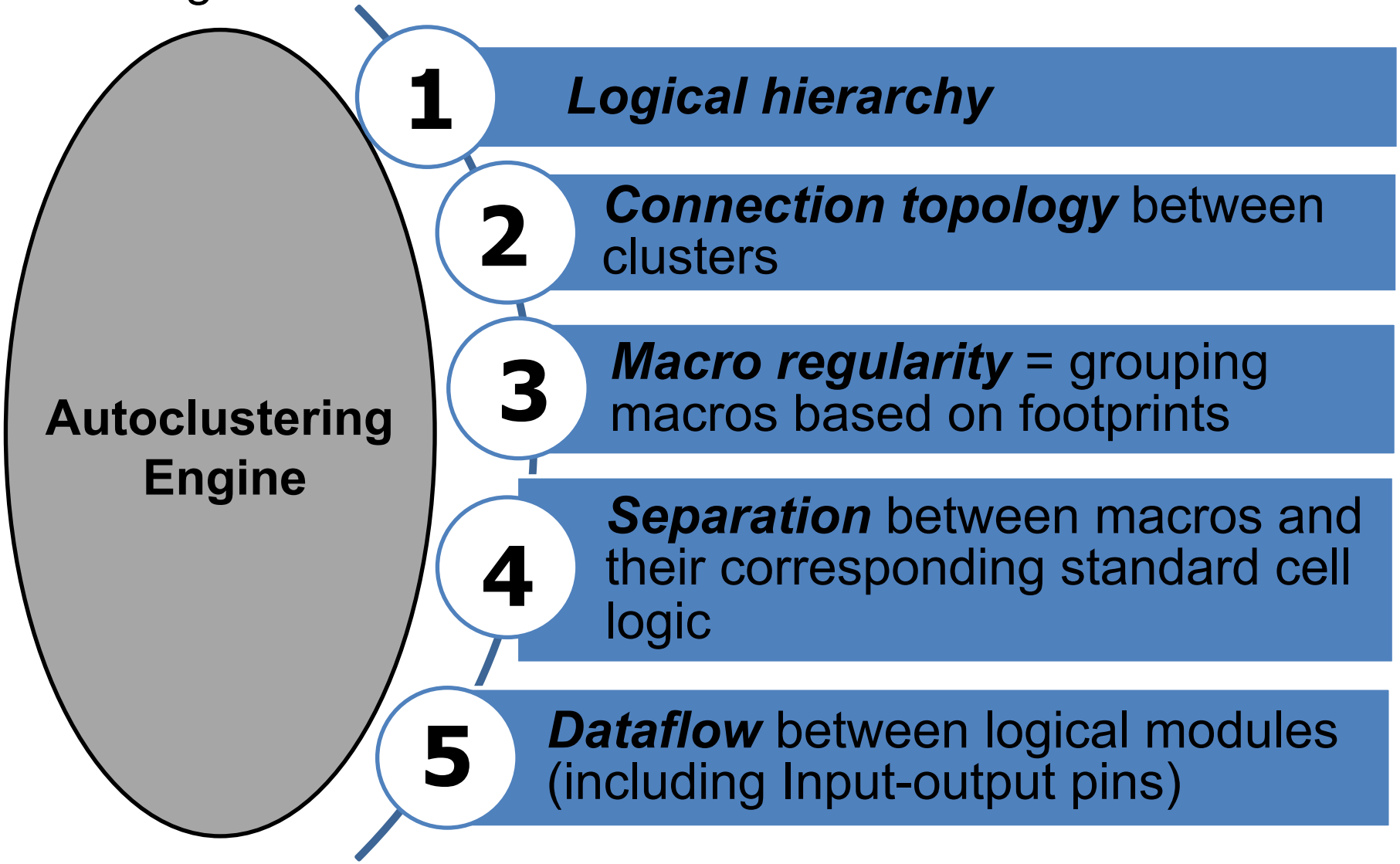


Autoclustering Engine

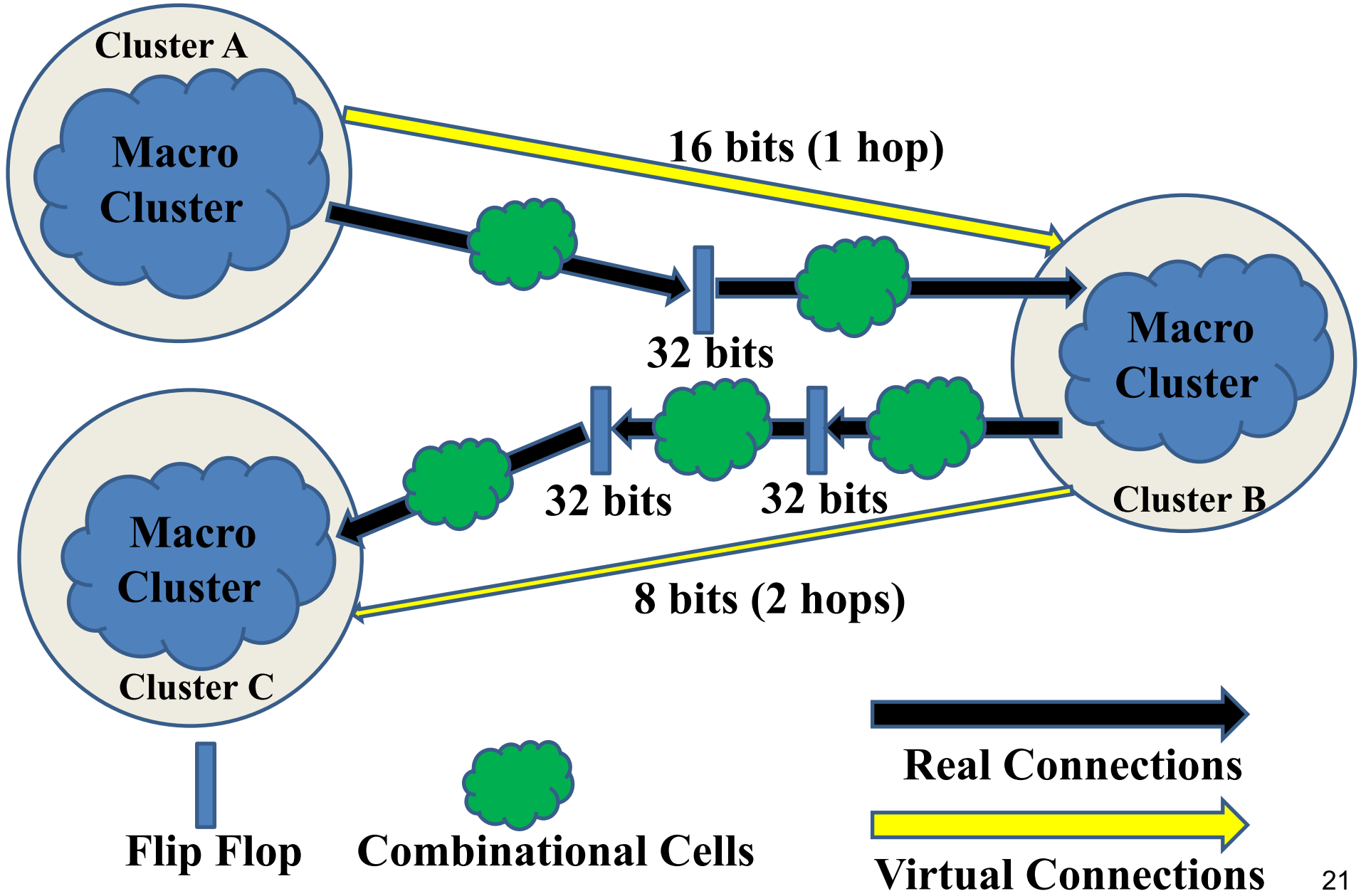
- Converts the structural netlist representation of the RTL design into a clustered netlist
- In the clustered netlist, we may have following two types of clusters
  - *Standard-Cell Cluster* containing only standard cells
  - *Macro Cluster* containing only macros

