IEEE/ACM
2023 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN
ICCAD
42nd Edition

Heterogeneous Acceleration for Design Rule Checking

Zhuolun He, **Bei Yu**

Department of Computer Science & Engineering
The Chinese University of Hong Kong

Nov. 01, 2023

香港中文大學
The Chinese University of Hong Kong

Outline

CPU-GPU Paradigm is Promising

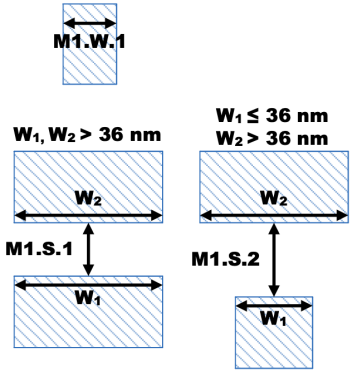
37,888 GPUs (8,335,360 cores) + 9,472 CPUs(606,208 cores) $\Rightarrow$ one ExaFLOPS (10^{18})

GPU-ccelerated EDA

- Placement
  - Global Placement: DreamPlace [DAC'19], X-Place [DAC'22]
  - Detailed Placement: ABCDPlace [TCAD'20]
- Routing
  - pattern routing: FastGR [TCAD'22]
  - maze routing: GAMER [TCAD'22]
  - Steiner tree construction [ICCAD'22]
- Static Timing Analysis [ICCAD'20] [DAC'21]
- gate-level logic simulation [DAC'22], circuit simulation [DAC'21], logic optimization [DAC'22], capacitance extraction [DATE'13]
- ...

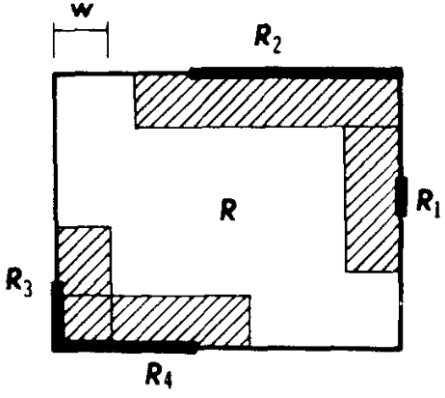
This Talk: Design Rule Checking[1]

- verifies a deck of layout constraints
- consists of complex rules nowadays
  - geometric, inter-layer, conditional rules...
- is ultra time-consuming in the design flow

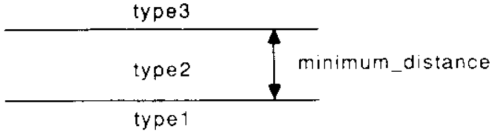


[1]Figure from ASAP7 design rule manual

Algebraic Design Rule Checking



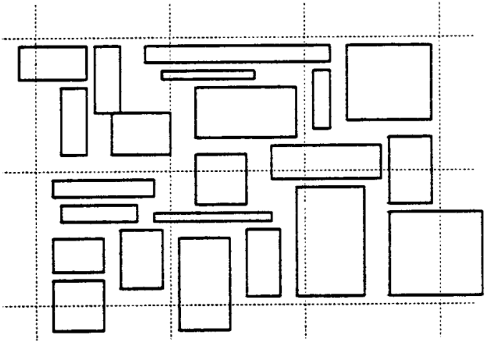
Set-based



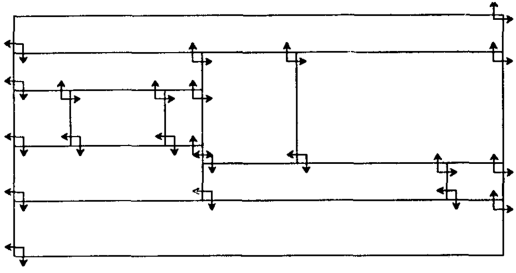
Edge-based

Example: if type1 and type 3 are 'outside', type2 is 'inside', it represents width check

