A Novel Framework for DTCO: Fast and Automatic Routability Assessment with Machine Learning for Sub-3nm Technology Options

Chidi Chidambaram[‡], Andrew B. Kahng[†], Minsoo Kim[†,*], Giri Nallapati[‡], S.C. Song[‡] and Mingyu Woo[†]
[†]UC San Diego, La Jolla, CA, USA, [‡]Qualcomm Technologies Inc., San Diego, CA, USA (*E-mail: mik226@ucsd.edu)

Abstract

We report, for the first time, block-level area scaling that is reversed from cell-level scaling in cell height <150nm regime, using a machine learning (ML)-assisted design-technology co-optimization (DTCO) methodology to optimize block-level area accounting for routability. >400 unique standard-cell architectures are studied, combining cell height (CH, 80∼150nm), contacted poly pitch (CPP, 39∼57nm), metal pitch (MP, 16∼30nm) and use/non-use of buried power rails (BPR). Cell- and design-level routability assessments are performed to obtain routability index (a.k.a. K_{th}) and utilization of designs. CH <120nm with four available tracks increases block-level area due to routing difficulty, showing diminishing return from further pushing of ground rules. BPR improves block-level scaling with CH <120nm, but benefit slows down with CH <100nm. A 1:2 gear ratio (MP:CPP) improves block-level area compared to a 2:3 ratio. This DTCO flow is automated, and the ML-based prediction expedites the process.

Introduction

Design-technology co-optimization (DTCO) is an essential, high-value element of technology enablement. However, DTCO has proved to be a very expensive process; large turnaround times and engineering efforts are needed to develop standard-cell libraries and perform comprehensive block implementation experiments. Turnaround time of DTCO at advanced nodes is weeks to months with hundreds of engineers. Meanwhile, standard cells have been optimized to aim for minimum cell area (cell height) for decades. However, cell-level area scaling does not necessarily result in block-level area scaling at advanced nodes. As standard-cell area is scaled down, local routing and pin accessibility of standard cells become significantly difficult. Various *scaling boosters*, such as buried power rails (BPR) [7], backside PDN and supervias, are proposed to improve power, performance, area and cost (PPAC). Scaling boosters also have an impact on routability. Due to significant routability impacts of standard-cell architectures and scaling boosters, block-level area must be estimated at an early stage of technology development.

Recently, DTCO methods for technology definition have been proposed in many aspects. [1] proposes a machine learning (ML)-based approach to find optimal combinations of design, technology and other ingredients for high-performance CPU designs. However, there is no automatic design enablement stage (i.e., standard-cell library generation), and the methodology requires four to six weeks of turnaround time. [8] proposes an ML-based modeling framework for devices. The framework can generate compact models of novel devices without prior knowledge. On the other hand, the work does not consider block-level evaluations.

In this work, we apply the PROBE2.0 framework [2] to evaluate various technology options and configurations for sub-3nm technology nodes. We generate 448 unique standard-cell architectures, combining cell height (*CH*, 80∼150nm), contacted poly pitch (*CPP*, 39∼57nm), metal pitch (*MP*, 16∼30nm) and use/non-use of buried power rails (BPR). We study achievable block-level area with automatically generated standard-cell libraries across four main axes: (i) available M2 routing tracks (RT) on standard cells, (ii) use of BPR, (iii) gear ratios for MP to CPP, and (iv) different available M2 RT with same CH. We also apply ML-assisted routability prediction to expedite assessment.

