



JULY 9-13, 2023

**MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA**





OpenDRC: An Efficient Open-Source Design Rule Checking Engine with Hierarchical GPU Acceleration

Zhuolun He ♠♥ Yihang Zuo ♥ Jiayi Jiang ♥ Haisheng Zheng ♥
Yuzhe Ma ♣ Bei Yu ♠

♠ The Chinese University of Hong Kong

♥ Shanghai Artificial Intelligence Laboratory

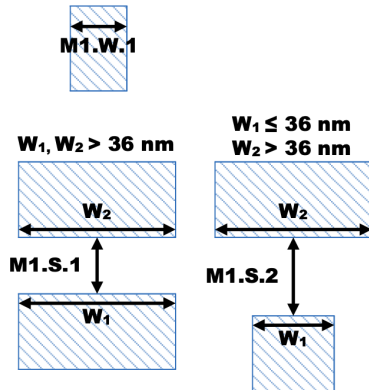
♣ HKUST (GZ)

Jul. 11, 2023



Design Rule Checking^[1]...

- Verifies a deck of layout constraints
- Consists of **Complex Rules** nowadays.
Geometric, Inter-Layer, Conditional Rules...
- Ultra **Time-Consuming** in the design flow.



^[1] Figure from ASAP7 Design Rule Manual.



Efficient DRC

There are several ways to improve DRC efficiency:

- **Better Algorithms/Data Structures**
 - Sweepline, Quad-Tree, R-Tree, ...
- **Parallel Computing**
 - Region-Based, Design Hierarchy, Edge-Based, Task Parallelism, ...
 - on SIMD Engines, Specialized Hardware, Distributed Systems, GPUs, ...
- **Approximation Methods**
 - Hotspot Detection, Violation Type/Number Prediction, ...
 - Other ML-Enhanced DRC Schemes.



Open-Sourcing Tools!

Open-source EDA tools have been inspiring and empowering the evolution of cutting-edge EDA research.



Open-Sourcing Tools!

Open-source 'Design Rule Checkers' often appear in

- Detailed Routers (e.g., TritonRoute^[2])
 - Not for verification purpose
 - Tightly coupled with path searching
- Layout Editors (e.g., Magic^[3], KLayout^[4])
 - GUI-Centric, not optimized for standalone checking
 - Different data structure for efficient editing

^[2] Andrew B Kahng et al. (2018). "TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies". In: *Proc. ICCAD*.

^[3] John K Ousterhout et al. (1985). "The Magic VLSI Layout System". In: *IEEE Design & Test*.

^[4] Matthias Köfferlein (2018). *KLayout*. <https://klayout.de>.



This Work: OpenDRC

We feel that a new (open-source) Design Rule Checking Engine is necessary!

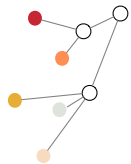
This work proposes OpenDRC, which

- Aims for Extremely **High Efficiency**.
- Supports **Hierarchical** Designs.
- Provides **GPU Acceleration**.
- is Available at <https://github.com/opendrc/opendrc>.

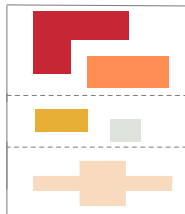


OpenDRC Overall Flow

Layout in BVH



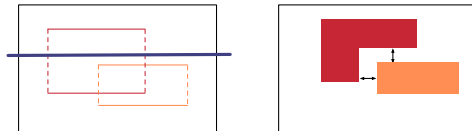
Rules from API



Adaptive partition



Sequential Mode

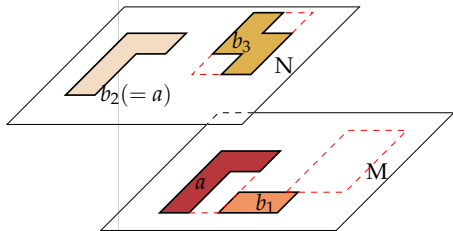


Parallel Mode



Hierarchical DRC

- Core data structure: Layer-wise *Bounding Volume Hierarchy* (BVH)
 - Does NOT flatten the layout; preserves design hierarchy
 - Maintains *minimum bounding rectangle* (MBR) of cells
- Redundant check elimination



- Some are trivial duplications: (a^M, b_1^N) and (b_1^M, a^N) when $M = N$
- Some results could be reused: (a^M, a^N) and (b_2^M, b_2^N) when $b_2 = a$
- Some violations cannot happen: (a^M, b_3^N)



