逻辑综合贝叶斯优化

摘要

摘要：在逻辑合成过程中优化电路的结果质量(QoR)是一项艰巨的挑战，需要探索指数级大小的搜索空间。虽然专家设计的操作有助于发现有效的序列，但逻辑电路复杂性的增加有利于自动化程序。本文复现了贝叶斯优化算法，并应用精英保留策略的快速非支配排序遗传算法，进行智能化组合逻辑优化与工艺映射问题，两个算法均取得较好的优化结果。

关键词：逻辑综合；贝叶斯优化；NSGA2

1 引言

在集成电路数字物理后端设计中，组合逻辑优化与工艺映射是数字设计前端的重要流程，组合逻辑的网表，经过对逻辑的化简和再映射为工艺库中的逻辑单元组成的等价网表，以求达到优化的面积和时序 [4]。但目前现有工具多采用固定的流程完成优化与映射，故需任务调度算法智能流程根据网表特征动态地给出合适的优化算法组合，输出功能等价的LUT（look-up table）网表，同时优化逻辑层数与LUT数量。

但随着超大规模集成电路的高速发展，逻辑综合读取处理输入aig网表设计也越来越大，优化算法的数量与设计参数越来越复杂，为了找到满足约束条件的最优解，必须检查庞大的设计空间，需要搜索的设计空间往往是多维非连续空间，很难找到一个规范的搜索集来系统的搜索整个空间，且设计空间的形状通常是由问题说明的，无法保证搜索整个空间，即使有一定约束条件，但也是NP-hard问题 [5]。

2 相关工作

电子设计自动化（EDA）工具中的逻辑综合将高级描述转换为门级网表，包括三个主要阶段：逻辑优化，技术映射和门级优化或后映射 [7]。每个阶段具有不同的质量度量和对应的算法。这些算法将迭代地应用于电路。然而，如何使用这些算法通常是由电路学决定的，它并不总是在所有电路上都能得到很好的优化。例如重写算法为技术映射前的预处理技术，通过共享公共逻辑而不增加延迟来减少面积，或在不增加面积的情况下最小化延迟 [9]；平衡算法用于改善工艺映射之后的延迟，以较小的面积代价实现延迟降低 [10]；基于定时技术的查找表映射算法用于计算顺序电路中所有节点的k-cut [11]。这些算法通过对序列的不同调整来提高最终网表实现的功率、性能和面积。

逻辑综合通常涉及到多个目标的优化，例如功耗、面积和延迟等。任务调度技术可以结合优化算法，根据不同的目标函数和约束条件，智能地调度综合任务的执行顺序和资源分配策略，以实现多目标的优化和权衡。

高层次综合的算法主要是调度和分配算法。高层次综合的算法主要是调度和分配算法。最初的调度算法只能对小规模的顺序操作进行调度，后来出现了支持条件分支、循环结构、多周期操作、链式操作、流水线操作和控制器综合的等的调度算法。目前调度算法的进一步研究，出现了支持多维约束（调度步数、时钟周期、面积、功耗等)的算法，也就是面向多目标的优化算法。因而诸如遗传算法、演化计算等有用武之地。分配算法主要解决操作步骤和硬件资源的匹配问题，开始研究的是操作步骤和简单元件之间的匹配，后来出现了操作步骤和高层次元件的匹配以及操作步骤和模板（高层次元件的组合）之间的匹配等，能同时完成调度和分配等多项任务的统一算法以及对某些特定(控制流为主)设计的形式化调度算法等。以及能同时完成调度和分配等多项任务的统一算法以及对某些特定（控制流为主）设计的形式化调度算法等。

3 本文方法

本文介绍了在逻辑综合过程中优化电路质量结果（QoR）的问题。作者强调了指数级搜索空间带来的挑战以及自动化流程的需求。作者提出了BOiLS算法，该算法将贝叶斯优化应用于综合操作的空间导航。BOiLS旨在高效且可扩展，无需人工干预。该算法利用高斯过程核函数和信任区域约束采集来平衡探索和开发。作者通过样本效率和QoR值的比较，展示了BOiLS相对于最先进方法的卓越性能。