DTCO Framework and Configuration for Sub-3nm Nodes

In this section, we summarize PROBE2.0 [2], the framework used for routability assessment in DTCO, along with the configurations for sub-3nm standard-cell libraries that we study. As shown in Fig. 1(a), the PROBE2.0 framework includes automatic standard-cell layout generation anddesign enablement generation, and (cell- and design-level) routability assessments. The basic idea of PROBE2.0 is to measure inherent routability, represented by K_{th} metric, through neighbor-swap operations applied to canonical placements. Fig. 2 illustrates the neighbor-swap operation. For a given cell, we randomly choose a neighboring cell and swap the locations of the cell pair. Neighbor-swaps progressively increase routing difficulty by "tangling" the placement. We let K denote the number of neighbor-swap operations, normalized to the number of instances (standard-cells) in the design. As K increases, the number of post-routing design rule check violations (#DRCs) likely increases; we define K_{th} as the maximum K for which #DRCs is less than a prescribed threshold. PROBE2.0 advances over the previous PROBE [5] with a satisfiability modulo theory (SMT)-based "automatic" standard-cell layout generation [3][6] and an ML-based K_{th} prediction, as illustrated in Fig. 1(b).

In this work, we generate nine standard-cell libraries to study routability and achievable block-level cell density at sub-3nm technology nodes. Table II shows sets of parameters for the nine libraries, as follows. (i) Lib{1,2,5,6} have 4 RT while Lib{3,4,7,8} have 5 RT. (ii) Lib{1,3,5,7,9} have BPR for power and ground pins while Lib{2,4,6,8} have M1 pins. (iii) Lib{1,2,3,4} have a 1:2 ratio for MP to CPP (20nm MP and 40nm CPP) and Lib{5,6,7,8} have a 2:3 ratio for MP to CPP (26nm MP and 39nm CPP). (iv) Lib{3,9} have 120nm CH but Lib3 has 5 RT and Lib9 has four tracks according to MP of Lib{3,9}. Fig. 3 illustrates OAI21_X1 gates in the nine libraries.

Routability Assessment and Block-level Area Case Study

We perform cell- and design-level routability assessments to evaluate generated standard cells. Cell-level assessment is used to assess inherent routability for each cell in a standard-cell library. Design-level assessment, i.e., at the level of an entire place-and-route block, is used to evaluate a set of standard cells in the same library. We also perform case study of cell-level and achievable block-level area. Cell height of a standard-cell library is linearly related to cell-level area. However, cell-level area benefit does not always result in block-level area benefit, due to routability effects. Our case study shows how we can evaluate technology options with respect to block-level area density using the PROBE2.0 framework.

Fig. 4 shows results for the *cell-level* routability assessments. We study 15 standard cells with size X1, in each of four libraries (Lib{1,2,3,4}). Results in Fig. 4(c) show that standard cells with M1 have better routability than those with BPR, and standard cells with five tracks have better routability than those with four tracks. Further, standard cells with larger numbers of input pins have worse routability than those with smaller numbers of input pins. Figs. 4(a) and (b) show #DRCs vs. K plots for NAND and OAI gates, respectively.

Fig. 5 shows results for *design-level* routability assessments with four designs, AES, LDPC, JPEG and VGA. Fig. 5(b) shows K_{th} metric per standard-cell library per design while Fig. 5(a) shows #DRCs vs. K plots for the AES design. Fig. 6 shows results of our achievable utilization study for AES and LDPC designs. We observe strong positive correlation between K_{th} and achievable utilization.

In addition, we perform case study for block-level area benefits across four main axes of technology options.

- Case 1: Available M2 routing tracks (RT) (Lib1 vs. Lib3).
- Case 2: Use of buried power rails (Lib1 vs. Lib2).
- Case 3: Gear ratios for MP to CPP (Lib1 vs. Lib5).
- Case 4: Different RT with the same CH (Lib3 vs. Lib9).

