ICS 35.080
L 77

中华人民共和国国家标准

GB/T 37691—2019

可编程逻辑器件软件安全性设计指南

Guide for programmable logic device software safety design

2019-08-30 发布　　2020-03-01 实施

国家市场监督管理总局
中国国家标准化管理委员会　发布

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由全国信息技术标准化技术委员会(SAC/TC 28)提出并归口。

本标准起草单位:中国航天科工集团公司第三研究院第三〇四研究所、中国电子技术标准化研究院、中国电子科技集团第三十研究所、国家卫星海洋应用中心。

本标准主要起草人:孟伟、杨楠、王黎、姜晓辉、胡勇、黄琼、王国锋、张津荣、李文鹏、朱琳、张国宇、肖崇俭、寇科男、李恺、巫忠跃、彭鸣、毛伟、许卓琦、张旸旸、陈元、史玥、陈鹏、刘廷、杨永生、王希、孙健、葛永娇。

可编程逻辑器件软件安全性设计指南

1 范围

本标准给出了可编程逻辑器件软件安全性设计的指导和建议，并给出了需考虑要点有关的信息。

本标准适用于可编程逻辑器件软件的系统需求分析、软件需求分析、设计和实现时的安全性设计。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 11457—2006 信息技术 软件工程术语

GB/T 18349 集成电路/计算机硬件描述语言 Verilog

GB/T 33781—2017 可编程逻辑器件软件开发通用要求

GB/T 33783—2017 可编程逻辑器件软件测试指南

3 术语和定义

GB/T 11457—2006、GB/T 33781—2017 和 GB/T 33783—2017 界定的以及下列术语和定义适用于本文件。

3.1

可编程逻辑器件 programmable logic device

允许用户编程(配置)实现所需逻辑功能的器件。

[GB/T 33781—2017，定义 3.1.1]

3.2

可编程逻辑器件软件 programmable logic device software

针对 FPGA、CPLD 等可编程逻辑器件进行设计而产生的程序、文档和数据。

[GB/T 33781—2017，定义 3.1.5]

3.3

软件安全性 software safety

软件运行不引起系统事故的能力。

3.4

软件失效 software failure

软件系统丧失完成规定功能能力的事件。

3.5

安全关键功能 safety critical function

针对特定的危险事件，为达到或保持受控设备的安全状态而实现的功能。

3.6

安全关键可编程逻辑器件软件 safety critical programmable logic device software

具有安全关键功能的可编程逻辑器件软件。

3.7

行波时钟　ripple clock

当前一级时序逻辑的数据输出被用作下一级时序逻辑的时钟输入时的时钟信号。

4　缩略语

下列缩略语适用于本文件：

BDA：双向分析(Bi-Directional Analysis)

CPLD：复杂可编程逻辑器件(Complex Programmable Logic Device)

DCM：数字时钟管理(Digital Clock Manager)

FMEA：故障模式及影响分析(Failure Mode and Effects Analysis)

FPGA：现场可编程门阵列(Field Programmable Gate Array)

FTA：故障树分析方法(Fault Tree Analysis)

HDL：硬件描述语言(Hardware Description Language)

I/O：输入/输出(Input/Output)

IP：知识产权(Intellectual Property)

PLDS：可编程逻辑器件软件(Programmable Logic Device Software)

PLL：锁相环(Phase Locked Loop)

VHDL：超高速集成电路硬件描述语言(Very high speed integrated circuit Hardware Description Language)

