Design Flow Enhancements for DNA Arrays*

Andrew B. Kahng Ion I. Măndoiu† Sherief Reda Xu Xu Alex Z. Zelikovsky‡
CSE Department, University of California at San Diego
†CSE Department, University of Connecticut
‡ CS Department, Georgia State University
E-mail: {abk,sreda,xuxu}@cs.ucsd.edu, ion@engr.uconn.edu, alexz@cs.gsu.edu

Abstract

DNA probe arrays have recently emerged as one of the core genomic technologies. Exploiting analogies between manufacturing processes for DNA arrays and for VLSI chips, we demonstrate the potential for transfer of methodologies from the 40-year old field of electronic design automation to the newer DNA array design field. Our main contributions in this paper are the following. (1) We give a new design flow for DNA arrays which enhances current methodologies by adding flow-awareness to each optimization step and introducing feedback loops. (2) We propose solution methods for new formulations integrating multiple design steps, including probe selection, placement, and embedding. (3) We give results of a comprehensive experimental study showing that significant improvements in solution quality can be achieved by using the enhanced methodologies.

1 Introduction

DNA probe arrays – DNA arrays or DNA chips for short – have recently emerged as one of the core genome technologies. They provide a cost-effective method for obtaining fast and accurate results in a wide range of genomic analyses, including gene expression monitoring, mutation detection, and single nucleotide polymorphism analysis (see [27] for a survey). The number of applications is growing at an exponential rate [16, 35], already covering a diversity of fields ranging from health care to environmental sciences and law enforcement. The reasons for this rapid acceptance of DNA arrays are a unique combination of robust manufacturing, massive parallel measurement capabilities, and highly accurate and reproducible results.

Today, most DNA arrays are manufactured through a highly scalable process, referred to as *Very Large-Scale Immobilized Polymer Synthesis (VLSIPS)*, that combines photolithographic technologies adapted from the semiconductor industry with combinatorial chemistry [1, 2, 13]. Similar to Very Large Scale Integration (VLSI) circuit manufacturing, multiple copies of a DNA array are simultaneously synthesized on a *wafer*, typically made out of quartz. To initiate synthesis, linker molecules including a photo-labile protective group are attached to the wafer, forming a regular 2-dimensional pattern of synthesis sites. Probe synthesis then proceeds in successive steps, with one nucleotide (A, C, T, or G) being synthesized at a selected set of sites in each step. To select which sites will receive nucleotides, photolithographic *masks* are placed over the wafer. Exposure to light de-protects linker molecules at the non-masked sites. Once the desired sites have been activated in this way, a solution containing a single type of nucleotide (which bears its own photo-labile protection group to prevent the probe from growing by more than one nucleotide) is flushed over the wafer's surface. Protected nucleotides attach to the unprotected linkers, initiating the probe synthesis process. In each subsequent step, a new mask is used to enable selective de-protection and single-nucleotide synthesis. This cycle is repeated until all probes have been fully synthesized.

As the number of DNA array designs is expected to ramp up in coming years with the ever-growing number of applications [16, 35], there is an urgent need for high-quality software tools to assist in the design and manufacturing process. The biggest challenges to rapid growth of DNA array technology are the drastic increase in design sizes with simultaneous decrease of array cell sizes – next-generation designs are envisioned to have hundreds of millions of cells of sub-micron size [2, 27] – and the increased complexity of the design process, which leads to unpredictability of design quality and design turnaround time. Surprisingly enough, despite huge research efforts invested in DNA array applications, very few works are devoted to computer-aided optimization of DNA array design and manufacturing. Current design practices are dominated by ad-hoc heuristics incorporated in proprietary tools with unknown suboptimality. This will soon become a bottleneck for the next generation of high-density arrays, such as the ones currently being de-

---

[*] Work partially supported by Cadence Design Systems, Inc., the MARCO Gigascale Silicon Research Center, and NIH Award 1 P20 GM065762-01A1. The work of I.M. was performed while he was with the ECE Department at the Univeristy of California, San Diego.

signed at Perlegen [2].

In this paper we exploit the similarities between manufacturing processes for DNA arrays and VLSI chips and demonstrate significant potential for transfer of electronic design automation methodologies [11, 30] to the newer DNA array design field. Our main contributions in this paper are the following:

- We give a new design flow for DNA arrays, which enhances current methodologies by adding *flow-awareness* to each optimization step and introducing *feedback loops* (Section 2).
- We propose new formulations and solution methods that integrate probe placement and embedding with probe selection and deposition sequence design (Section 3).
- We empirically demonstrate significant solution quality improvements for the enhanced methodologies. In particular, we show that 5-7% improvement in border length can be achieved over the highest-quality scalable flow previously reported in the literature [23] by a tighter integration of probe placement and embedding (more precisely, by replacing synchronous initial probe embedding with the so-called "as soon as possible" embedding, see Section 4). Furthermore, we show that up to 15% improvement in border length can be achieved by integrating probe selection with probe placement and embedding (Section 5).

