ATPG（自动测试模式生成，Automatic Test Pattern Generation）算法是一种用于数字集成电路（IC）故障检测的方法。它可以自动生成用于检测电路中潜在缺陷的测试模式。ATPG算法的主要目标是通过最小化所需测试模式数量来实现高覆盖率和低测试成本。

ATPG算法的工作原理：

* 建立一个电路模型：使用标准硬件描述语言(HDL)或其他方式对设备进行建模。
* 定义故障列表：基于设备模型，列出可能导致电路失效的所有单点故障。常见的故障类型包括连接断开、短路和程控故障。
* 生成测试模式：使用算法来自动生成检测故障列表中每个故障的输入向量。这些输入向量将被应用到设备上，以观察其输出是否与预期一致。
* 评估和优化：分析生成的测试模式的覆盖率和效率，并根据需要优化算法以改进结果。

这种方法的主要优势是能提高故障检测的速度和效率，从而有助于生产高质量的产品并降低制造成本。ATPG算法已广泛应用于半导体行业，特别是在制造数字IC和微处理器等设备方面。

以下是一些常见的ATPG（自动测试模式生成，Automatic Test Pattern Generation）算法：

故障模拟法（Fault Simulation）：通过对部分输入向量进行假设，使用模拟器计算预期输出。同时，比较正常电路和故障电路的输出值以确定当前输入向量是否能够检测到故障。

D算法（D-algorithm）：利用错误传播机制来识别故障的深度优先搜索（DFS）策略。它通过控制和感知路径来判断是否可以区分正常和故障状态。

PODEM算法（Path-Oriented Decision Making）：这是一种启发式搜索算法，通过递归回溯和决策树生成输入向量。算法尝试通过调整输入值在控制和感知路径上传播故障。

FAN算法（Fanout-Oriented Cone）：FAN算法是对D算法的改进，将电路分解成多个锥形子结构以减少搜索空间。它能更快地找到区分正常和错误状态的测试向量。

SAT-Based ATPG：该方法将故障检测问题转换为可满足性问题（SAT），并利用现有的SAT求解器来识别故障。这种方法通常具有较高的内存效率和计算速度。

Genetic Algorithm-based ATPG：基于遗传算法的ATPG方法使用类似自然进化的策略找到最佳测试模式。这种方法可以有效地发现全局最优解并降低搜索空间。

动态编译技术（Dynamic Compilation Techniques）：将故障模拟器从软件转移到硬件（如FPGA），以加速模拟过程，进而提高ATPG的效率。

D算法是第一个完备的ATPG算法,也是当下主流的ATPG算法。它的主要思想是逐级敏化从故障源到电路所有输出的全部可能的通路。在D算法之后，又衍生出一些优化算法，如PODEM，FAN，CPT等等，不同算法的思想、方法各有不同，但目的都是相同的。作为ATPG算法的鼻祖，学习D算法的思想的流程对我们将来从事DFT行业是非常有益的。

1. 简介

D算法是一种有效的故障检测方法，用于发现集成电路和电子系统中的潜在问题。它采用布尔逻辑方程来表示目标系统，并根据生成的测试向量对其进行故障注入以及传播分析。通过对比正常和有故障的输出结果，可以确定故障位置和类型。

2. D算法原理与步骤

D算法的基本步骤如下：

初始化： 确定输入向量、故障发现概率阈值，将所有节点标记为未访问。

故障模拟： 模拟故障电路，计算假设故障所导致的输出差异。

故障传播： 从输出端开始，沿着故障可能传播的路径回溯，直至找到一个合适的测试点或达到预定的搜索深度。

D-前驱计算： 根据故障传播路径，计算D-前驱集，即故障可能影响的上游节点。

故障定位： 通过对比正常电路和含故障电路的输出，找到可能的故障位置。

生成测试向量： 基于D-前驱集，生成可区分故障与正常电路的输入测试向量。

迭代过程: 重复以上步骤，直至找到所有潜在的故障位置和类型。

3. 示例实现

给定一个简单的数字系统，其中Y = A ∧ (B ∨ C)，故障仅发生在与非门上。假设存在两种可能的故障：输出永远保持高电平（stuck-at-1，SA1）或低电平（stuck-at-0，SA0）。接下来使用D-算法进行分析。