在此基础上，引入精英保留策略的快速非支配排序多目标遗传算法(NSGA 2)在求解任务调度问题取得了较为不错的解集。故本文在NSGA2的基础上进行智能求解组合逻辑优化与工艺映射问题。

3.1 本文方法概述

3.1.1 逻辑综合贝叶斯优化

组合逻辑优化采用反向器图（AIG）表示电路的逻辑功能，目标是通过应用一系列初等变换简化AIG图，找到产生最高QoR的变换序列：

$$\mathrm{QoR}_c(\mathrm{seq}) = \frac{\mathrm{Area}_c(\mathrm{seq})}{\mathrm{Area}_c(\mathrm{ref})} + \frac{\mathrm{Delay}_c(\mathrm{seq})}{\mathrm{Delay}_c(\mathrm{ref})}.$$

深度学习和强化学习技术在组合逻辑优化表现出很高的样本复杂性，高维度数据需求意味在给定的电路中进行大量的计算，如利用CNN时，每个电路有10000个序列，或在RL中超过数千个代理环境交互次数。

本文是第一个采用现代贝叶斯优化来探索综合操作空间的算法，贝叶斯优化的特点是不需要人工经验，不需要对损失函数求梯度，搜索组合逻辑空间效率高，能够权衡探索和开发。贝叶斯优化目标就是在采集函数指导下，让f*尽量接近f(x)，在这个过程当中，本质是在不断地优化对目标函数f(x)的估计，随着观测的点越来越多，对函数的估计是越来越准确的，因此也有越来越大的可能性可以估计出f(x)真正的最小值。

用于全局优化问题的迭代优化方法。它通过在目标函数上进行选择性的采样来寻找最优解。基于贝叶斯推断建立高斯过程代理模型，来近似目标函数的潜在性质。该代理模型通过观察已有的采样数据进行训练，并提供了一个概率分布，用于指导下一个采样点的选择。

（1）逻辑综合的高斯函数

方法分两步进行。首先，利用面向AIG优化序列的核函数，对QoR数据拟合代理高斯过程。该高斯过程能够提高采样效率，同时实现函数不确定性估计。 QoR数据：

$$\mathcal{D}_t = \{\mathrm{seq}_i, -\mathrm{QoR}_{\mathcal{C}}(\mathrm{seq}_i)\}_{i=1}^{n_t}$$

假设：

$$-\mathrm{QoR}_{\mathcal{C}}(\mathrm{seq}) \sim \mathcal{GP}(0, k_\theta^{(LS)}(\mathrm{seq}, \mathrm{seq}'))$$

将电路优化序列表示为操作字符串，每个元素都是来自AIG优化算法作为优化算子，使用字符串子序列内核（sub-sequence string kernel，SSK）通过公共子串的数量来度量字符串之间的相似性。字符串seq与seq’之间的顺序定义为：$k_\theta^{(\mathrm{LS})}(\mathrm{seq}, \mathrm{seq}') = \sum_{u\in\sum^l} c_u(\mathrm{seq})c_u(\mathrm{seq}')$，其中，$C_u(\mathrm{seq})$，定义子序列$u$对seq的贡献为:$c_u(seq) = \theta_m^{|u|}\sum_{i\le i_1<\cdots<i_{|u|}\le|\mathrm{seq}|}\theta_g^{gap(u,i)}\Pi_u(\mathrm{seq}_i)\ u$，定义为： $gap(u,i) = i_{|u|} - i_1 + 1 - |u|$来控制字符串长度，匹配和间隙衰减：$\theta = (\theta_m, \theta_g) \in [0,1]^2$。投影梯度为$\theta_{update} = \mathrm{Pro}ject ion_{[0,1]^2}(\theta_{current} - \eta\nabla_\theta J(\theta_{current}))$，通过最小化负对数边界似然确定超参数$theta$:$\min_\theta J(\theta) = \frac{1}{2}\det(K_\theta(x_{1:n}, x_{1:n})) + \frac{1}{2}g^T K_\theta^{-1}(x_{1:n}, x_{1:n})g$