2 Main Steps of the DNA Array Design Flow

In this section we introduce the main steps of the design flow for DNA arrays (see Figure 1, solid arcs), noting the similarity to the VLSI design flow and briefly reviewing previous work. Then we discuss how the current DNA array design flow may be enhanced by adding *flow-awareness* to each optimization step and introducing *feedback loops* between steps - techniques that have proved very effective in the VLSI design context [11, 30].

2.1 Probe Selection

Analogous to logic synthesis in VLSI design, the probe selection step is responsible for implementing the desired functionality of the DNA array. Although probe selection is application-dependent, several underlying selection criteria are common to all designs, regardless of the intended application [1, 2, 26, 4, 21, 29].

First, in order to meet array functionality, the selected probes must have low hybridization energy for their intended targets and high hybridization energy for all other target sequences. Hence, a standard way of selecting probes is to select a probe of minimum hybridization energy from the set of probes which maximize the minimum number of mismatches with all other sequences [26]. Second, since selected probes must hybridize under similar operating conditions, they must have similar melting temperatures.[1] Finally, to simplify array design, probes are often constrained to be substrings of a predetermined nucleotide deposition sequence. Typically, there are multiple probe candidates satisfying these constraints.

2.2 Deposition Sequence Design

The number of synthesis steps directly affects manufacturing time and the number of masks in the mask set, and also directly affects the quantity of defective probes synthesized on the chip. Therefore, a basic optimization in DNA array design is to minimize the number of synthesis steps. In the simplest model, this optimization has been reformulated as the classical *shortest common supersequence* (SCS) problem [24, 34]: Given a finite alphabet Σ (for DNA arrays $\Sigma = \{A, C, T, G\}$) and a set $P = \{p_1, ..., p_t\} \subseteq \Sigma^n$ of probes, find a minimum-length string $s_{opt} \in \Sigma^*$ such that every string of P is a subsequence of s_{opt}. (A string p_i is a subsequence of s_{opt} if s_{opt} can be obtained from p_i by inserting zero or more symbols from Σ.) The SCS problem has been studied for over two decades from the point of view of computational complexity, probabilistic and worst-case analysis, approximation algorithms and heuristics, experimental studies, etc. (see, e.g., [5, 6, 7, 9, 14, 15, 20]).

The general SCS problem is NP-hard, and cannot be approximated within a constant factor in polynomial time unless $P = NP$ [20]. On the other hand, a $|\Sigma|$-approximation is produced by using the *trivial periodic supersequence* $s = (x_1 x_2 \dots x_{|\Sigma|})^n$, where $\Sigma = \{x_1, x_2, \dots, x_{|\Sigma|}\}$ Better results are produced in practice by a simple greedy algorithm usually referred to as the "majority merge" algorithm [14], or variations of it that add randomization, lookahead, bidirectionality, etc. (see, e.g., [24]).

Current DNA array design methodologies bypass the deposition design step and use a predefined, typically periodic deposition sequence such as $ACTGACTG\dots$ (see, e.g., [24, 34]).

2.3 Design of Control and Test Structures

DNA array manufacturing defects can be classified as *non-catastrophic*, i.e., defects that affect the reliability of hybridization results, but do not compromise its functionality when maintained within reasonable limits, and *catastrophic*, i.e., defects that render the chip unusable. Non-catastrophic defects are caused by systematic error sources in the VLSIPS manufacturing process, such as unintended illumination due to diffraction, internal reflection, and scattering. Their impact on hybridization reliability of the chip is reduced by using the Perfect Match/Mismatch strategy [1, 27], which is also used to reduce the contribution of

---

[1] Below the melting temperature, two complementary strands of DNA are always bound to each other, while above it they separate. A practical method for estimating the melting temperature is suggested in [21].

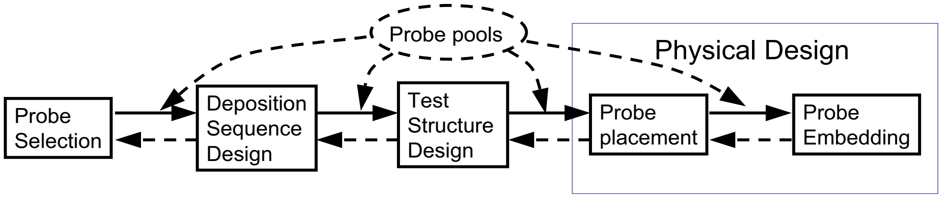

Figure 1. A typical DNA array design flow with solid arcs and proposed enhancements represented by dashed arcs.

background and cross-hybridization [27]. Under this strategy, a single nucleotide polymorphic probe ("Mismatch Probe") is synthesized next to each functional probe ("Perfect Match Probe"). Catastrophic defects are caused by missing, out-of-order, or incomplete synthesis steps, wrong or misaligned masks, etc.