In Case 1, Lib3 has 20% larger cell area than Lib1. However, based on the achievable utilization (0.71 and 0.92 for Lib1 and Lib3, respectively), the achievable block-level area with Lib1 is 7% larger than with Lib3. It is important to note that in this case, reduction of RT brings less cell-level area, but the block-level area actually increases. In Case 2, Lib2 has M1 PGpin while Lib1 has BPR, so that Lib2 also has 20% larger cell area than Lib1. Applying the same calculation as with Case 1, we obtain that the achievable area with Lib2 is 19% greater than with Lib1. Thus, in Case 2, the cell-level area benefit from use of BPR is reflected as a design-level area benefit. In Case 3, Lib1 has a 1:2 gear ratio for MP to CPP and Lib5 has a 2:3 gear ratio. Since Lib5 has 26nm MP, Lib5 has 30% more cell area than Lib1. But, block-level area of Lib5 is 14% larger than that of Lib1 since the respective Lib1 and Lib5 achievable utilizations are 0.71 and 0.62. Last, in Case 4, Lib9 has the same CH (120nm) as Lib3, but MP of Lib9 is 24nm instead of 20nm. Larger MP brings benefits of reduced resistance, but Lib3 incurs 15% area penalty at design-level. Table III summarizes our case study. As shown in Table III and Fig. 6(b), block-level area no longer changes linearly with CH at advanced nodes. Even when CH increases, block-level area can decrease.

Machine Learning (ML)-Assisted Routability Assessment

PROBE2.0 [2] uses ML-based K_{th} prediction to reduce the number of place-and-route (P&R) implementation runs needed for accurate routability assessment. However, large training sets (e.g., 80% of libraries) are used. In this work, we apply Latin hypercube sampling (LHS) [4] to select an asymptotically more efficient training set for ML-based K_{th} prediction. In a case study, we create 448 standard-cell libraries as follows: (i) CPP = 39, 42, 45, 48, 51, 54, 57nm; (ii) MP = 16, 18, 20, 22, 24, 26, 28, 30nm for M1 and M2; (iii) 4 and 5 RT; (iv) BPR and M1 PGpin; and (v) M3 pitch:CPP ratios of 1:2 and 2:3. We choose training sets of 64, 128 and 256 data points (i.e., standard-cell library and K_{th}) by LHS and use all remaining (resp. 384, 320, 192) data points for model testing. Fig. 7 shows comparisons for golden

K_{th} values (which are extracted from P&R experiments) and predicted K_{th} values from our ML-based prediction. Figs. 7(a) and (b) show the golden and predicted K_{th} comparisons when 128 and 320 data points are used for training and testing, respectively. Fig. 7(c) shows the error distribution for the testing dataset. For the three LHS-based training set sizes (64, 128, 256) we obtain 3.07, 3.00, 2.90 average K_{th} prediction error in testing.

Conclusion

A machine learning (ML)-assisted design-technology co-optimization (DTCO) framework [2] is applied to study sub-3nm node cell and block-level DTCO. We perform cell- and design (block)-level routability assessments for >400 unique standard-cell architectures by varying technology options along axes of CPP, MP, RT, and use/non-use of BPR. The regime of CH <120nm with four available tracks shows increasing block-level area due to routing difficulty, indicating diminishing return on efforts to push ground rules. BPR improves block-level scaling with CH <120nm, but scaling benefits slow down with CH <100nm. A 1:2 gear ratio (MP:CPP) improves block-level area compared to a 2:3 ratio. The proposed DTCO method will extend to evaluate power and performance along with the cell- and block-level area density assessment presented in this study.

Acknowledgments

We thank Chung-Kuan Cheng, Hayoung Kim, Dongwon Park and Daeyeal Lee for their collaboration in the work of [2].

References

[1] A. Ceyhan *et al.*, *IEEE IEDM*, 2019, pp. 36.6.1-36.6.4.
[2] C.-K. Cheng *et al.*, *IEEE Trans. on CAD*, In revision. https://bit.ly/3pVX9sB
[3] C.-K. Cheng *et al.*, *Proc. ISCAS*, 2020, doi: 10.1109/ISCAS45731.2020.9180592.
[4] M. D. McKay *et al.*, *Technometrics* 21(2), pp. 239–245, doi:10.2307/1268522.
[5] A. Kahng *et al.*, *IEEE Trans. on CAD 37(7) (2018)*, pp. 1459-1472.
[6] D. Lee *et al.*, *IEEE Trans. on CAD*, doi: 10.1109/TCAD.2020.3037885.
[7] D. Prasad *et al.*, *Proc. IEDM*, 2019, pp. 19.1.1-19.1.4.
[8] Z. Zhang *et al.*, *SNW*, 2019, doi: 10.23919/SNW.2019.8782897.
[9] H2O AutoML. https://www.h2o.ai