Layout Data Structures



Binning



Corner Stitching [TCAD'84]

Parallel DRC

- Various parallelism
  - region-based, hierarchy-based, edge-based, $\cdots$
  - task parallelism, data + task
- Various platforms
  - SIMD, multiprocessors, GPU, specialized hardware, $\cdots$

		Multiprocessor	GPU	Hardware	Distributed
Data-	Region-				[VLSID'94] [DAC'11] [CS'92]
	Hierarchy-	[ICPP'84]	[ICCAD'22] [DAC'23]	[DAC'84] [VLSID'20]	
	Edge-	[DAC'88]			
Task-					[TR'86]
Task- and Data-		[JPDC'96]			

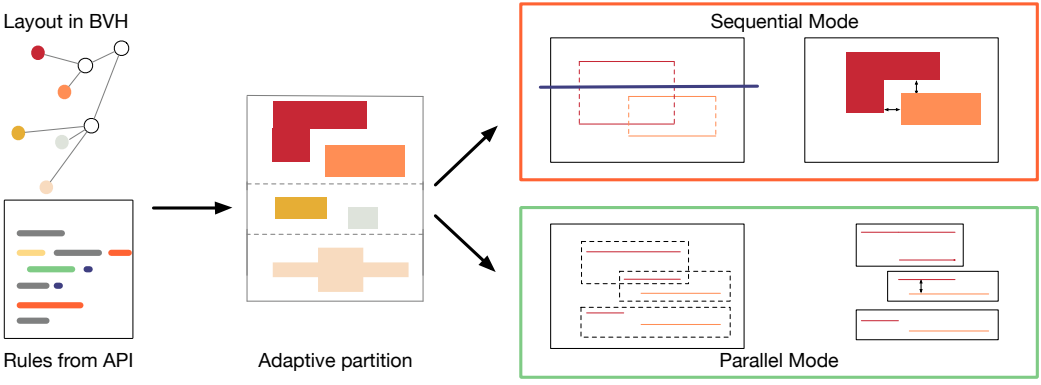
This Work: OpenDRC

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



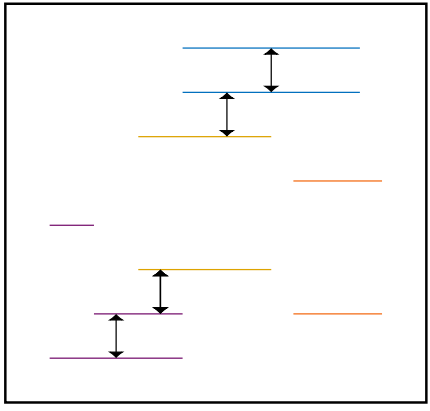
Problem Formulation

(We only consider horizontal edges.)

Problem (Distance Check)

Given a set $\mathcal{H}$ of horizontal segments in $\mathbb{R}^2$, report the segment pairs from $\mathcal{H}^2$ whose horizontal projection is nonempty, and vertical distance is smaller than δ. Formally, we want to report:

$$\{([l_1, r_1] \times y_1, [l_2, r_2] \times y_2) \in \mathcal{H}^2\}$$
$$s.t.\ [l_1, r_1] \cap [l_2, r_2] \neq \emptyset, |y_1 - y_2| < \delta$$



Parallel prefix sums

$$a[] = (4, 5, 3, 6, 2, 5, 1, 1, 0)$$

Suppose we have 3 threads.