DNA array test structures are the equivalent of built-in self-test (BIST) structures in VLSI design, and seeking to detect catastrophic manufacturing defects. The current approach to detecting catastrophic defects in DNA arrays is to synthesize a small set of test probes. Hubbell and Pevzner [19] have recently introduced a combinatorial approach that constructs such a small set of *fidelity* probes that, besides detecting manufacturing defects, can be used to identify the erroneous manufacturing steps. The approach relies on using multiple identical copies of the same fidelity probe, deliberately manufactured using different synthesis steps. A known target is then hybridized to these probes, and hybridization results reflect the quality of the manufacturing process. Further recent progress on the test structure design problem include results of [3, 8, 31].

2.4 Physical Design

Physical design for DNA arrays is equivalent to the physical design phase in VLSI design. It consists of two steps: *probe placement*, which is responsible for mapping selected probes onto locations on the chip, and probe embedding, which embeds each probe into the deposition sequence (i.e., determines synthesis steps for all nucleotides in the probe). The result of probe placement and embedding is the complete description of the reticles used to manufacture the array.

Under ideal manufacturing conditions, the functionality of a DNA array is not affected by the placement of the probes on the chip or by the probe synthesis schedules. In practice, since manufacturing process is prone to errors, probe locations and synthesis schedules affect to a great degree the hybridization sensitivity and ultimately the functionality of the DNA array. There are several types of synthesis errors that take place during array manufacturing. First, a probe may not loose its protective group when exposed to light, or the protective group may be lost but the nucleotide to be synthesized may not attach to the probe. Second, due to diffraction, internal reflection, and scattering, unintended illumination may occur at sites that are geometrically close to intentionally exposed regions. The first type of manufacturing errors can be effectively controlled by careful choice of manufacturing process parameters, e.g., by proper control of exposure times and by insertion of correction steps that irrevocably end synthesis of all probes that are unprotected at the end of a synthesis step [1]. Errors of the second type result in synthesis of unforeseen sequences in masked sites and can compromise interpretation of hybridization intensities. To reduce such uncertainty, one can exploit the freedom available in assigning probes to array sites during placement and in choosing among multiple probe embeddings, when available. The objective of probe placement and embedding algorithms is therefore to minimize the sum of border lengths in all masks, which directly corresponds to the magnitude of the unintended illumination effects. Reducing these effects improves the signal to noise ratio in image analysis after hybridization, and thus permits smaller array sites or more probes per array [18].

The border minimization problem was first considered for *uniform arrays* (i.e., arrays containing all possible probes of a given length) by Feldman and Pevzner [12], who proposed an optimal solution based on 2-dimensional Gray codes. Hannenhalli et al. [17] gave a heuristic for synthesizing arbitrary sets of probes, but considered only *synchronous* probe embeddings. This embedding method requires a periodic deposition sequence, and mandates that the i^{th} nucleotide of a probe be embedded at the unique matching position available in the i^{th} period (see Figure 2(b)). The method in [17] is to order the probes in a traveling salesman problem (TSP) tour that heuristically minimizes the total Hamming distance between neighboring probes. The tour is then *threaded* into the two-dimensional array of sites, using a technique similar to one previously used in VLSI design [25]. For the same synchronous context, [22] suggested an *epitaxial*, or "seeded crystal growth", placement heuristic similar to heuristics explored in the VLSI circuit placement literature by [28, 32].

The general border minimization problem, which allows arbitrary, or *asynchronous* probe embeddings (see Figure 2(c)), was introduced by Kahng et al. [22]. They proposed a dynamic programming algorithm that embeds a given probe optimally with respect to fixed embeddings of the probe's neighbors. This algorithm is used as a building block for designing several algorithms that improve a placement by re-embedding probes, but without re-placing them. Very recently, [23] proposed methods with near-linear run-

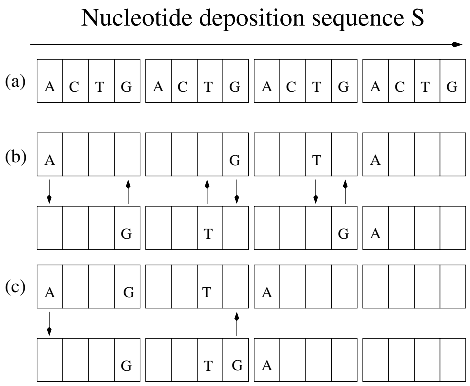

Figure 2. (a) Periodic deposition sequence. (b) Synchronous embedding of the probes $AGTA$ and $GTGA$ gives 6 border conflicts (indicated by arrows). (c) "As soon as possible" asynchronous embedding of the probes $AGTA$ and $GTGA$ gives only 2 border conflicts.

time combining simple ordering-based heuristics for *initial placement*, such as lexicographic sorting followed by threading, with heuristics for *placement improvement*, such optimal reassignment of an "independent" set of probes [33] chosen from a sliding window [10], or a row-based implementation of the epitaxial algorithm that speeds-up the computation by considering only a limited number of candidates when filling each array site.[2]

2.5 Flow Enhancements

The current design flow can be significantly improved by introducing flow-aware problem formulations, adding feedback loops between optimization steps, and/or integrating multiple optimizations. These enhancements, which are represented schematically in Figure 1 by the dashed arcs, are similar to flow enhancements that have proved very effective in the VLSI design context [11, 30].