Autoclustering Engine

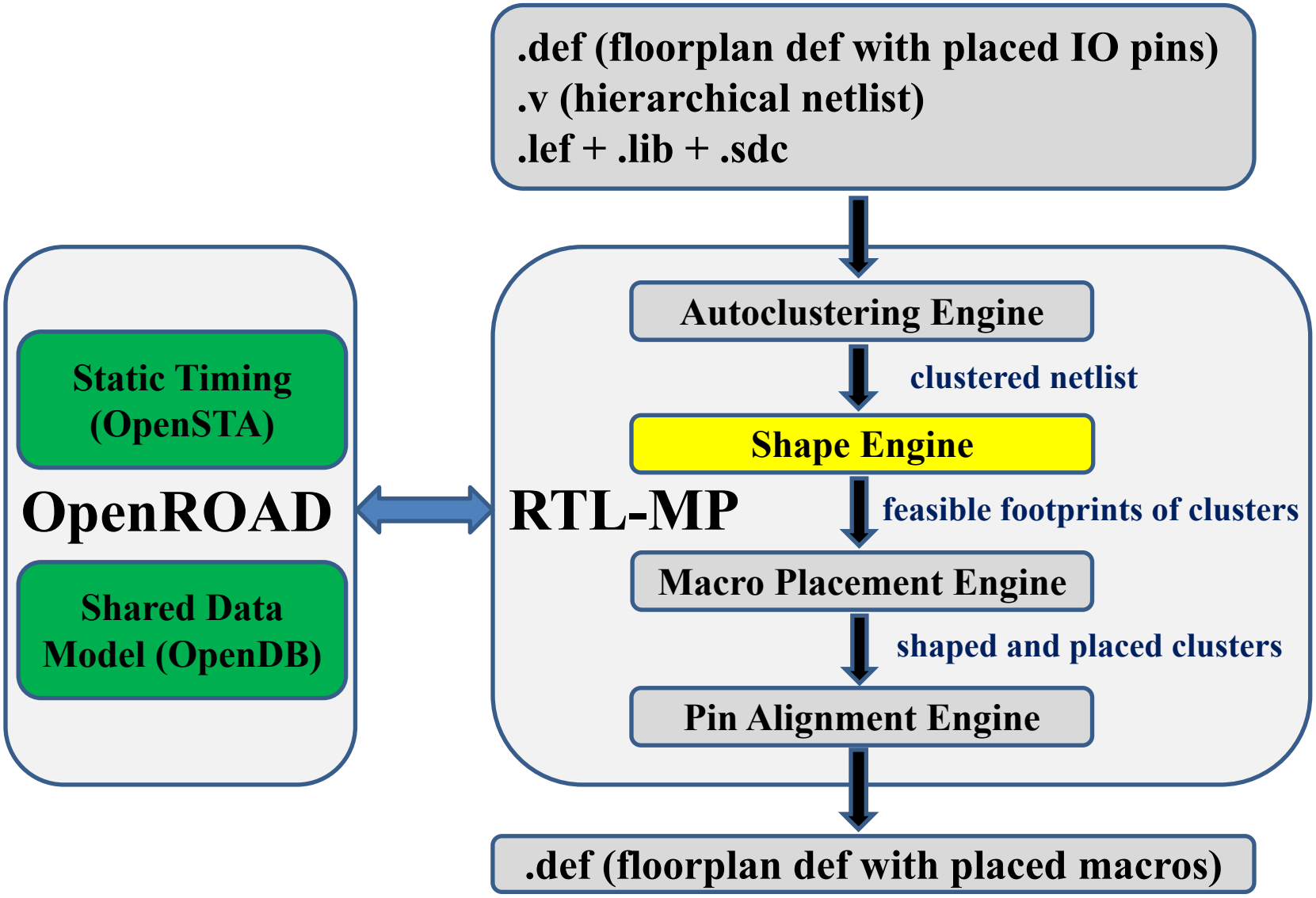
- Converts the structural netlist representation of the RTL design into a clustered netlist