Adaptive Layout Partition

Intuition (due to row-based placement):

- Layouts can be partitioned into non-overlapping regions (rows) along the y-axis
- By such grouping, x-coordinates of cells in a row are also likely to be separated

Solution: adaptive row-based partition

- Solvable as interval merging problem in $\Theta(k + N)$ or $\Theta(k \log k)$
- k is the number of cells, N is the number of unique coordinates ($k \gg N$)



Interval Merging

Require: A set S of intervals to be merged

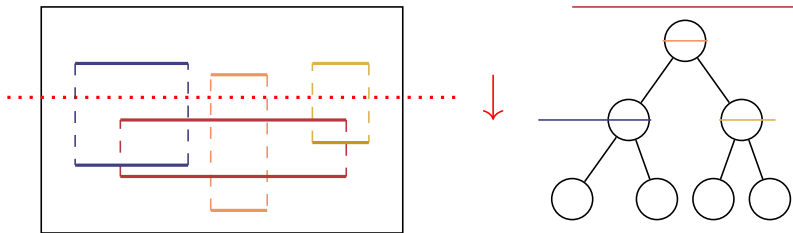
Ensure: Non-overlapping intervals covering the domain of S

- 1: Initialize an array A with indices
 - 2: **for all** interval $[l, r] \in S$ **do**
 - 3: Update $A[l] \leftarrow \max(A[l], r)$
 - 4: **end for**
 - 5: Initialize current interval end $e \leftarrow -1$
 - 6: **for** the i -th element $\in A$ **do**
 - 7: **if** $i > e$ **then**
 - 8: Create a new interval and reset e
 - 9: **end if**
 - 10: Update current interval end $e \leftarrow \max(e, A[i])$
 - 11: **end for**
- ▷ Step1: Initialize
▷ Step2: Merge
▷ Step3: Scan
▷ moving across interval boundary



Sequential Mode

- Standard Sweepline Algorithm^[5] (But we use Interval Tree)

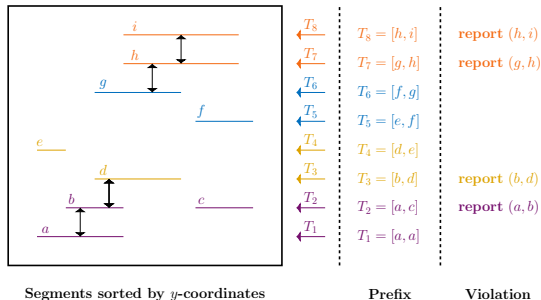


^[5] Jon Louis Bentley et al. (1980). "An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles". In: *IEEE TC*.



Parallel Mode w/ GPU Acceleration

- Parallel DRC row-by-row
- Edges of polygons are packed into a flattened array and transferred to device memory
- Parallel Sweepline [6]
 - 1 Parallel *scan* to determine check range.
 - 2 Parallel edge to edge(s) check.



[6] Zhuolun He et al. (2022). "X-Check: GPU-Accelerated Design Rule Checking via Parallel Sweepline Algorithms". In: *Proc. ICCAD*.



OpenDRC Software Architecture

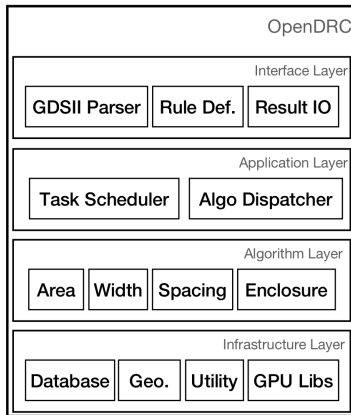


Figure: Overview of OpenDRC Software Architecture



General Programming Interface

```
1  auto db = odrcc::gdsii::read(/* path-to-gdsii */);
2  auto e  = odrcc::engine();
3  e.add_rules({
4      db.polygons().is_rectilinear(),
5      db.layer(19).width().greater_than(18),
6      db.layer(20).polygons().ensures(
7          [](const auto& p){return !p.name.empty();}
8      )
9  });
10 e.check(db);
```



Experimental Evaluation (Intra-Polygon Rules)