In this paper we concentrated on two such enhancements, both aiming for further reductions in total border length. The first enhancement is a tighter integration between probe placement and embedding; this enhancement is discussed in Section 4. The second enhancement is the integration between physical design and probe selection, which is achieved by passing the entire pools of candidates available for each probe to the physical design step. As shown in Section 5, this enhancement enables significant improvements (up to 15%) in border length compared to best previous flows [23].

Other feedback loops and integrated optimizations are possible but are not explored in this paper. Faster and more targeted probe selection may be achievable by adding a feedback loop to provide updated selection rules and parameters to the probe selection step. Integrating deposition sequence design with probe selection may lead to further reductions in the number of masks by exploiting the freedom available in choosing the candidates for each probe. Although we show in Section 6 that a simple method for integrating physical design with deposition sequence design leads to insignificant improvements in solution quality, this does not rule out future improvements from tighter integrations between the two steps, such as the addition of a feedback loop transferring to the deposition sequence design step conflict map information (i.e., border length distribution across deposition steps) generated by physical design.

3 Formulation and Core Algorithms for Integrated Probe Selection and DNA Array Physical Design

In this section we formally introduce the border-length minimization objective. We then give an integrated formulation capturing several of the flow enhancements suggested in the previous section and further studied in Sections 4-5. We conclude the section by describing two algorithms used as building blocks in these studies.

Let $M_1, M_2, \ldots, M_K$ denote the sequence of masks used in the synthesis of an array, and let $s_i \in \{A, C, T, G\}$ be the nucleotide synthesized after exposing mask M_i. Every array probe must be a subsequence of the *nucleotide deposition sequence* $S = s_1 s_2 \ldots s_K$. Often, a probe corresponds to multiple subsequences of S, and one such subsequence must be chosen as the synthesis schedule for the probe. Clearly, the geometry of the masks is uniquely determined by the placement of the probes on the array and the particular synthesis schedule used for each probe.

Formally, array design can be viewed as a *3-dimensional placement* problem [22]: two dimensions represent the site array, and the third dimension represents the nucleotide deposition sequence S (see Figure 3). Each layer in the third dimension corresponds to a mask that induces deposition of a particular nucleotide (A, C, G, or T); a probe is *embedded* within a "column" of this 3-dimensional placement representation. Border length of a given mask is computed as the number of *conflicts*, i.e., pairs of adjacent exposed and masked sites in the mask. Given two adjacent embedded probes p and p', the *conflict distance* $d(p, p')$ is the number of conflicts between the corresponding columns. The border length of the embedding is the sum of conflict distances between adjacent probes, and the border length minimization problem seeks to minimize this quantity.

To integrate probe selection and physical design, we pass the entire pools of candidates for each probe to the physical design step (Figure 1). This gives additional freedom during placement and embedding, and may potentially reduce final border cost. Indeed, theoretical analysis omitted from

---

[2] The work of [23] also extends probe placement algorithms to handle practical concerns such as pre-placed control probes, presence of polymorphic probes, unintended illumination between non-adjacent array sites, and position-dependent border conflict weights.

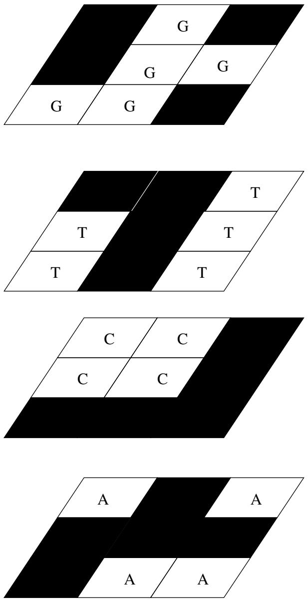

Figure 3. 3-dimensional probe placement with 4 masks and $S = ACTG$. Total border length is 24 (7 on the A mask, 4 on the C mask, 6 on the T mask, and 7 on the G mask).

the paper due to space constraints shows that the expected Hamming distance between two random probes of length 25 (which is proportional to the number of border conflicts between them under the commonly used synchronous embedding of the probes [17]) is 18.75. On the other hand, the expected Hamming distance between the closest candidates from two pools of size 10 is less than 13.