Autoclustering Engine - Dataflow Analysis

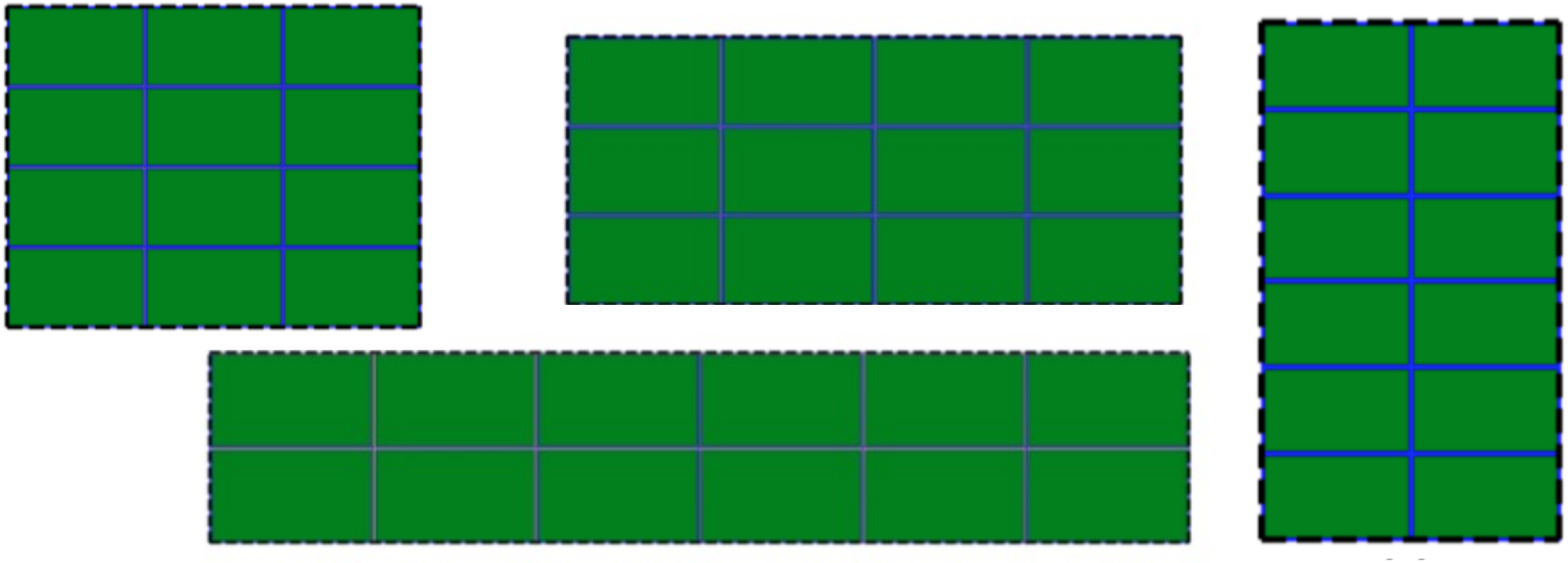


Our Methodology - RTL-MP Flow

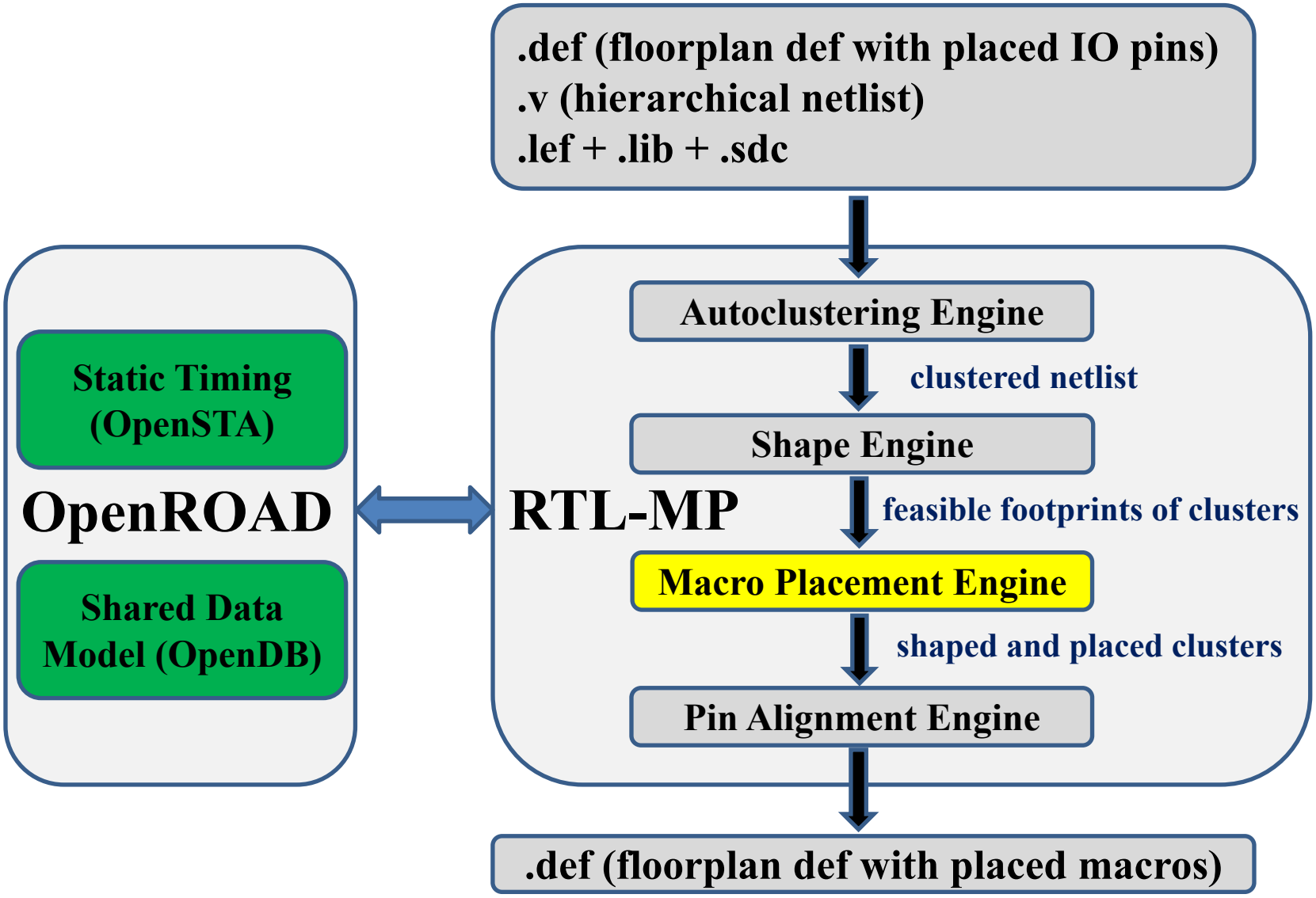


Shape Engine

- Determines possible shapes and aspect ratio constraints for all clusters
  - *Standard-Cell cluster* : the aspect ratio is specified by the upper bound (max_ar) and lower bound (min_ar)
  - *Macro cluster* : enumerate all the possible macro tilings and keep the macro tilings with minimum area



Our Methodology - RTL-MP Flow



Macro Placement Engine

- Determines position and shape for each cluster
- *Sequence Pair* to represent clusters in the netlist
- *Simulated Annealing* to optimize the cost function
- "*Go-with-the-winners*" scheme to improve the performance of Simulated Annealing

Macro Placement Engine

- **Handle multiple constraints**

1. *Fixed outline:* All clusters should be placed within the fixed outline specified by users
2. *Macro peripheral bias:* All macros should be pushed to peripheries as much as possible
3. *Macro blockage:* All macros should not overlap with macro placement blockages
4. *Preplaced macros:* Other macros should not overlap with preplaced macros
5. *Pin access*: All macros should be kept from blocking access of input-output pins.
6. *Macro guidance:* All macros should be placed near specified regions if users provide such constraints
7. *Notch avoidance:* A decent floorplan should avoid “dead space” which cannot be used effectively by P&R tools