Design	Rule	KLayout			X-Check	OpenDRC		Rule	KLayout			X-Check	OpenDRC	
		flat	deep	tile		Seq.	Par.		flat	deep	tile		Seq.	Par.
aes	M1.W.1	3.45	12.69	0.49	0.41	0.02	0.03	M1.A.1	3.34	3.32	0.65	-	0.02	0.03
	M2.W.1	1.37	3.83	0.23	0.14	0.04	0.04	M2.A.1	1.35	1.33	0.37	-	0.04	0.04
	M3.W.1	2.52	2.98	0.36	0.11	0.03	0.03	M3.A.1	2.49	2.51	0.51	-	0.03	0.03
ethmac	M1.W.1	11.88	45.84	1.56	1.21	0.07	0.08	M1.A.1	11.55	11.55	2.05	-	0.07	0.08
	M2.W.1	3.76	10.72	0.52	0.42	0.10	0.11	M2.A.1	3.62	3.63	1.01	-	0.10	0.11
	M3.W.1	6.36	7.64	0.77	0.31	0.08	0.08	M3.A.1	6.20	6.24	1.24	-	0.08	0.08
gcd	M1.W.1	0.13	0.44	0.13	0.11	< 0.01	< 0.01	M1.A.1	0.13	0.13	0.13	-	< 0.01	< 0.01
	M2.W.1	0.05	0.08	0.05	< 0.01	< 0.01	< 0.01	M2.A.1	0.05	0.05	0.05	-	< 0.01	< 0.01
	M3.W.1	0.06	0.07	0.06	< 0.01	< 0.01	< 0.01	M3.A.1	0.06	0.06	0.06	-	< 0.01	< 0.01
ibex	M1.W.1	3.60	12.38	0.50	0.43	0.02	0.03	M1.A.1	3.52	3.52	0.65	-	0.02	0.03
	M2.W.1	1.30	3.61	0.24	0.14	0.03	0.04	M2.A.1	1.27	1.28	0.36	-	0.04	0.04
	M3.W.1	2.38	2.88	0.36	0.10	0.03	0.03	M3.A.1	2.36	2.35	0.51	-	0.03	0.03
jpeg	M1.W.1	13.32	55.35	1.68	1.39	0.08	0.08	M1.A.1	13.01	13.00	2.17	-	0.07	0.08
	M2.W.1	3.05	8.77	0.46	0.40	0.10	0.10	M2.A.1	2.98	2.95	0.95	-	0.09	0.09
	M3.W.1	4.86	6.14	0.59	0.29	0.08	0.08	M3.A.1	4.79	4.81	1.10	-	0.08	0.07
sha3	M1.W.1	3.48	12.36	0.49	0.43	0.02	0.03	M1.A.1	3.40	3.40	0.63	-	0.02	0.03
	M2.W.1	1.10	2.95	0.21	0.12	0.03	0.03	M2.A.1	1.07	1.09	0.33	-	0.03	0.03
	M3.W.1	1.79	2.15	0.30	0.09	0.02	0.02	M3.A.1	1.79	1.77	0.42	-	0.02	0.02
uart	M1.W.1	0.15	0.40	0.15	0.11	< 0.01	< 0.01	M1.A.1	0.14	0.14	0.15	-	< 0.01	< 0.01
	M2.W.1	0.06	0.12	0.06	< 0.01	< 0.01	< 0.01	M2.A.1	0.06	0.06	0.06	-	< 0.01	< 0.01
	M3.W.1	0.08	0.09	0.08	< 0.01	< 0.01	< 0.01	M3.A.1	0.08	0.08	0.08	-	< 0.01	< 0.01
Average		37.7×	82.1×	9.6×	4.5×	0.9×	1.0×		37.6×	37.6×	13.0×	-	1.0×	1.0×



Experimental Evaluation (Inter-Polygon Rules)