（2）最大化信赖域采集函数

在第二步中利用高斯过程构造新的综合流进行预测评估。为了有效处理高维搜索问题而利用局部搜索最大化信赖域采集函数。将高斯过程与上一步的核函数拟合，利用局部搜索最大化信赖域采集函数：$\max_{seq\in Aig^K}\alpha_{EI}(seq \mid D_t)$。在每一轮t，探寻迄今为止观察到的最优序列：$\widehat{seq_t}$。

对于不确定函数内部规律的问题，基于贝叶斯优化的算法探索逻辑综合优化空间可以取得较好的参数搜索效率。基于贝叶斯优化的逻辑综合算法有利于求解黑箱问题，除了基于信赖域的方法还有与蒙特卡罗树搜索结合，其难点主要在于结合具体问题设计代理模型、采集函数。

3.1.2 逻辑综合贝叶斯优化算法流程

Algorithm 1 逻辑综合贝叶斯优化算法流程

Input: 电路，最大求值次数N_{max}，每个序列K最大转换次数;

初始化与内核调优;
随机抽取样本n_0建立初始数据集$\mathcal{D}_0 = \{\mathrm{seq}_i, -\mathrm{QoR}_{\mathcal{C}}(\mathrm{seq}_i)\}_{i=1}^{n_t}$;
设置TR半径为$R_{n_0} = K$;
迭代优化：
for t = 0, ... , $N_{max} - 1$ do:
    使用$\mathcal{D}_t$拟合GP
    获得$seq_{t+1} \in \arg\max_{seq \in TR(\widehat{seq}_t, \rho_t)} \alpha_{EI}(seq \mid \mathcal{D}_t)$
    评估$\mathrm{QoR}_c(\mathrm{seq}_{t+1})$并记录数据
    更新TRs半径ρ_{t+1}
end for
输出探寻到操作的最佳序列$\widehat{seq_t}$

具体实现步骤如下：

（1）定义目标函数：明确要优化逻辑综合智能流程的目标函数：$\mathrm{QoR}_c(\mathrm{seq}) = \frac{\mathrm{Area}_c(\mathrm{seq})}{\mathrm{Area}_c(\mathrm{ref})} + \frac{\mathrm{Delay}_c(\mathrm{seq})}{\mathrm{Delay}_c(\mathrm{ref})}$,其中QoR为评估序列seq的性能，$c$代表逻辑与的两个fanin节点组成的AIG网表，Area为AIG映射后FPGA门级网表的面积，Delay为门级网表的时延，ref表示参考序列。

（2）选择先验模型：选择一个代理模型作为目标函数的先验模型，常用的模型包括高斯过程（Gaussian Process）和随机森林（Random Forest）等，这里选择为高斯过程。

（3）初始化采样：在目标函数上选择一组初始采样点，实际选择初始化为ref参考序列，后续为随机选择序列提供初始训练。

（4）训练高斯过程代理模型：使用初始采样数据训练代理模型，得到目标函数的近似模型。

（5）选择下一个采样点：根据代理模型的不确定性或期望改进，使用完全指定GP的协方差函数，构建输出预测分布来选择下一个采样点，训练和测试函数值联合分布。

（6）评估目标函数：在选定的采样点上评估目标函数的值，并获取真实的目标函数反馈。

（7）更新高斯过程代理模型：将新的采样数据与已有数据合并，重新训练代理模型，GP管道中的剩余成分涉及引入使用边际值进行调整的内核超参数，从而更好的适应数据集，以获得更准确的模型。

（8）重复步骤5-7：根据代理模型的预测结果和采样策略，迭代选择下一个采样点、评估目标函数和更新代理模型的步骤，直到达到预设的停止条件，如达到最大评估次数、目标函数收敛。