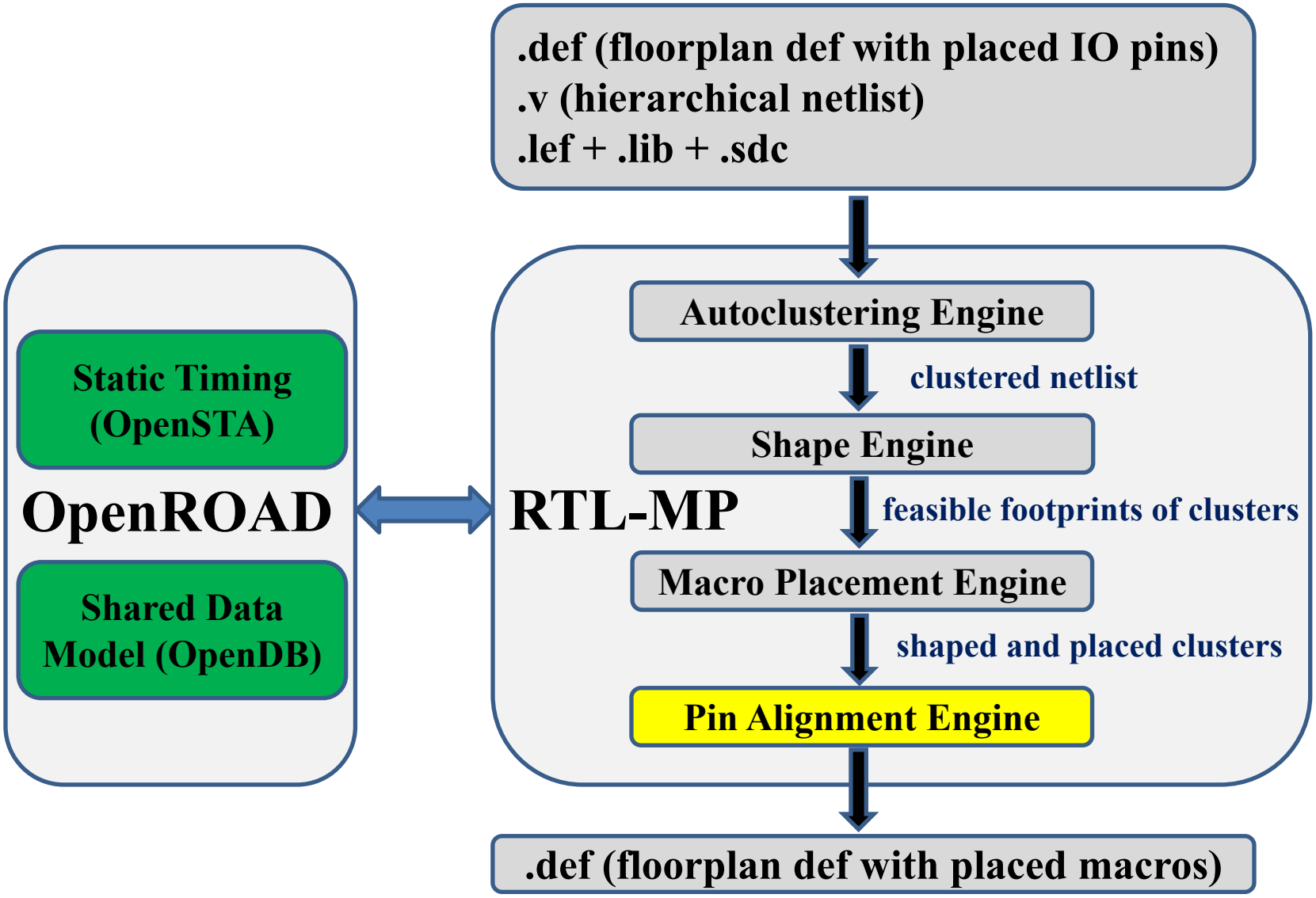
Macro Placement Engine

- **Handle multiple constraints**
  - Model each constraint as a term in the cost function

$$cost = \alpha \times Area + \beta \times WL + \gamma \times p_{outline} + \zeta \times p_{bias} + \eta \times p_{blockage} + \theta \times p_{guidance} + \lambda \times p_{notch}$$

  - Autotuning tool – Tune is used to tune the weights for different objectives in the cost function

Our Methodology - RTL-MP Flow



Pin Alignment Engine

- Finalizes the location and orientation of macros in each macro cluster, one macro cluster at a time
- For a given macro cluster
  - Other clusters behave like *fixed terminals*
  - *Simulated Annealing* to optimize the cost function
  - *"Go-with-the-winners"* scheme to improve the performance of Simulated Annealing
  - *Sequence Pair* to represent the macros in the cluster
    - Op1: Swap two macros in the first sequence
    - Op2: Swap two macros in the second sequence
    - Op3: Swap two macros in both sequences
    - *Op4: Flip all the macros*

Outline

- Background and Motivation
- Main Contributions
- Our Methodology
- Experimental Setup and Results
- Conclusion and Future work

Experimental Setup and Results

- **Macro placement scenarios**
  - *Comm:* Macro placement obtained by the 2020 release of a state-of-the-art commercial P&R tool using high-effort settings
  - *Manual:* All macros are manually placed by backend engineers
  - *TMP:* Macro placement is generated by TritonMP (default macro placer in the OpenROAD project)
  - *HiDaP:* Macros are placed by a recent state-of-the-art academic macro placer HiDaP
  - *RTL-MP:* Results are obtained using our macro placer
    - $RTL\text{-}MP_B$*:* Results are obtained using RTL-MP with user-specified macro blockages
    - $RTL\text{-}MP_L$ *and* $RTL\text{-}MP_T$ *:* Results are obtained using RTL-MP with "loose" and "tight" macro guidance separately

Experimental Setup and Results

- **Testcases (Implemented in GF12LP)**

Design	Std Cells	Macros	I/Os	Nets	Macro Blockage	Macro Guide
swerv_ wrapper	78K	28	1416	94K		
ariane	114K	37	495	131K		
simd	207K	46	4210	236K	😀	
coyote	208K	15	784	327K		
bp_single	323K	49	135	508K		
ca53	445K	25	1352	483K		😀