DNA array physical design with probe pools is captured by the following problem formulation:[3]

Integrated DNA Array Design Problem

Given:

- Pools of candidates $\{p_{ij} \mid j = 1, \ldots, l_i\}$ for each probe $i = 1, \ldots, N^2$, where $N \times N$ is the size of array
- The number of masks K

Find:

1. A single probe p_{ij} among the l_i available candidates for probe i, $i = 1, \ldots, N^2$,
2. A deposition sequence $S = s_1, \ldots, s_K$ which is a supersequence of all chosen probes p_{ij},
3. A placement of the chosen probes p_{ij} into an $N \times N$ array,
4. An embedding of the chosen probes p_{ij} into the deposition sequence S

Such that:

- The total number of conflicts between adjacent embedded probes is minimized

Although the flow in Figure 1 suggests a particular order for making the choices 1-4, the integrated formulation above allows interleaving these decisions. The following two algorithms capture key optimizations in the integrated formulations, and are used as core building blocks in the solution methods evaluated in Sections 4–5. They are "probe pool" versions of the Row-epitaxial and re-embedding algorithms proposed in [23], and degenerate to the latter ones in the case when each probe pool contains a single candidate.

- The *Row-Epitaxial algorithm* (REPTX) is the extension to probe pools of a probe placement algorithm in [23]. REPTX performs choices 1 and 3 for given choices 2 and 4, i.e., it simultaneously chooses already embedded candidates from the respective pools and places them in the $N \times N$ array. The input of REPTX consists of probe candidates p_{ij} embedded in the deposition sequence S written as a sequence of length $K = |S|$ in the alphabet $\{A, C, T, G, Blank\}$, where A, C, T, G denote embedded nucleotides and $Blank$'s denote positions of S left unused by the embedded candidate. REPTX consists of the following steps: (1) Lexicographic sorting of probes (based on the first candidate, when more than one candidate is available); (2) Threading the sorted probes in row-by-row order into the $N \times N$ array; (3) Finding, in row-by-row order, the best probe candidate[4] among the not yet placed probes within a prescribed lookahead region.
- The *sequential re-embedding* algorithm is the extension to probe pools of the probe embedding algorithm in [23]. It complements REPTX by iteratively modifying candidate selections and their embeddings (choices 2 and 4) as follows. In row-by-row order, for each position in the $N \times N$ array, and for each candidate p_{ij} from the pool of the respective probe, an embedding having minimum number of conflicts with the existing embeddings of the neighbors is computed, and then the best embedded candidate replaces the current one.

4 Improved Integration of Probe Placement and Embedding

As noted in [22], allowing arbitrary, or asynchronous, embeddings leads to further reductions in border length (see Figure 2). An interesting question is how to best exploit the placement and embedding degrees of freedom. Previous methods [22, 23] can be divided into two classes: (1) methods that perform placement and embedding decisions simultaneously, and (2) methods that exploit the two degrees of freedom one at a time. Currently, best methods in the second class (e.g., synchronous row-epitaxial followed by chessboard/sequential in-place probe re-embedding [23])

---

[3] This formulation also integrates deposition sequence design. For simplicity, we leave out design of control and test sequences.

[4] I.e., the candidate having the minimum number of conflicts with already placed neighbors.

outperform the methods in the first class (e.g., the asynchronous epitaxial algorithm in [22]) in terms of both runtime and solution quality.

All known methods in the second class perform synchronous probe placement followed by iterated in-place re-embedding of the probes (with locked probe locations). More specifically, these methods perform the following 3 steps:

- Synchronous embedding of the probes.
- Probe placement with costs given by the Hamming distance between the synchronous probe embeddings.
- Iterated in-place probe re-embedding.

We note that significant reductions in border cost are possible by performing the placement based on asynchronous, rather than synchronous, embeddings of the probes, and therefore modify the above scheme as follows:

- Asynchronous embedding of the probes.
- Probe placement with costs given by the Hamming distance between the fixed asynchronous probe embeddings.
- Iterated in-place probe re-embedding.

Since solution spaces for placement and embedding are still searched independently of one another and the computation of the initial asynchronous embedding adds insignificant overhead, the proposed change is unlikely to adversely affect the runtime. However, because placement optimization is now applied to embeddings more similar to those sought in the final optimization stage, there is significant potential for improvement.

In the current embodiment of the modified scheme, we implement the first step by using for each probe the "as soon as possible," or *ASAP*, embedding. Under ASAP embedding the nucleotides in a probe are embedded sequentially by always using the earliest available synthesis step. The intuition behind using ASAP embeddings is that, since ASAP embeddings are more densely packed, the likelihood that two neighboring probes will both use a synthesis step increases compared to synchronous embeddings. This translates directly into reductions in the number of border conflicts. Indeed, theoretical analysis omitted here shows that the expected number of border conflicts between two random probes of length 25 is only ≈ 31.1 when both probes are embedded using ASAP, compared to 37.5 when the probes are embedded synchronously.