1. Batching: each thread computes sums of 3 consecutive elements.

$$s = (?, ?, 12, ?, ?, 13, ?, ?, 2)$$

2. Sweeping: sweep the partial sums

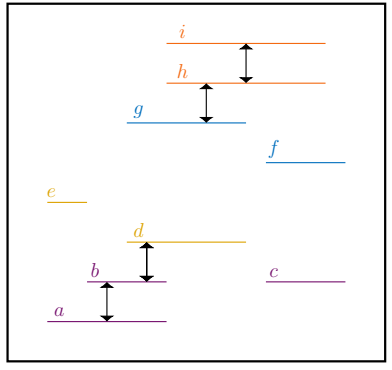
$$s = (?, ?, 12, ?, ?, 25, ?, ?, 27)$$

3. Refining: compute other prefix sums

$$s = (4, 9, 12, 18, 20, 25, 26, 27, 27)$$

Vertical Sweeping

- Key idea: the prefix structure contains a set $\mathcal{S}$ of segments that are below current segment within δ in y-direction
- Remains to check if each pair of segments overlap in the x-direction



	Prefix	Violation
T_8	$T_8 = [h, i]$	report (h, i)
T_7	$T_7 = [g, h]$	report (g, h)
T_6	$T_6 = [f, g]$	
T_5	$T_5 = [e, f]$	
T_4	$T_4 = [d, e]$	
T_3	$T_3 = [b, d]$	report (b, d)
T_2	$T_2 = [a, c]$	report (a, b)
T_1	$T_1 = [a, a]$	

Segments sorted by y-coordinates

Heterogeneous Acceleration

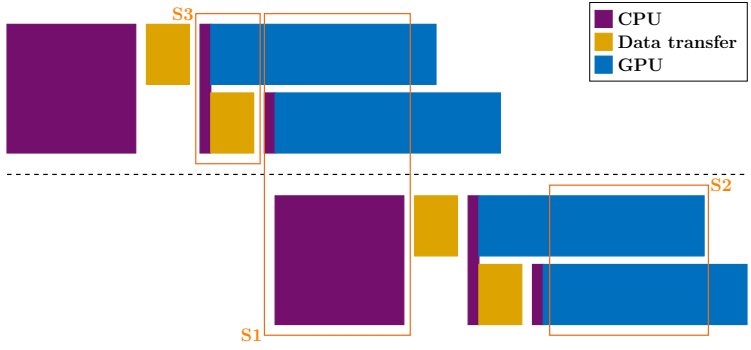
General strategies:

- Concurrent GPU computation and CPU computation.
- Concurrent GPU computation between streams.
- Overlap data transfer and computation.
- Minimize data transfer overhead.
- Avoid GPU invocation for small data batch.

Techniques in OpenDRC

Background: In OpenDRC, layout is adaptively partitioned into rows.

- S1: GPU computation of the previous row and CPU preprocessing of the next row can be executed concurrently.
- S2: GPU computation for horizontal edges and vertical edges can be executed concurrently by different streams.
- S3: Data movement for one batch of data and GPU sorting of another can be overlapped.



Techniques in OpenDRC

Background: In OpenDRC, layout is adaptively partitioned into rows.

- S1: GPU computation of the previous row and CPU preprocessing of the next row can be executed concurrently.
- S2: GPU computation for horizontal edges and vertical edges can be executed concurrently by different streams.
- S3: Data movement for one batch of data and GPU sorting of another can be overlapped.
- S4: Differentiate horizontal and vertical edges.
- S5: No GPU computation will be invoked if a row has only a limited number of objects.

Result

Design	Size	Rows	CPU	GPU	GPU\S1	GPU\S2	GPU\S3	GPU\S4	GPU\S5	GPU\all
aes	294052	277	2128	189	187	193	213	192	186	202
ethmac	1007152	507	14318	436	441	455	430	520	440	534
ibex	303004	277	2404	187	194	197	194	202	193	212
jpeg	1182541	537	19692	455	465	480	458	526	503	575
sha3	301382	277	2298	180	178	192	177	185	191	217
Average			19.22	**1.00**	1.01	1.05	1.03	1.09	1.04	1.18

- Removing each strategy: 1% to 9% speed-down
- Still 16.3× faster than CPU when removing all five

Summary

- Heterogeneous acceleration is promising
- Review efforts for efficient DRC
  - Algebraic
  - Layout data structures
  - parallel DRC
- Heterogeneous acceleration for DRC

THANK YOU!