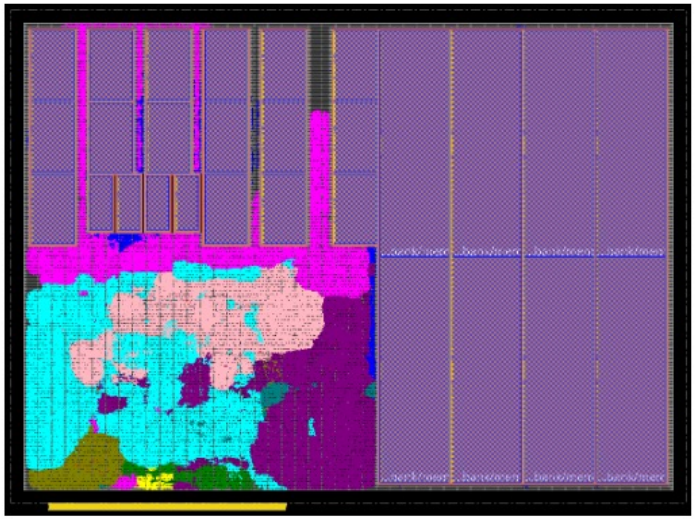
Experimental Setup and Results

- **Summary of post-routing metrics**

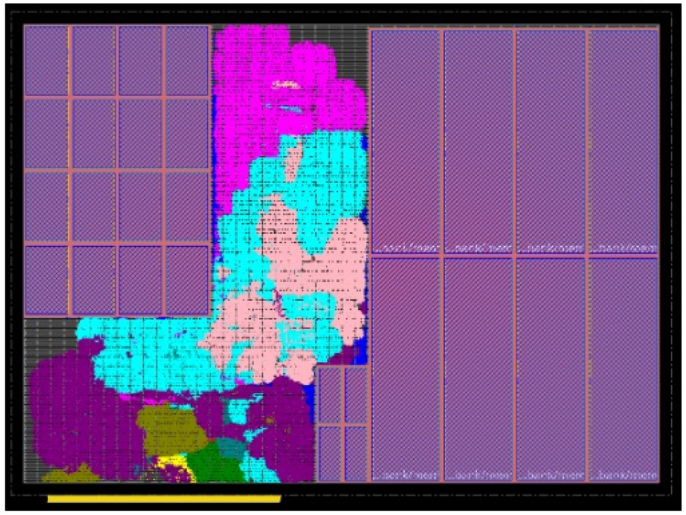
Design	Flow	Std Cells	WL (m)	DRC	WNS (ns)	TNS (ns)	Power (mW)	TAT (min)
swerv_wrapper	*Comm*	101K	1.48	0	-0.010	-0.352	116.75	1.00
	Manual	101K	1.38	0	-0.020	-0.497	115.75	weeks
	TMP	102K	1.46	1	-0.013	-0.200	118.66	1.53
	HiDaP	111K	1.39	0	-0.009	-0.511	115.55	1.68
	RTL-MP	100K	1.37	0	-0.018	-0.319	115.99	59.18
ariane	*Comm*	134K	2.27	0	-0.065	-12.269	174.52	1.50
	Manual	132K	1.89	0	-0.012	-0.678	168.98	weeks
	TMP	132K	1.86	0	-0.006	-0.314	169.44	2.18
	RTL-MP	132K	1.83	0	-0.012	-0.108	169.11	49.68
simd	*Comm*	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Manual	256K	4.14	0	-0.026	-1.195	288.70	weeks
	RTL-MP$_B$	255K	3.99	0	-0.032	-3.003	299.26	87.07
coyote	*Comm*	325K	3.22	0	-0.025	-0.094	110.14	2.00
	Manual	322K	3.17	0	-0.017	-0.129	109.84	weeks
	TMP	323K	3.27	0	-0.023	-0.222	110.17	2.92
	RTL-MP	322K	3.16	0	-0.035	-0.090	109.77	38.83
bp_single	*Comm*	523K	5.53	136	-0.105	-111.976	424.86	3.00
	Manual	512K	5.05	35	-0.086	-3.148	414.83	weeks
	TMP	507K	5.61	0	-0.086	-2.905	423.04	5.87
	RTL-MP	507K	4.95	0	-0.116	-3.202	414.09	22.08
ca53	*Comm*	508K	7.95	1	-0.047	-3.177	651.30	4.50
	Manual	505K	7.66	0	-0.014	-1.894	636.55	weeks
	TMP	507K	7.73	0	-0.021	-2.842	640.01	8.80
	RTL-MP$_G$	504K	7.47	1	-0.017	-2.887	630.35	71.55
	RTL-MP	503K	7.41	0	-0.015	-2.639	627.26	58.00
	RTL-MP$_L$	505K	7.50	0	-0.018	-2.985	636.06	57.15
	RTL-MP$_T$	504K	7.50	0	-0.021	-1.645	631.80	55.28

Experimental Setup and Results

- **Comparison between HiDaP and RTL-MP**



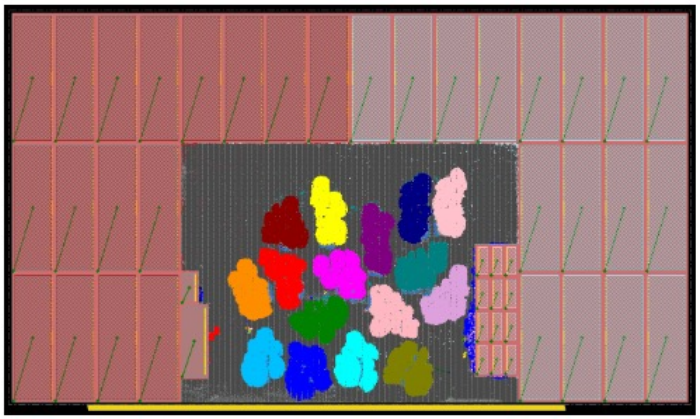
(A) swerv_wrapper (HiDaP)



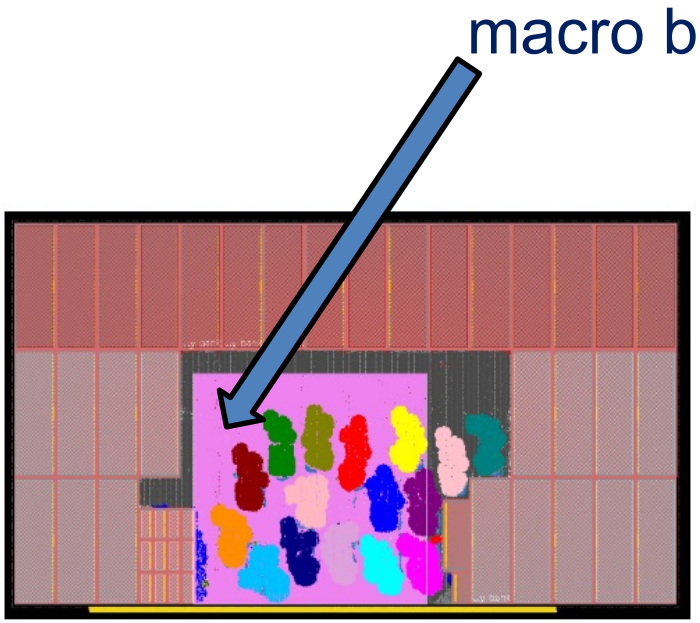
(B) swerv_wrapper (RTL-MP)