To empirically evaluate the advantages of ASAP embedding we compared on testcases ranging in size from 100×100 to 500×500 the "champion" method in [22, 23], which uses synchronous initial embeddings for the probes, with the corresponding method based on ASAP initial probe embeddings.[5] For both methods, the second and third steps are implemented using REPTX and sequential in-place probe re-embedding algorithms in [23] (see also Section 3).

Tables 1 and 2 give the border-length and CPU time (in seconds) for the two methods. Each number in these tables represents the average over 10 testcases of the given size. Surprisingly, the simple switch from synchronous to ASAP initial embedding results in 5-7% reduction in total border-length. Furthermore, the runtimes for the two methods are comparable. In fact, sequential re-embedding becomes faster in the ASAP-based method compared to the synchronous-based one since there is less room for optimizing the REPTX placement and hence the number of iterations drops (from 9 to 3 on the average).

5 Integrated Probe Selection and Physical Design

We explored two methods for exploiting the availability of multiple probe candidates during placement and embedding. The first method uses the row-epitaxial and sequential in-place probe re-embedding algorithms described in Section 3. This method is an instance of integration between multiple flow steps, since probe selection decisions are made during probe placement and can be further changed during probe re-embedding. The detailed steps are as follows:

- Perform ASAP embedding of all probe candidates.
- Run the REPTX placement algorithm using border costs given by the Hamming distance between the ASAP embeddings.
- Run the iterated sequential in-place probe re-embedding algorithm.

The second method follows the separation between candidate selection and placement+embedding. However, we modify probe selection to make it flow-aware, i.e., to make its results more suitable for the subsequent placement and embedding optimizations. Building on the observation that shorter probe embeddings lead to improved border length, we choose from the available candidates the one that embeds in the *least* number of steps of the standard periodic deposition sequence using ASAP:

- Perform ASAP embedding of all probe candidates.
- Select from each pool of candidates the one that embeds in the least number of steps using ASAP.
- Run the REPTX placement algorithm using only the selected candidates and border costs given by the Hamming distance between the ASAP embeddings.
- Run the iterated sequential in-place probe re-embedding algorithm, again using only selected candidates.

---

[5] All experiments reported in this paper were performed on testcases obtained by generating each probe candidate uniformly at random. The probe length was set to 25, which is the typical value for commercial arrays [1]. Unless otherwise noted, we used the canonical periodic deposition

Chip	Synchronous Initial Embedding			ASAP Initial Embedding			Percent Improv.
	Sync Embed	REPTX	Re-Embed	ASAP Embed	REPTX	Re-Embed	
100	619153	502314	415227	514053	393765	389637	5.2
200	2382044	1918785	1603745	1980913	1496937	1484252	6.7
300	5822857	4193439	3514087	4357395	3273357	3245906	6.9
500	18786229	11203933	9417723	11724292	8760836	8687596	7.0

Table 1. Total border cost (averages over 10 random instances) for synchronous and ASAP initial probe embedding followed by row-epitaxial and iterated sequential in-place probe re-embedding.

Chip	Synchronous Initial Embedding			ASAP Initial Embedding		
	Sync+REPTX	Re-Embed	Total	ASAP+REPTX	Re-Embed	Total
100	166	81	247	188	29	217
200	1227	340	1567	1302	114	1416
300	3187	748	3935	2736	235	2971
500	8495	2034	10529	6391	451	6842

Table 2. CPU seconds (averages over 10 random instances) for synchronous and ASAP initial probe embedding followed by row-epitaxial and iterated sequential in-place probe re-embedding.

Table 3 gives the border-length and the runtime (in CPU seconds) for the two methods of combining probe placement and embedding with probe selection (each number represents the average over 10 testcases of the given size). We varied the number of candidates available for each probe between 1 and 16.

As expected, for each method and chip size, the improvement in solution quality grows monotonically with the number of available candidates. The improvement is significant (up to 15% when running the first method on a 100×100 chip with 16 candidates per probe), but varies non-uniformly with the method and chip size. For small chips the first method gives better solution quality than the second. For chips of size 200×200 the two methods give comparable solution quality, while for chips with size 300×300 or larger the second method is better (by over 5% for 500×500 chips with 8 probe candidates). The second method is faster than first for all chip sizes. The speedup factor varies between 5× and 40× when the number of candidates varies between 2 and 16. Interestingly, the runtime of the second method is slightly improving with the number of candidates, the reason being that the number of iterations of sequential re-embedding decreases when the length of the ASAP embedding of the selected candidates decreases.