Design	Rule	KLayout			X-Check	OpenDRC		Rule	KLayout			X-Check	OpenDRC	
		flat	deep	tile		Seq.	Par.		flat	deep	tile		Seq.	Par.
aes	M1.S.1	4.33	13.78	0.62	0.17	0.21	0.06	V1.M1.EN.1	468.24	462.28	15.97	0.20	6.44	0.12
	M2.S.1	1.55	4.15	0.29	0.13	0.09	0.02	V2.M2.EN.1	2.93	1.64	0.59	0.14	0.18	0.09
	M3.S.1	2.64	3.25	0.38	0.12	0.15	0.02	V1.M2.EN.2	469.96	468.89	15.71	0.20	0.24	0.12
ethmac	M1.S.1	14.67	48.50	1.89	0.39	0.72	0.14	V1.M1.EN.1	3045.02	3038.10	57.76	2.00	42.35	0.41
	M2.S.1	4.35	11.71	0.59	0.20	0.23	0.05	V2.M2.EN.1	8.29	4.74	1.45	0.23	0.47	0.22
	M3.S.1	6.68	8.17	0.82	0.16	0.39	0.04	V1.M2.EN.2	3031.20	3034.67	55.63	0.36	0.84	0.32
gcd	M1.S.1	0.15	0.46	0.14	0.11	< 0.01	0.01	V1.M1.EN.1	3.06	2.96	3.09	0.11	0.06	< 0.01
	M2.S.1	0.05	0.09	0.05	0.11	< 0.01	< 0.01	V2.M2.EN.1	0.07	0.05	0.08	0.10	< 0.01	< 0.01
	M3.S.1	0.06	0.07	0.06	0.11	< 0.01	< 0.01	V1.M2.EN.2	2.95	2.95	2.99	0.10	< 0.01	< 0.01
ibex	M1.S.1	4.45	13.15	0.63	0.17	0.22	0.06	V1.M1.EN.1	477.86	473.62	16.03	0.21	7.14	0.13
	M2.S.1	1.49	3.96	0.25	0.13	0.09	0.02	V2.M2.EN.1	2.78	1.56	0.56	0.15	0.18	0.08
	M3.S.1	2.50	3.08	0.39	0.12	0.14	0.02	V1.M2.EN.2	479.79	477.17	15.90	0.17	0.24	0.12
jpeg	M1.S.1	15.82	57.36	2.01	0.43	0.80	0.16	V1.M1.EN.1	3609.55	3580.46	58.29	1.59	55.07	0.49
	M2.S.1	3.48	9.79	0.49	0.21	0.20	0.05	V2.M2.EN.1	7.07	4.04	1.22	0.22	0.40	0.20
	M3.S.1	5.17	6.70	0.64	0.16	0.30	0.03	V1.M2.EN.2	3611.69	3588.04	57.01	0.35	0.87	0.32
sha3	M1.S.1	4.23	13.02	0.60	0.16	0.21	0.06	V1.M1.EN.1	476.10	472.44	15.87	0.49	7.07	0.12
	M2.S.1	1.16	3.23	0.22	0.12	0.07	0.02	V2.M2.EN.1	2.32	1.29	0.48	0.13	0.14	0.07
	M3.S.1	1.87	2.31	0.30	0.11	0.11	0.02	V1.M2.EN.2	468.70	467.92	17.28	0.15	0.22	0.11
uart	M1.S.1	0.19	0.44	0.19	0.11	< 0.01	0.01	V1.M1.EN.1	3.61	3.50	3.62	0.10	0.06	< 0.01
	M2.S.1	0.07	0.13	0.07	0.11	< 0.01	< 0.01	V2.M2.EN.1	0.10	0.06	0.10	0.12	< 0.01	< 0.01
	M3.S.1	0.08	0.10	0.08	0.10	< 0.01	< 0.01	V1.M2.EN.2	3.49	3.48	3.54	0.10	< 0.01	< 0.01
Average		47.6×	99.5×	12.0×	5.6×	3.2×	1.0×		514.9×	429.0×	61.5×	2.9×	4.7×	1.0×



Runtime Breakdown for Sequential Spacing Check

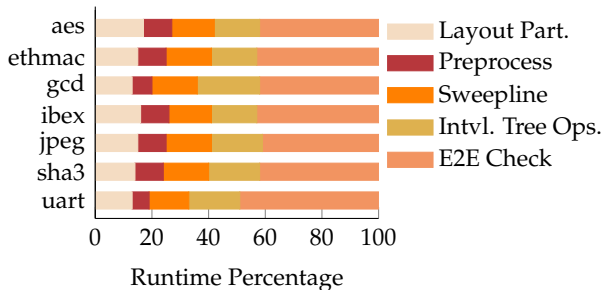


Figure: The Runtime Breakdown of OpenDRC Sequential *Minimum Spacing* Checks.

- 'Layout Part' refers to Adaptive Layout Partitioning;
- 'Intvl. Tree Ops.' refers to Interval Tree Operations insert, remove, and query;
- 'E2E Check' refers to Edge-To-Edge Checks.



Summary

We develop **OpenDRC**, a new open-source Design Rule Checking Engine

- Adaptive row-based layout partition.
- Efficient sequential/parallel hierarchical DRC procedures.
- Significant speedup compared with SOTA multi-threading/GPU design rule checkers.

Future work:

- Systematic Evaluation of heterogeneous computing in DRC.
- Data Compression techniques for memory footprint reduction.
- Supports for general geometric shapes.





THANK YOU!