Experimental Setup and Results

- **The effect of macro blockage for RTL-MP**



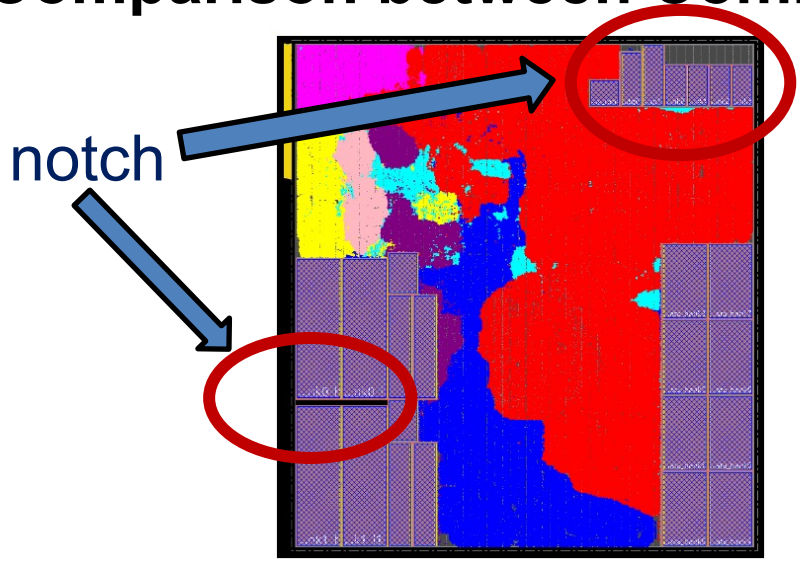
(C) SIMD (Manual)



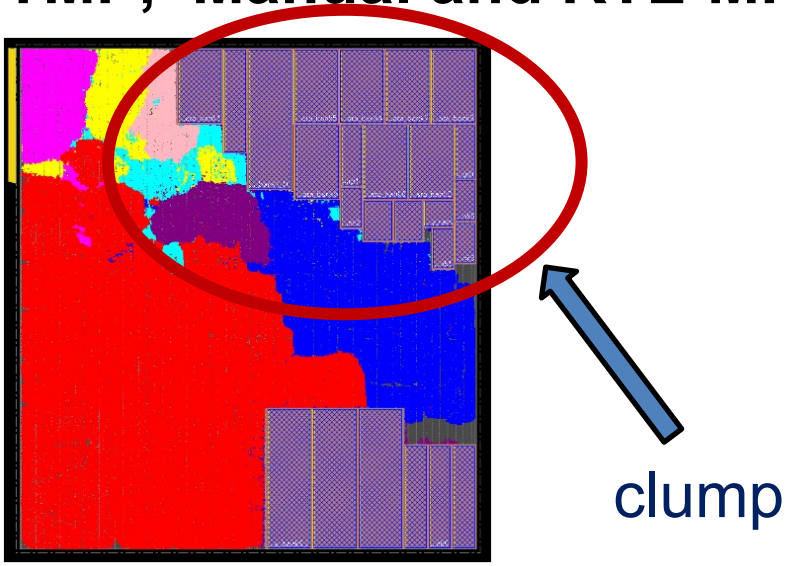
(D) SIMD ($RTL\text{-}MP_B$)

Experimental Setup and Results

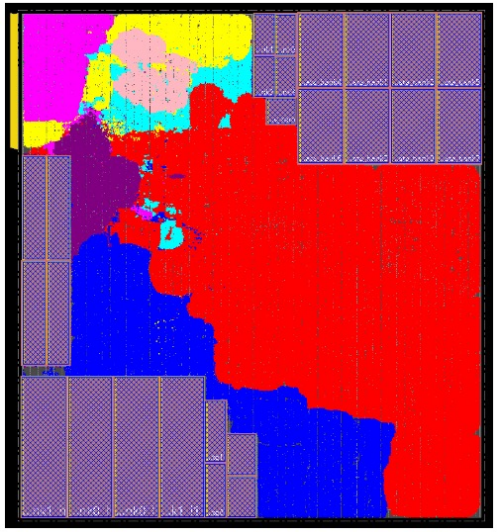
- **Comparison between Comm, TMP, Manual and RTL-MP**



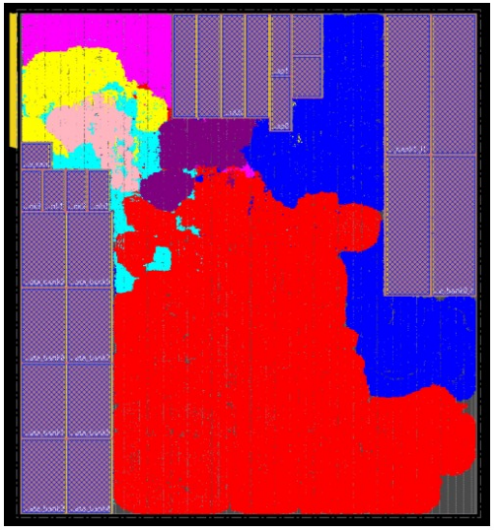
(E) CA53 (Comm)



(F) CA53 (TMP)



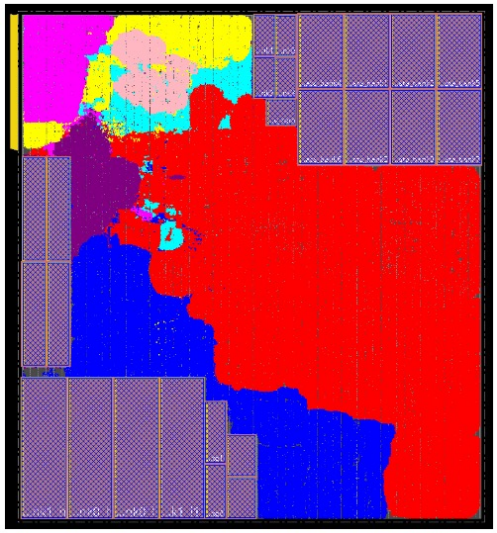
(G) CA53 (Manual)



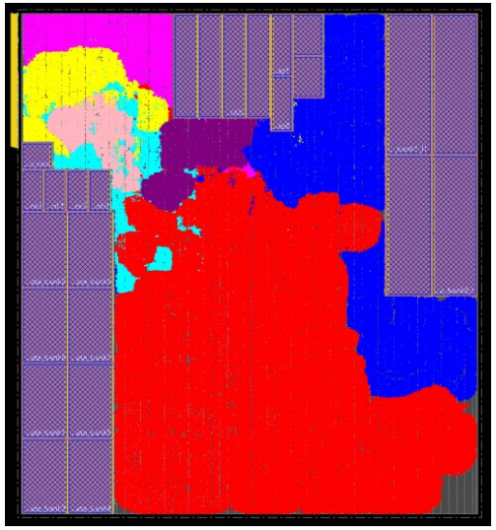
(H) CA53 (RTL-MP)