5　总则

5.1　PLDS 安全性设计

PLDS 安全性贯穿于 PLDS 全生存周期过程，宜与 PLDS 生存周期过程活动紧密结合。可通过下列过程，保证 PLDS 设计的安全性：

a)　系统需求分析：

1)　明确系统/分系统中应重点防范的系统危险事件。

2)　根据系统/分系统规格说明和系统/分系统设计说明开展系统级安全性分析，确定 PLDS 的系统级安全性要求。

3)　明确提出 PLDS 的安全性要求，并完全覆盖系统/分系统规格说明和系统/分系统设计说明中的相关要求。

4)　明确安全等级。系统人员根据系统危险分析结果以及行业相关规定，确定 PLDS 的安全性等级。

5)　对于安全关键 PLDS，安全性需求宜给出重点防范的系统危险事件、失效容限以及安全性保障水平等要求。

6)　给出必要的检错、纠错和容错要求。

7)　对于安全关键 PLDS，宜提出失效模式以及规避失效的策略。

8)　根据给出的系统危险事件及失效模式，确定 PLDS 的安全关键功能。

b)　软件需求分析：

1)　根据系统需求分析时给出的危险事件及失效模式，进一步确认 PLDS 的安全关键功能。

2)　进一步分析系统危险事件及失效模式，根据需要扩充 PLDS 的安全关键功能。

3)　明确应完成的规避失效风险的技术措施。

4） 落实系统需求分析时给出的检错、纠错和容错要求。
5） 对于安全关键 PLDS，宜使用故障模式及影响分析、故障树分析等方法进行安全性分析。
6） 完全覆盖系统需求分析时提出的安全性要求。
7） 衍生的安全性要求宜反馈到系统需求分析过程，并进一步分析其对于安全性的影响。

c） 设计和实现：
1） PLDS 设计和实现覆盖软件需求分析时给出的所有安全性要求及措施。
2） 依据设计准则或编码标准开展 PLDS 设计和实现。
3） 根据可编程逻辑器件的安全关键功能，确定 PLDS 的安全关键部件和单元。
4） 对安全关键部件和单元进行安全性分析和测试，测试宜覆盖所有的安全性要求。
5） 配置项级别宜明确防止错误扩大化的措施，如数据处理时前一帧错误数据不会影响后续正常数据的处理。
6） 衍生的安全性要求宜反馈到需求分析过程，并进一步分析其对于安全性的影响。

5.2 PLDS 更改

确定 PLDS 继承性，对已纳入配置管理的受控 PLDS 的更改宜进行影响域分析，并分析 PLDS 更改对系统安全的影响，重点关注 PLDS 更改对时序关系的影响。

5.3 PLDS 外购、外协或重用

安全关键软件采用外购、外协软件或重用软件时，重点关注：

a） 决定重用某 PLDS 或使用 IP 核来完成安全关键功能之前，确定其适用性，并充分分析其安全性影响。在 PLDS 开发过程中，对重用 PLDS 产品及 IP 核进行安全性分析和评价，并对其进行验证，确定不存在不可接受的安全性风险。
b） 外协 PLDS 产品的需方对外协产品的安全性负责，对外协产品的开发过程进行监控，并对外协产品进行安全性分析和评价。

6 需要考虑的因素

6.1 系统需求分析

在系统需求分析中，分析系统的结构、功能、性能需求、工作环境、实际外部接口时序（考虑外部电路对信号延时的影响）等对 PLDS 的设计需求，需要明确的内容包括：

a） 应遵循的相关安全性标准。
b） 编程语言建议选用 VHDL 或 Verilog HDL，使用 Verilog HDL 宜遵循 GB/T 18349 中要求。
c） 继承性要求。
d） 可编程逻辑器件的运行环境。
e） 可编程逻辑器件的开发环境。
f） 可编程逻辑器件的功耗要求。
g） 可编程逻辑器件芯片规格。确认选用的可编程逻辑器件的芯片等级、速度等级、设计资源数、工作频率、封装、抗空间辐照等指标满足要求。
h） 系统分配给 PLDS 功能的合理性分析。分配的软件任务复杂度不宜超出可编程逻辑器件的能力范围。
i） 使用片上可编程系统要求。若使用片上可编程系统，按软件相关标准分析处理器软件的安全性要求。
j） 接口和信号要求。给出所有接口和信号描述，明确上电及复位后接口信号状态和管脚绑定

要求。

k) 软件可编程要求。针对与软件配合工作的可编程逻辑器件，明确软件对 PLDS 的操作要求、操作时序以及接口协议，包括可编程寄存器名称、地址、复位状态、读/写操作等。

l) IP 核复用要求。对 IP 核进行安全性分析、评价及验证，确定其不存在不可接受的安全性风险。

m) 安全性设计要求。如给定的错误情况如何处理。

n) 余量要求。包括时钟频率和可编程逻辑器件逻辑资源、管脚资源使用等。

o) 如有抗空间辐照设计要求，对有单粒子效应敏感的静态随机存储器型可编程逻辑器件宜提出抗单粒子效应防护设计要求，如采用三模冗余设计、纠/检错编码设计、动态刷新等设计方法。

6.2 软件需求分析

6.2.1 安全性需求的来源

宜正确、完整地追踪安全性需求来源。PLDS 安全性需求来源可分为通用的安全性需求和专用的安全性需求，其中：

a) 通用的安全性需求一般为某一应用领域或多个类似的项目共同的安全性需求；

b) 专用的安全性需求一般为所属 PLDS 系统特定的安全性需求，是由系统向下传递的，或者是系统初步危险分析结果中危险原因与 PLDS 相关的安全性需求，其中也包括开发过程中新识别出再由 PLDS 系统采纳下达的安全性需求。