（9）输出最优解：根据代理模型和已有的采样数据，确定目标函数的最优解或近似最优解。

3.2 NSGA2算法

精英保留策略的快速非支配排序遗传算法 [2]（NSGA2）是 Srinivas 和 Deb 在NSGA的基础上提出，较NSGA算法更加优越，其采用了快速非支配排序，计算复杂度较NSGA 大大的降低；采用了拥挤度和拥挤度比较算子，代替了需要指定的共享半径 share Q，并在快速排序后的同级比较中作为胜出标准，使准 Pareto 域中的个体能扩展到整个 Pareto 域，并均匀分布，保持了种群的多样性；引入了精英策略，扩大了采样空间，防止最佳个体的丢失，提高了算法的运算速度和鲁棒性。其余方面NSGA2与基础遗传算法无异。

3.2.1 快速非支配排序

在快速非支配排序算法中，每个解集都会被分配两个参数：S_i与n_i,其中n_i表示支配解i的数量，S_i表示支配解i的解集。

排序流程：

（1）通过计算遍历获取各个解集的S_i与n_i;

（2）若第一个非支配层上的解满足$n_i = 0$，将其存入集合F_i中，对于F_i集中n_i的解，将其所支配个体的n_i减少1.此时新出现的$n_i = 0$的个体作为第二个非支配层上的解，将其放入另一个独立集合Q中；

（3）对于集合Q中的个体，重复（2）中操作，找到第三层非支配层；

（4）重复以上步骤，找到所有的非支配层；

（5）完成每层解划分后给该层解分配一个对应等级rank。

由于在第二层及以上层的解最多被$N-1$个解集支配，所以这些解在n_i变为0，及完成分层之前，最多会被访问$N-1$次，这些解最多有$N-1$个，因此，此排序最大计算复杂度为O(N^2)。

3.2.2 拥挤度以及拥挤比较算子

种群中每个个体i均设定一个拥挤度参数，用于评估个体适应度的指标。它表示个体周围的密度，即个体所在的区域有多少其他个体存在。拥挤度越大，表示个体所在的区域越密集，个体之间的距离越近。为了估计总体中某一特定解周围的解的密度，计算该点两侧沿每个目标的两个点的平均距离。如图1具体化表示这个量的距离是用最近邻作为顶点形成的长方体周长的估计(称为拥挤距离)。在图 1中，第i个个体在其前面(用圆标记)的拥挤距离是长方体(用虚线框表示)的平均边长。

拥挤度算子是用于计算个体的拥挤度的操作。在NSGA2算法中，拥挤度算子通过对每个目标函数值进行排序和标准化，然后计算每个个体在各个目标函数上的差值之和来计算拥挤度。具体而言，拥挤度算子按照目标函数值对个体进行排序，然后对于每个目标函数，计算个体在该目标函数上的差值，并将差值累加起来得到拥挤度。通过上述计算，每个个体i赋予：

(1)非支配等级rank;

(2)拥挤距离$i_{distance}$;

故拥挤比较运算$\prec_n$定义为：

$$i \prec_n j \quad \text{if} \quad (i_{rank} < j_{rank}) \text{ or } ((\mathrm{i}_{rank} = j_{rank})\mathrm{and}(i_{d\check{z}tance} > j_{\mathrm{distance}})).$$