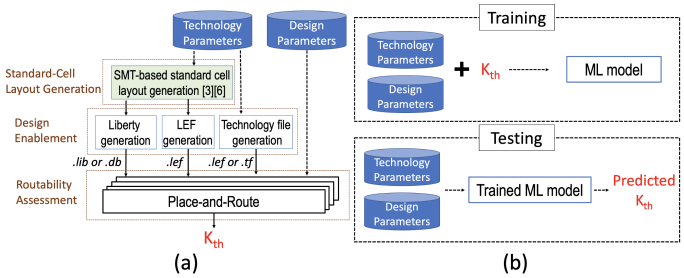

Fig. 1: (a) Overall flow of PROBE2.0 [2] and (b) machine learning-based K_{th} prediction.

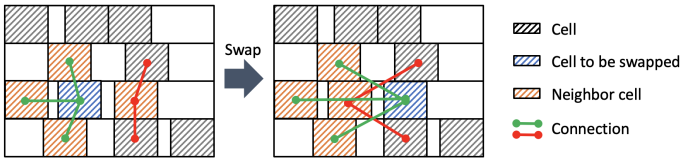

Fig. 2: An example of neighbor-swap operation. The blue-colored cell is swapped with one of the red-colored neighboring cells.

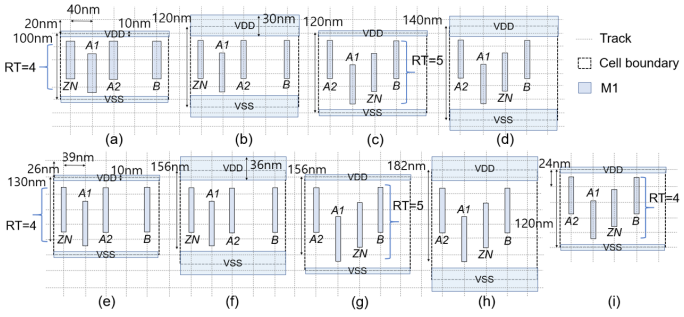

Fig. 3: Standard-cell layout of the nine standard-cell libraries in Table II. OAI21_X1 of Lib1 to Lib9 are illustrated in (a) to (i), respectively. We show a LEF view (cell boundary and pin) of layout of OAI21_X1 per library.

TABLE I: Technology and design parameters in this work. We use all the parameter types defined in [2].

Param. Type	Name	Option	Name	Option
Technology	Fin	2, 3	PGpin	BPR, M1
	CPP	39, 40	CH	5, 6, 7
	MP	20, 24, 26	MPO	3
	RT	4, 5	DR	EUV-Loose
Design	BEOL	13M	Tool	Tool A
	PDN	Sparce	Util	0.6
	Design	Knight's tour, AES, LDPC, JPEG, VGA		

TABLE II: Technology parameters for the nine standard-cell libraries which we study in this work.

Name	Fin	CPP	MP	RT	PGpin	CH (T)	CH (nm)
Lib1	2	40	20	4	BPR	5T	100
Lib2	2	40	20	4	M1	6T	120
Lib3	3	40	20	5	BPR	6T	120
Lib4	3	40	20	5	M1	7T	140
Lib5	2	39	26	4	BPR	5T	130
Lib6	2	39	26	4	M1	6T	156
Lib7	3	39	26	5	BPR	6T	156
Lib8	3	39	26	5	M1	7T	182
Lib9	2	40	24	4	BPR	5T	120