Experimental Setup and Results

- **The effect of macro guidance**



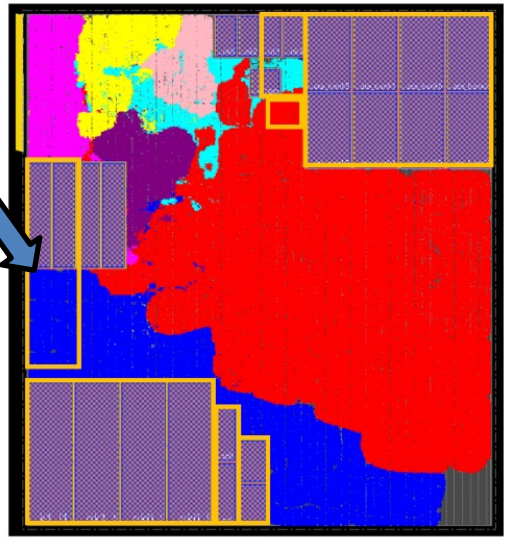
(G) CA53 (Manual)



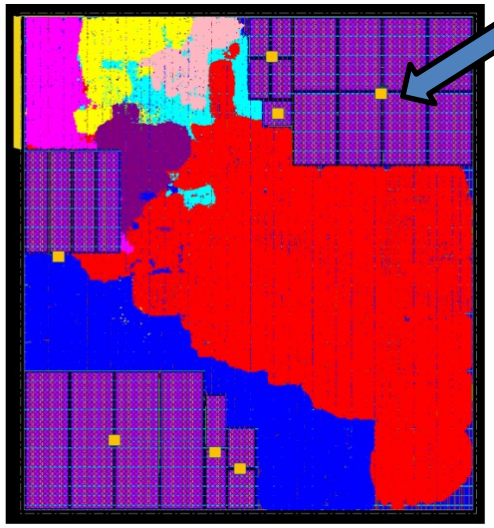
(H) CA53 (RTL-MP)

tight macro guidance

loose macro guidance



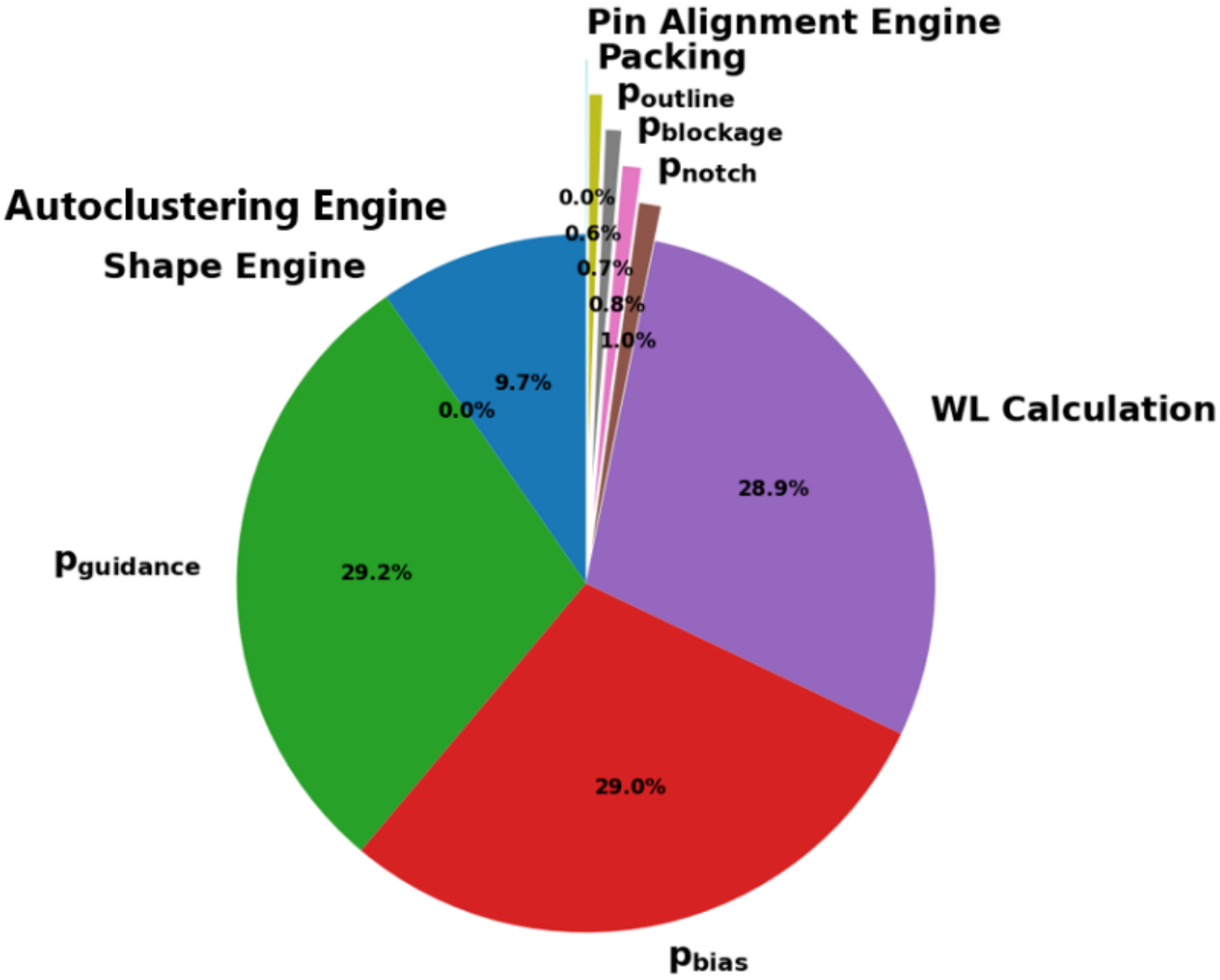
(I) CA53 ($RTL\text{-}MP_L$)



(J) CA53 ($RTL\text{-}MP_T$)