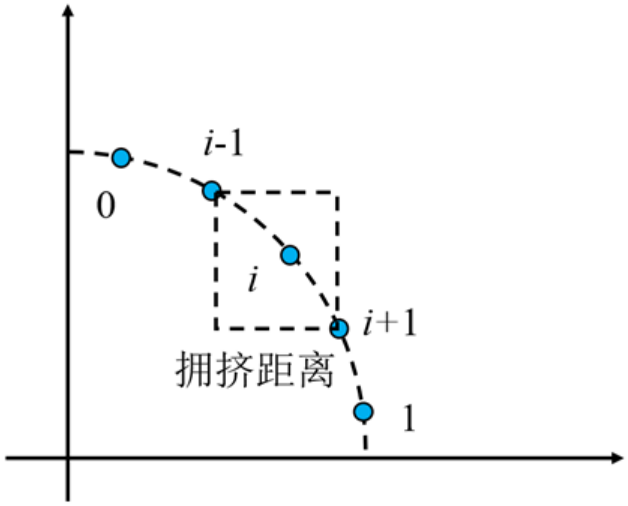

图 1. 拥挤距离

在两个具有不同非支配等级的解之间，更倾向于具有较好等级的解。否则，若两个解决方案属于同一等级，优先选择拥挤度较小的解。

拥挤度的计算可以保持种群的多样性，即促使个体在解空间中分布更加均匀。通过拥挤度，NSGA2算法能够选择出拥有较大拥挤度的个体，从而保留更多的多样性。拥挤度算子的使用使得NSGA2能够更好地平衡个体的多样性和适应度，从而提高算法的性能。

3.2.3 NSGA2算法流程

（1）生成初始种群P_0，并完成快速非支配排序;

（2）在初始种群中使用锦标赛选择机制、复制、变异算子生成子代种群Q_0;

（3）初始种群与子代种群Q_0合并后构成规模为$2N$的种群，在$2N$的种群上完成快速非支配排序、拥挤度计算，根据拥挤度比较算子（即偏序关系$<_n$）从第一层开始进行选择直到选够N个可以作为新父代种群的个体，实现精英保留策略。