6.2.2 安全性需求分析

系统特定的 PLDS 安全性需求的开发与分析宜依据系统安全性需求、环境需求、标准、项目专用规范、工具或者设施需求和接口需求，查找安全关键功能，确定安全性需求。PLDS 的安全性分析方法参见附录 A。

6.2.3 安全关键功能的安全保证措施

安全关键功能的安全保证措施如下：

a) 安全运行模式、运行状态与安全条件：

1) PLDS 安全性需求包括有效的运行模式或状态，以及禁止和不适用的模式或状态，宜避免 PLDS 进入禁止或不适用的模式或状态。

2) 在整个软件需求中查找是否存在可能导致不安全状态的条件和潜在的失效隐患，如不按顺序、错误的事件，不适当的量值，不正确的极性，无意的命令，环境干扰造成的错误，以及控制失效模式之类的条件。对所有的不安全状态的条件和潜在失效隐患，制定适当的响应要求。

3) 针对状态机需求，分析确认所需状态机的状态完整性、状态转换完整性、输入和输出变量完整性、初始状态等。

b) 容错和容失效：

1) 明确系统是否能容错或容失效，或两者兼有。

2) 采用容错机制防止微小的差错造成失效。

3) 安全关键功能宜单独划分模块，避免受其他模块干扰。

注 1：系统中的安控处理模块宜单独分出，并根据安全级别可选择热备份、冷备份、三冗余备份等容错或容失效方式。

c) 接口：

1) 对接口进行分析，分析接口出错方式及出错概率，并以此来确定通信方法、数据编码、错误

检查、同步方法以及校验和纠错码等。

注2：接口错误检查或者纠正措施适用于接口的出错概率。如，根据安全级别考虑协议是否满足安全性要求，包括帧头和帧尾不能为FF和00等特殊字符、帧头和帧尾不能相同、校验和、纠错码、起始位、停止位等；协议接收时是否做安全处理，包括是否有效判断帧头、帧尾、校验位和数据长度、数据间隔是否满足时序要求、错误帧是否正确处理等；若为异步传输，需考虑异步信号是否进行有效判断；若为同步传输，需考虑是否有使能信号，时钟、数据和使能信号的布线延时是否影响同步。

2） 明确接口时序要求，PLDS与外设通信时，宜考虑通信时序是否满足芯片手册要求，如数据建立、保持时间等。若时序以时钟为基准，为保证通信正常，还宜考虑PLDS主时钟是否能产生外设驱动时钟，若不能产生需增加外部时钟。

3） 明确实时性要求，必要时增加缓存，避免PLDS与其他外设通信过程中数据丢失。

d） 数据：

1） 定义软件所使用的各种数据，包括外部输入输出数据及内部生成数据，列出这些数据的清单，说明对数据的约束，规定数据采集要求，说明被采集数据的特性、要求和范围。

2） 对重要的数据在使用前后进行检查，如数据值域检查。

3） 建立数据字典，说明数据的来源、处理及目的地。

e） 抗空间辐照：

根据系统安全等级要求，评估抗空间辐照能力，确定可编程逻辑器件的使用及抗空间辐照实现方式，如选用的可编程逻辑器件对单粒子效应敏感，宜基于器件抗单粒子阈值、单粒子环境作用模型及数据、使用情况及可编程逻辑器件的利用率，估算可编程逻辑器件单粒子翻转截面，提供给系统和分系统，明确抗空间辐照实现方式，典型的抗单粒子翻转方式包括PLDS的三模冗余、配置刷新、硬件加固等。

f） 工况：

明确可编程逻辑器件实际工作时的内核温度和电压范围，考虑可编程逻辑器件在最大、最小、典型三种工况下的时序收敛性。

g） 规模：

1） 根据PLDS功能、数据、I/O接口数量及速率、吞吐量、外部时钟等，估计可编程逻辑器件规模和资源占用情况。

2） 根据规模选择适合的可编程逻辑器件，一方面要避免资源浪费，造成功耗、体积等不必要消耗；另一方面要避免资源不足，导致可编程逻辑器件逻辑功能或时序实现不满足要求。