6 Conclusions

In this paper we have proposed enhancing the design flow for DNA arrays by introducing flow-aware optimization formulations and adding feedback loops between optimization steps. We have proposed integrated formulations for probe selection and physical design, and experimentally verified that integrated optimizations lead to significant improvements in solution quality.

Given the significant reduction in border length achieved by integrating physical design with probe selection, it is natural to explore integration with other design steps, in particular with deposition sequence design. In a preliminary assessment of this integrated optimization, we ran the winning algorithm from Section 4 (REPTX followed by sequential re-embedding using ASAP initial embedding of the probes) for all 24 periodic deposition sequences. On chips of size 100×100, the best deposition sequence gives only 0.17% improvement compared to the cost averaged over all 24 sequences.[6] This improvement is comparable with the inherent noise in the placement and embedding algorithm: for the same chip size, randomly choosing the order of nucleotides in the lexicographic sorting step of REPTX leads to an improvement of 0.08% for the best order versus the average over the 24 possible orders. In conclusion, consideration of multiple periodic deposition sequences during placement and embedding leads to improvements in solution quality that are too small to justify the increase in runtime. It is possible that more sophisticated integration methods (such as consideration of multiple *aperiodic* sequences, or the addition of a feedback loop transferring to the deposition sequence design step the conflict map information generated by physical design) to be more successful. We leave their study as subject of future research.

Other direction of future research is to find formulations and methods for integrated optimization of test structure design and physical design. Since test structures are typically pre-placed at sites uniformly distributed across the array, integrated optimization can have a significant impact on the

---

sequence, $(ACTG)^{25}$, and a lookahead of 10,000/chipsize in the REPTX algorithm. All reported runtimes are for a 2.4 GHz Intel Xeon server with 2GB of RAM running under Linux.

[6] The improvement remains in the same range when we take into consideration multiple probe candidates using the second method given in Section 5.

Chip Size	Pool Size	Multi-Candidate			ASAP-Based Selection		
		Border Cost	CPU Sec.	% Imp.	Border Cost	CPU Sec.	% Imp.
100	1	389637	217	–	389637	217	–
	2	372951	1040	4.3	377026	212	3.2
	4	357562	1796	8.2	363944	193	6.6
	8	343604	3645	11.8	351540	191	9.8
	16	330600	7315	15.2	339636	185	12.8
200	1	1484252	1416	–	1484252	1416	–
	2	1438182	6278	3.1	1435712	1176	3.3
	4	1386527	12750	6.6	1385556	1189	6.6
	8	1334273	27382	10.1	1336851	1121	9.9
	16	1284550	44460	13.5	1289566	1117	13.1
300	1	3245906	2971	–	3245906	2971	–
	2	3185015	14956	1.9	3141088	2724	3.2
	4	3093633	26514	4.7	3018490	2771	7.0
	8	2985393	51226	8.0	2921195	2603	10.0
	16	2878886	98189	11.3	2835695	2760	12.6
500	1	8687596	6842	–	8687596	6842	–
	2	8611468	51847	0.9	8407839	6090	3.2
	4	8477014	86395	2.4	8105358	6709	6.7
	8	8248838	161651	5.1	7807763	6085	10.1
	16	–	–	–	7518331	5986	13.5

Table 3. Total border cost and runtime (averages over 10 random instances) for the two methods of combining probe placement and embedding with probe selection. The improvement (in percents) is relative to the single-candidate version of the same code.

total border length.

References

[1] http://www.affymetrix.com

[2] http://www.perlegen.com

[3] N. Alon, C. J. Colbourn, A. C. H. Lingi and M. Tompa, "Equireplicate Balanced Binary Codes for Oligo Arrays", *SIAM Journal on Discrete Mathematics* 14(4) (2001), pp. 481-497.

[4] A.A. Antipova, P. Tamayo1 and T.R. Golub, " A strategy for oligonucleotide microarray probe reduction", Genome Biology 2002 3(12):research0073.1-0073.4

[5] K. Nandan Babu and S. Saxena, "Parallel algorithms for the longest common subsequence problem", *Proc. 4th Intl. Conf. on High-Performance Computing*, Dec. 1997, pp. 120-125.

[6] J. Branke and M. Middendorf, "Searching for shortest common supersequences", *Proc. Second Nordic Workshop on Genetic Algorithms and Their Applications*, 1996, pp. 105-113.

[7] J. Branke, M. Middendorf and F. Schneider, "Improved heuristics and a genetic algorithm for finding short supersequences", *OR Spektrum* 20(1) (1998), pp. 39-46.

[8] C.J. Colbourn, A.C.H. Lingi and M. Tompa, "Construction of optimal quality control for oligo arrays", *Bioinformatics* 18(4) (2002), pp. 529–535.

[9] V. Dancik, "Common subsequences and supersequences and their expected length", *Combinatorics, Probability and Computing* 7(4) (1998), pp. 365-373.