SA1 故障分析:

初始化： 设定输入向量为(A, B, C)，所有节点标记为未访问。

故障模拟： 当故障发生时，Y始终保持为1。即使输入满足正常逻辑，Y也不会变成0。

故障传播： 要使该故障在输出端可观察到，我们需要找到一组输入，使得在正常情况下Y=0，而在故障情况下Y=1。

D-前驱计算： 为了使Y=0，我们可以将A设置为0，这样无论(B ∨ C)的值为多少，Y都会是0。在故障情况下，Y将保持为1。

故障定位： 对比正常和故障电路的输出，发现当A=0时，Y的状态不同，因此故障位于与非门上。

生成测试向量： 任何具有A=0的输入向量都可作为测试向量，例如(0, 0, 0)、(0, 1, 0)、(0, 0, 1)、(0, 1, 1)。

SA0 故障分析:

初始化： 设定输入向量为(A, B, C)，所有节点标记为未访问。

故障模拟： 当故障发生时，Y始终保持为0。即使输入满足正常逻辑，Y也不会变成1。

故障传播： 要使该故障在输出端可观察到，我们需要找到一组输入，使得在正常情况下Y=1，而在故障情况下Y=0。

D-前驱计算： 为了使Y=1，我们需要A=1且(B ∨ C)=1。在故障情况下，Y将保持为0。

故障定位： 对比正常和故障电路的输出，发现当A=1且(B ∨ C)=1时，Y的状态不同，因此故障位于与非门上。

生成测试向量： 任何具有A=1且(B ∨ C)=1的输入向量都可作为测试向量，例如(1, 1, 0)、(1, 0, 1)、(1, 1, 1)。

3.1 MATLAB示例

MATLAB代码实现了一个简化版的D算法，适用于一个3输入端口的AND门。用户需要手动选择要注入的故障节点以及生成的测试向量。示例代码见报告附录A。

优缺点：

优点：

高故障覆盖率：D-算法能有效发现和定位数字电路中的故障。

适用范围广：适用于不同类型和规模的数字电路。

计算效率较高：通过聚焦故障传播路径，减少了搜索空间。

缺点：

对于某些复杂电路，D-算法可能面临搜索空间过大的问题。

某些情况下可能难以生成测试向量，需要额外的优化手段。

PODEM

PODEM（Path-Oriented Decision Making）算法是一种用于自动测试模式生成（ATPG, Automatic Test Pattern Generation）的故障检测方法。它主要应用于数字电路，以诊断单一固定故障（stuck-at faults）。与D-算法类似，PODEM算法也使用逻辑回溯和隐式测试分析来生成能区分正常和故障电路的输入测试向量。

主要步骤：

初始化： 确定目标故障、初始决策节点、所有节点的值。设所有节点未访问。

故障模拟： 模拟故障电路，计算故障所导致的输出差异。

隐式测试： 使用隐式测试策略选择一个观察点，尝试在该点上检测故障。

传播约束： 从观察点开始，沿着故障可能传播的路径回溯，找到一个合适的决策节点或达到预定的搜索深度。

决策： 如果当前决策满足故障检测条件，则移向下一个决策节点；否则回溯至上一个决策节点并尝试其他选项。

测试向量生成： 当所有决策均满足故障检测条件时，基于这些决策生成输入测试向量。

优缺点：

优点：

高故障覆盖率：PODEM算法能有效发现和定位数字电路中的故障。

良好的计算效率：聚焦于具有潜在故障影响的路径，减少搜索空间。

适用范围广：适用于不同类型和规模的数字电路。

缺点：

当多个故障同时出现时，算法可能需要额外的优化手段。

假设电路N中有节点g，其原始输入和输出分别为n个和m个，则有

要诊断故障，需要解方程

并且至少存在一个j，使得

PODEM算法将解方程的n为空间搜索问题转化为分支判决树问题。