3） 充分考虑可编程逻辑器件的寄存器、I/O接口、存储器、全局时钟资源、IP核等片上资源是否满足要求。

6.3 设计和实现

6.3.1 系统设计

系统设计需要考虑的因素如下：

a） 系统内各软件与PLDS、PLDS之间、PLDS与外设的时序配合，避免出现时序不匹配的情况。

b） 可编程逻辑器件与外围芯片的延时。

c） 宜采用以下方式开展PLDS层次结构设计：

1） 采用层次化设计方式开展PLDS设计，以提高代码的可读性。

2） 结构的层次不宜太深，一般不要超过5层。

3） 顶层模块仅包含对其下层模块的组织和调用。较为合理的顶层模块由输入输出管脚声明、模块的调用与实例化、全局时钟资源、全局置位/复位、三态缓冲和一些简单的组合逻辑等构成。

4） 不宜建立子模块间跨层次的接口。

5） 合理划分子模块，综合考虑子模块的功能、结构、时序、复杂度等多方面因素。

d） 组合逻辑模块的输出信号不宜直接或通过层次关系间接连接到同一模块的输入端口。

注：由于组合逻辑模块的输出信号再重入会产生组合逻辑反馈环，这种结构的输入端有任何变化都有可能使输出值立刻改变，此时就会造成毛刺的产生，导致时序的严重混乱。

e） PLDS 设计中，宜遵循面积和速度的平衡与互换原则。

6.3.2 跨时钟域设计

跨时钟域设计需要考虑的因素如下：

a） 避免组合逻辑信号跨时钟域传递，对跨时钟域信号进行同步处理；

注 1：跨时钟域信号包括异步信号进入时序电路和不同时钟域的信号传递，同步处理方式可采用多次寄存、双口存储器、先进先出队列和握手等。

b） 将组合逻辑模块的输出信号寄存后再输出；

注 2：将组合逻辑从一个时钟域连接到另一个时钟域时，组合逻辑产生的毛刺会传播到下一个时钟域，将组合逻辑模块的输出信号寄存后再输出，可消除毛刺的跨时钟域传播。

c） 采用同步设计，避免使用异步设计。

6.3.3 冗余设计

在资源允许的情况下，宜根据相应的安全关键等级对可编程逻辑器件的关键信号、数据等采用三模冗余的设计方法。

注：电磁、空间辐照等干扰会引起 FPGA 位翻转，从而影响软件功能。

6.3.4 余量设计

余量设计需要考虑的因素如下：

a） PLDS 的设计宜满足系统的资源余量要求，如无明确的系统要求，宜保留 20％的资源余量；

b） PLDS 工作的时序宜满足系统的时序余量要求，如无明确的系统要求，宜保留 20％的时序余量。

6.3.5 编码

编码需要考虑的因素如下：

a） 避免在同一进程中对同一信号多次赋值或同一时刻有多个信号驱动同一端口。

b） 避免存在冗余代码，提高代码的可读性和可维护性。

c） if 条件判断包含所有条件分支，case 包含 default 分支。

d） 在 Verilog HDL 中正确使用阻塞赋值与非阻塞赋值：

1） 对组合逻辑建模宜采用阻塞式赋值。

2） 对时序逻辑建模宜采用非阻塞式赋值。

3） 宜采用多个 always 块分别对组合和时序逻辑建模。

e） 进程中的敏感列表完整正确。

注 1：进程的敏感列表不完整会导致仿真和综合后实现不一致，甚至功能实现错误。

f） 在 VHDL 中正确使用信号和变量：

1） 变量只能在进程语句、函数语句和子程序结构中使用，它是一个局部量。

2） 变量的赋值是立即生效的。信号是电路内部硬件连接的抽象，信号的赋值在进程结束后才生效。

3） 如果在一个进程中对同一信号多次赋值只有最后的值有效，变量的赋值立即生效，在赋新

值前保持原来的值。

g) 条件判断的表达式中宜使用逻辑运算符。

h) 宜将所有寄存器、输出信号初始化为确定值，避免系统上电后出现不确定状态。

i) 在设计过程中宜避免使用锁存器。

注 2：任何对数据的干扰都会直接反映到锁存器的输出，从而降低对毛刺的过滤能力，使用对数据干扰有较好过滤能力的触发器代替锁存器。

6.3.6 时钟设计

时钟设计需要考虑的因素如下：

a) 在资源允许的条件下，宜使用由可编程逻辑器件全局时钟输入管脚驱动的全局时钟；

注 1：全局时钟资源能够提供器件中最短的时钟到输出的延时，是最简单和最可预测的时钟。