（4）对新父代种群进行选择、交叉、变异，生成子代种群，合并子代和父代种群后进入下一轮循环如图2，直至迭代次数达到预先设定的最大值。

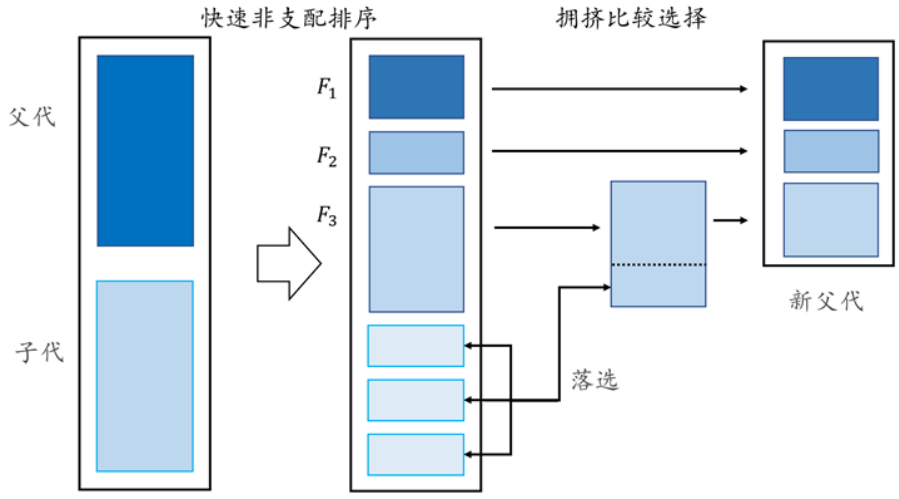

图 2. 新父代产生方式

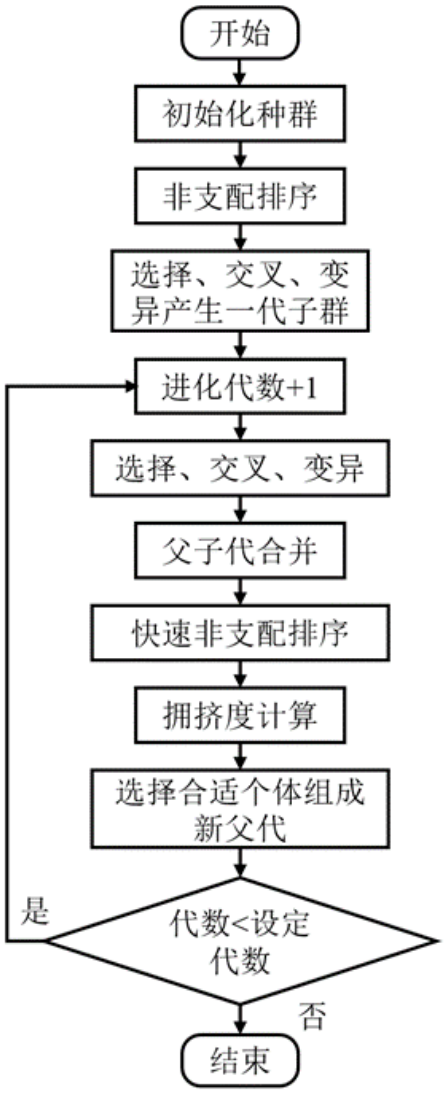

图 3. NSGA2算法流程图

4 复现细节

复现逻辑综合的贝叶斯优化算法，通过实验验证贝叶斯算法的正确性与有效性，实现精英保留策略的快速非支配排序多目标遗传算法（NSGA 2）进行比较并进行改进。

4.1 与已有开源代码对比

原代码逻辑综合优化算子基于ABC v1.01平台，本文基于适用于OSCC(Open Source Chip Community) iEDA平台的iMap v0.1。此外，本文利用原代码的部分功能，如目标函数，组合序列等功能，实现引入精英保留策略的快速非支配排序多目标遗传算法来求解智能求解组合逻辑优化与工艺映射问题。

4.2 实验环境搭建

实验环境基于Ubuntu18.04，python3.7与anaconda。以下为部署环境说明：

环境部署

```
#下载iMap
git clone https://gitee.com/oscc-project/iMAP.git
#下载NSGA2与Boils
git clone https://gitee.com/wang-rui13132008217/nsga2.git
#将NSGA2下的文件移动到iMAP/ai_infra目录下
mv nsga2/* iMap/ai_infra
cd iMap
#编译imap
mkdir build && cd build
cmake ..
make -j 8
#复制imap执行文件至ai_infra目录下
cd ..
cp bin/imap ai_infra
# 修改文件aig文件与结果存储路径：
# 修改results_storage_root_path.txt内为ai_infra文件夹所在路径。
# 创建python3.7虚拟环境
conda create -n yourEnv python=3.7 #创建python3.7虚拟环境
conda activate yourEnv
pip install -r requirements.txt #下载必要库
# NSGA2运行示例：
python GA/main_nsga2.py –designs_group_id arbiter –n_gen 100 –pop_size 20 –seq_length 10 –seed 0
# Boils运行示例：
python main_boils.py –designs_group_id arbiter –seq_length 10 –device -1 –seed 0
```

4.3 创新点

快速非支配排序算法：NSGA2采用了快速非支配排序算法，降低了计算复杂度，使得算法能够更快速地对个体进行非支配排序。

拥挤度和拥挤度比较算子：NSGA2引入了拥挤度和拥挤度比较算子，代替了需要指定的共享半径的适应度共享策略。这些算子有助于维持种群的多样性，促进种群中个体的均匀分布，从而有利于快速收敛。

二进制锦标赛配对选择：NSGA2使用二进制锦标赛选择算子，通过比较个体的非支配等级和拥挤度来进行选择，增加了选择压力，有助于加速收敛。

5 实验结果分析

通过对7种不同层级与门个数设计的QoR结果测验，与门个数在1000门以下限制优化时间为1分钟，1000至10000之间限制时间为5分钟，10000门以上限制1小时。优化序列长度均为20。采用计算与序列resyn2相比的相对改进（以%为单位）：

$$\frac{\mathrm{QoR}_c(\mathrm{resyn2}) - \mathrm{QoR}_c(\widehat{\mathrm{seq}_t})}{\mathrm{QoR}_c(\mathrm{resyn2})}.$$

BOiLS面积与深度的优化相对百分比提升均值为29.56%，NSGA2面积与深度的优化相对百分比提升均值为37.03%，NSGA2较BOiLS相对优化提升9.79%。