[10] K. Doll, F. M. Johannes, K. J. Antreich, "Iterative Placement Improvement by Network Flow Methods", *IEEE Transactions on Computer-Aided Design* 13(10) (1994), pp. 1189-1200.

[11] J. J. Engel et al. "Design methodology for IBM ASIC products", *IBM Journal for Rewsearch and Development* 40(4) (1996), pp. 387.

[12] W. Feldman and P.A. Pevzner, "Gray code masks for sequencing by hybridization", *Genomics*, 23 (1994), pp. 233–235.

[13] S. Fodor, J. L. Read, M. C. Pirrung, L. Stryer, L. A. Tsai and D. Solas, "Light-Directed, Spatially Addressable Parallel Chemical Synthesis", *Science* 251 (1991), pp. 767-773.

[14] D. E. Foulser, M. Li and Q. Yang, "Theory and algorithms for plan merging", *Artificial Intelligence* 57(2-3) (1992), pp. 143-181.

[15] C. B. Fraser and R. W. Irving, "Approximation algorithms for the shortest common supersequence", *Nordic J. Computing* 2 (1995), pp. 303-325.

[16] D.H. Geschwind and J.P. Gregg (Eds.), *Microarrays for the neurosciences: an essential guide*, MIT Press, Cambridge, MA, 2002.

[17] S. Hannenhalli, E. Hubbell, R. Lipshutz and P.A. Pevzner, "Combinatorial Algorithms for Design of DNA Arrays", in *Chip Technology* (ed. J. Hoheisel), Springer-Verlag, 2002.

[18] E. Hubbell and M. Mittman , *personal communication* (Affymetrix, Santa Clara, CA), July 2002.

[19] E. Hubbell and P.A. Pevzner, "Fidelity Probes for DNA Arrays", *Proc. Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 113-117.

[20] T. Jiang and M. Li, "On the approximation of shortest common supersequences and longest common subsequences", *SIAM J. on Discrete Mathematics* 24(5) (1995), pp. 1122-1139.

[21] L. Kaderali and A. Schliep, "Selecting signature oligonucleotides to identify organisms using DNA arrays", Bioniformatics 18:1340-1349, 2002.

[22] A.B. Kahng, I.I. Măndoiu, P.A. Pevzner, S. Reda, and A. Zelikovsky, "Border Length Minimization in DNA Array Design", *Proc. 2nd International Workshop on Algorithms in Bioinformatics (WABI 2002),* R. Guigó and D. Gusfield (Eds.), Springer-Verlag Lecture Notes in Computer Science Series 2452, pp. 435-448.

[23] A.B. Kahng, I.I. Măndoiu, P.A. Pevzner, S. Reda, and A. Zelikovsky, "Engineering a Scalable Placement Heuristic for DNA Probe Arrays", *Proc. 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2003.

[24] S. Kasif, Z. Weng, A. Derti, R. Beigel, and C. DeLisi, "A computational framework for optimal masking in the synthesis of oligonucleotide microarrays", *Nucleic Acids Research* vol. 30 (2002), e106.

[25] T. Kozawa et al., "Automatic Placement Algorithms for High Packging Density VLSI", *Proc. 20th Design Automation Conference (DAC 1983)*, pp. 175–181.

[26] F. Li and G.D. Stormo, "Selection of optimal DNA oligos for gene expression arrays," Bioinformatics 17(11):1067-1076, 2001.

[27] R.J. Lipshutz, S.P. Fodor, T.R. Gingeras, D.J. Lockhart, "High density synthetic oligonucleotide arrays" *Nat. Genet.* 21 (1999), pp. 20–24.

[28] B. T. Preas and M. J. Lorenzetti, eds., *Physical Design Automation of VLSI Systems*, Benjamin-Cummings, 1988.

[29] S. Rahmann. "Rapid large-scale oligonucleotide selection for microarrays", *Proc. IEEE Computer Society Bioinformatics Conference (CSB)*, 2002.

[30] N.A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Norwell, MA, 1995

[31] R. Sengupta and M. Tompa, "Quality Control in Manufacturing Oligo Arrays: a Combinatorial Design Approach", *Journal of Computational Biology* 9 (2002), pp. 1–22.

[32] K. Shahookar and P. Mazumder, "VLSI Cell Placement Techniques", *Computing Surveys* 23(2) (1991), pp. 143-220.

[33] L. Steinberg, "The backboard wiring problem: a placement algorithm", *SIAM Review* 3 (1961), pp. 37–50.

[34] A.C. Tolonen, D.F. Albeanu, J.F. Corbett, H. Handley, C. Henson, and P. Malik, "Optimized in situ construction of oligomers on an array surface", *Nucleic Acids Research* vol. 30 (2002), e107.

[35] J.A. Warrington, R. Todd, and D. Wong (Eds.). *Microarrays and cancer research* BioTechniques Press/Eaton Pub., Westboro, MA, 2002.