Experimental Setup and Results

- **Runtime profiling of RTL-MP on the ca53 testcase**

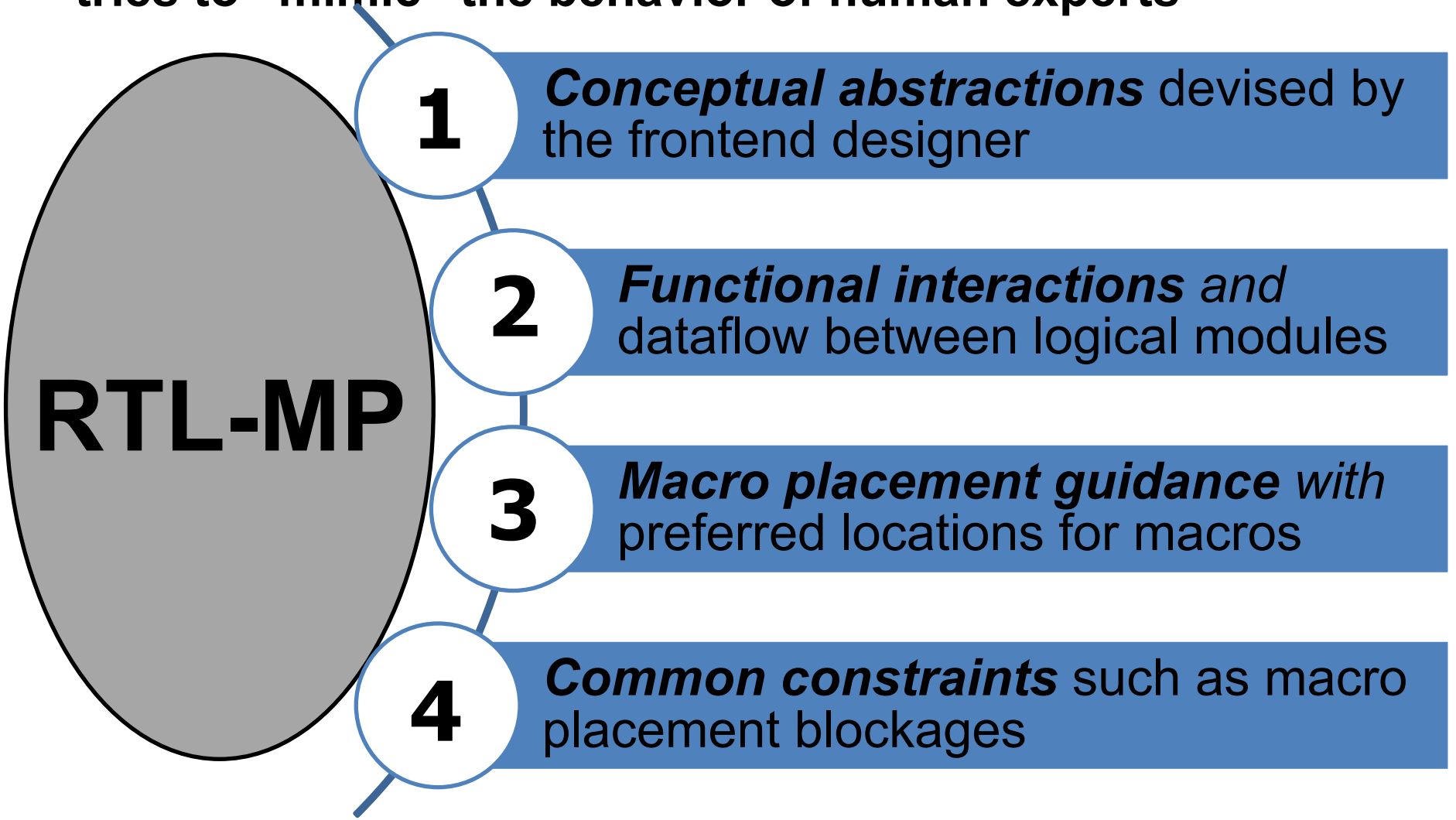


Outline

- Background and Motivation
- Main Contributions
- Our Methodology
- Experimental Setup and Results
- Conclusion and Future work

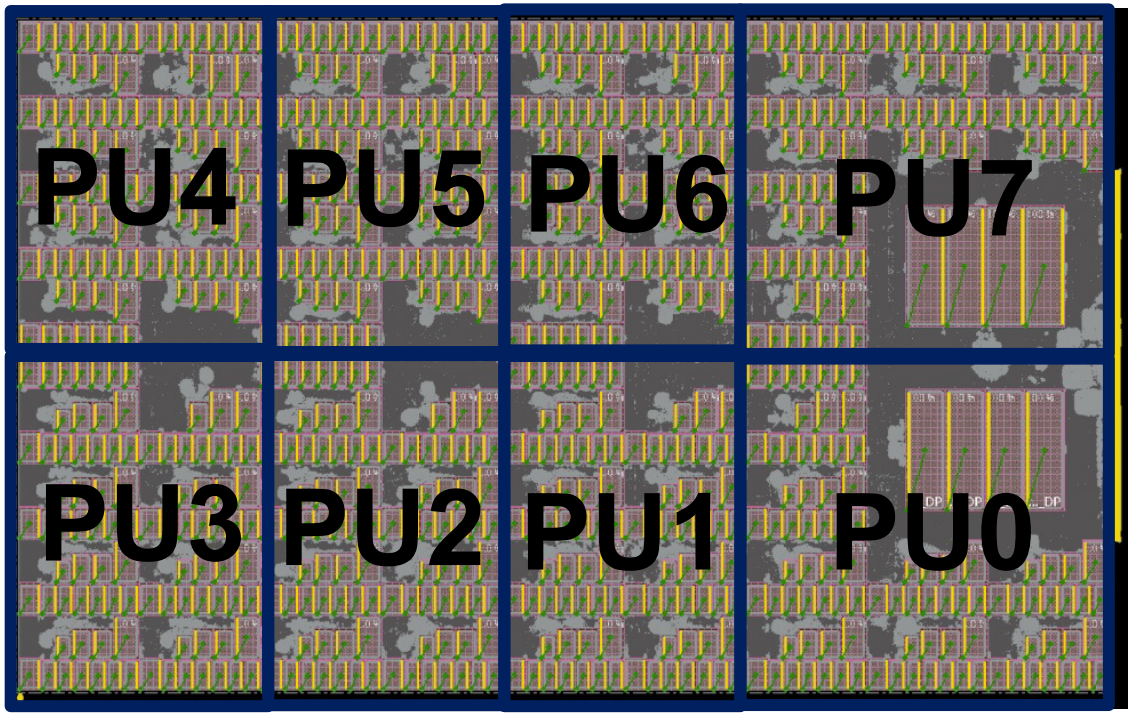
Conclusion

- **We propose RTL-MP, which utilizes RTL information and tries to "mimic" the behavior of human experts**



Future Work

- Reduce runtime
- Improve the autoclustering engine to support **hierarchical clustering**, so we can handle designs like TABLA shown below.



THANK YOU!