表 1. 优化QoR

设计名称	与门个数	BOiLS优化百分比	NSGA2优化百分比	NSGA2较BOiLS提升
router	257	52.54%	60.99%	10.91%
priority	978	12.35%	12.75%	0.43%
i2c	1342	17.57%	20.88%	3.34%
wb_dma_comb	3161	16.73%	17.66%	0.95%
tv80_comb	8921	11.77%	14.96%	3.43%
arbiter	11839	75.54%	94.52%	30.92%
aes_core_comb	21119	20.42%	37.43%	18.52%
	优化百分比均值：	29.56%	37.03%	9.79%

之所以NSGA2较BOiLS取得较好的结果，可能原因为BOiLS的高斯拟合过程较为耗时，NSGA2在此期间搜索到的解集较其拟合更为优异。侧面反应出BOiLS的收敛速度较为缓慢。

6 总结与展望

本文通过设计复现贝叶斯优化与NSGA2两种算法，进行智能化组合逻辑优化与工艺映射流程，通过基于OSCC(Open Source Chip Community)开源iEDA平台的iMap v0.1实验两种算法，均取得较好的实验结果，由于限制优化时间，贝叶斯优化算法取得效果并不十分理想，后续还需优化降低算法时间复杂度或减少计算量。

此外，有学者通过机器学习智能探索预测的方式，也取得了较好的性能收益 [12, 14]。后续逻辑综合组合逻辑优化与工艺映射的智能流程问题改进思路可以结合传统方式与机器学习、深度学习等方式进行探寻优化，求解优化算子的更优序列。

参考文献

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[3] Sofiane Ellouz, Patrice Gamand, Christophe Kelma, Bertrand Vandewiele, and Bruno Allard. Combining internal probing with artificial neural networks for optimal rfic testing. In *2006 IEEE International Test Conference*, pages 1–9, 2006.

[4] Antoine Grosnit, Cedric Malherbe, Rasul Tutunov, Xingchen Wan, Jun Wang, and Haitham Bou Ammar. Boils: Bayesian optimisation for logic synthesis. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1193–1196, 2022.

[5] Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda. Drills: Deep reinforcement learning for logic synthesis. *CoRR*, abs/1911.04021, 2019.

[6] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, Xuefei Ning, Yuzhe Ma, Haoyu Yang, Bei Yu, Huazhong Yang, and Yu Wang. Machine learning for electronic design automation: A survey, 2021.

[7] Xing Li, Lei Chen, Fan Yang, Mingxuan Yuan, Hongli Yan, and Yupeng Wan. Himap: A heuristic and iterative logic synthesis approach. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 415–420, New York, NY, USA, 2022. Association for Computing Machinery.

[8] Gai Liu and Zhiru Zhang. Pimap: A flexible framework for improving lut-based technology mapping via parallelized iterative optimization. *ACM Trans. Reconfigurable Technol. Syst.*, 11:23:1–23:23, 2019.

[9] A. Mishchenko, S. Chatterjee, and R. Brayton. Dag-aware aig rewriting: a fresh look at combinational logic synthesis. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 532–535, 2006.

[10] Alan Mishchenko, Robert Brayton, Stephen Jang, and Victor Kravets. Delay optimization using sop balancing. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 375–382, 2011.

[11] Peichen Pan and Chih-Chang Lin. A new retiming-based technology mapping algorithm for lut-based fpgas. In *Symposium on Field Programmable Gate Arrays*, 1998.

[12] Eleonora Testa, Mathias Soeken, Luca Gaetano Amar, and Giovanni De Micheli. Logic synthesis for established and emerging computing. *Proceedings of the IEEE*, 107(1):165–184, 2019.

[13] Samuel Ward, Duo Ding, and David Z. Pan. Pade: A high-performance placer with automatic datapath extraction and evaluation through high-dimensional data learning. In *DAC Design Automation Conference 2012*, pages 756–761, 2012.

[14] Chenghao Yang, Yinshui Xia, Zhufei Chu, and Xiaojing Zha. Logic synthesis optimization sequence tuning using rl-based lstm and graph isomorphism network. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(8):3600–3604, 2022.

[15] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. Developing synthesis flows without human knowledge. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, 2018.