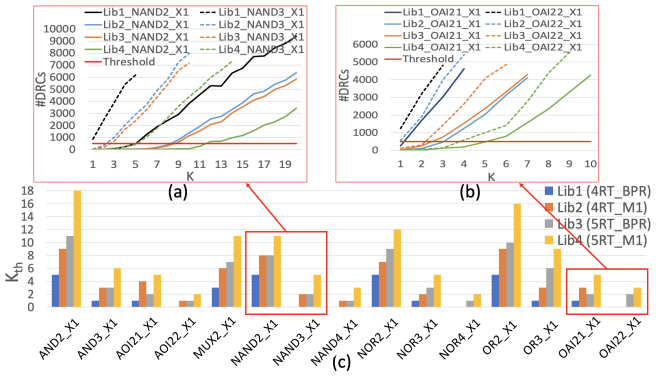

Fig. 4: Results for cell-level routability assessments. (a) #DRC vs. K_{th} plot for NAND2_X1 and NAND3_X1. (b) #DRC vs. K_{th} plot for OAI21_X1 and OAI22_X1. (c) K_{th} values for 15 cell types per four standard-cell libraries.

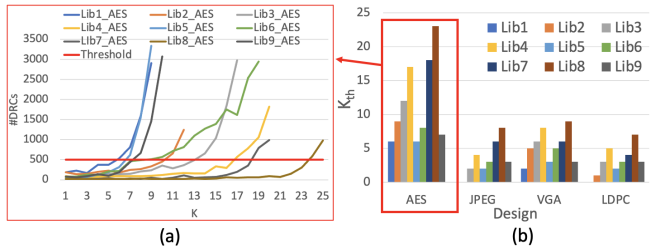

Fig. 5: Design-level routability assessment for the nine standard-cell libraries. (a) #DRC vs. K_{th} plot for AES design. (b) K_{th} values for AES, JPEG and VGA designs. We use 0.6 Util for AES, JPEG and VGA and 0.15 Util for LDPC design.

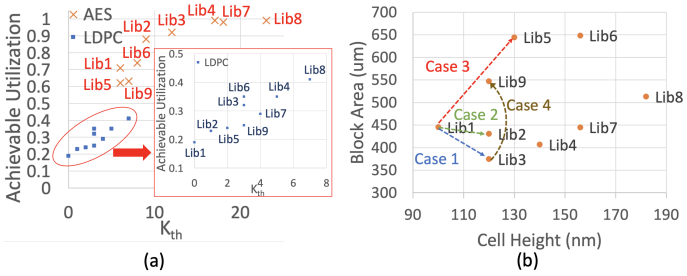

Fig. 6: Achievable density and block-area case study. (a) Plot for K_{th} vs. achievable utilization for AES and LDPC designs. (b) Achievable block-area study with the nine libraries and AES design. The arrows show block-level area and cell height differences in the case study, as also shown in Table III.

TABLE III: Results for achievable block-level area case study, expressed as overhead of *Lib B* relative to *Lib A*, i.e., $(Area_B - Area_A)/Area_A \times 100\%$

Case	*Lib A*		*Lib B*		Cell-Level Area Overhead	Block-Level Area Overhead
	Name	CH	Name	CH		
Case 1	Lib1	100nm	Lib3	120nm	**20%**	**-7%**
Case 2	Lib1	100nm	Lib2	120nm	**20%**	**19%**
Case 3	Lib1	100nm	Lib5	130nm	**30%**	**14%**
Case 4	Lib3	100nm	Lib9	120nm	**20%**	**15%**

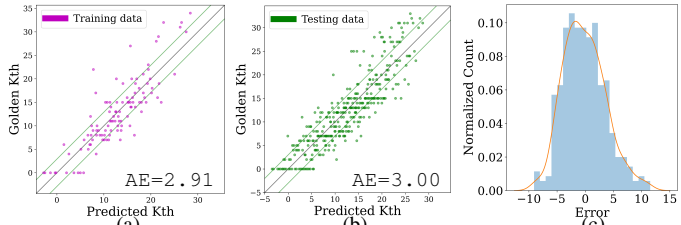

Fig. 7: Comparison between golden K_{th} (extracted from P&R experiments) and predicted K_{th} from the best model chosen by AutoML [9], (i.e., *GeneralizedLinearModel (GLM)*). (a) Results for 128 training data sampled by LHS. (b) Results for 320 testing data. (c) Error distribution on the testing dataset with kernel density estimation (KDE) plot.