b) 合理使用 PLL 和 DCM；

c) 避免使用组合逻辑产生的时钟；

注 2：由于组合逻辑容易产生毛刺，引起触发器的错误翻转，影响 PLDS 设计的安全性，在需要使用组合逻辑产生时钟时，可以采用时钟使能的方式实现。

d) 不宜在时钟路径上插入反相器或缓冲器；

注 3：在时钟路径上插入反相器或缓存器会增加时钟信号的偏移，引起时序错误。

e) 时钟信号不宜再汇聚；

注 4：再汇聚路径将导致时钟路径上发生时序冲突。

f) 不宜使用行波时钟；

注 5：行波时钟即一个触发器的输出用作另一个触发器的输入，在行波链上各触发器时钟之间会产生较大的时钟偏移，降低系统的实际速度，并且有可能超出最坏情况下的建立时间、保持时间。

g) 不宜对时钟信号进行三模冗余设计；

注 6：对时钟进行三模，不但不利于时序分析，而且在设计中由于走线延时的不确定性使设计中产生多路跨时钟域电路。

h) 避免将寄存器输出的信号直接作为时钟使用。

6.3.7 接口及通信设计

接口及通信设计需要考虑的因素如下：

a) 对输入接口进行满足系统要求的滤波和抗干扰设计；

b) 数据传输和采集：

1) 在确定数据传输前对通信信道进行测试；

2) 确定数据采集分辨率和外部数据精度相适应；

3) 确定数据采集频率和外部数据变化率相适应；

4) 接口时序满足芯片手册设计要求；

5) 关键的通信设置校验或握手机制；

6) 对通信异常情况进行保护处理，如串口校验位、停止位错误、读写地址错误等；

7) 接口通信协议保持一致；

8) 接口通信对接收数据进行有效性判断；

c) 确定未使用的输出接口状态满足安全性要求；

d) 确定输出接口特性满足系统要求，如值域、时序等；

e) 确定输入输出接口的电气相互兼容；

f) 调试端口在正式产品中宜作相应处理，如保留、禁用或删除。

6.3.8 配合存储器相关设计

配合存储器相关设计需要考虑的因素如下：

a) 外部存储器的接口时序满足要求，宜考虑时序余量设计、读写地址异常的情况；

b) 避免外部存储器的读写冲突；

c) 对存储资源的设计余量进行分析，考虑极限情况下的容量满足情况。

6.3.9 复位设计

复位设计需要考虑的因素如下：

a) 同一复位信号使用唯一的有效电平，保证系统模块同时复位；

b) 保证一定长度的复位有效时间，确保复位信号能被正确采样；

注 1：对于同步复位，复位信号的有效时长至少为 1 个采样时钟周期，对于异步复位，能保证复位释放时能被时钟正确采样。

c) 异步复位采用同步释放的方式；

注 2：异步复位采用同步释放的方式可避免异步复位释放时的亚稳态。

d) 设计中使用同一个复位域。

注 3：存在多个复位域可能影响设计的初始状态，引起信号的冲突。

6.3.10 状态机设计

状态机设计需要考虑的因素如下：

a) 状态机设置合法初始状态；

注 1：避免出现状态机死锁。

b) 对状态机的无效状态进行适当处理；

注 2：状态机设计时可能存在一些非法的或无关的状态，充分考虑各种可能出现的状态以及一旦进入非法状态后可以强迫状态机进入合法状态。

c) 状态机编译模式选择安全模式；

注 3：在综合时对综合属性进行设置，编译模式选择安全模式确保状态机综合后包含对无效状态的处理。

d) 状态机的跳转不能仅依赖外部信号；

e) 根据状态机对速度、资源占用量等方面要求，合理使用状态机编码；

f) 宜避免状态机寄存器的复制。

6.3.11 IP 核使用

宜对 IP 核使用进行版本控制，在 IP 核的调用语句后添加版本号注释。

6.3.12 综合设计

综合设计需要考虑的因素如下：

a) 将关键路径和非关键路径、组合逻辑和时序逻辑划分为不同进程实现；

b) 对关键功能的综合后网表进行确认，以确定综合前后逻辑等价，避免综合过程的不确定性。

6.3.13 管脚约束

管脚约束设计需要考虑的因素如下：

a) 对未使用的输入管脚进行上拉设置；

b) 宜将未使用管脚定义为输出，并进行上拉处理；

c) 避免可编程逻辑器件硬件管脚间的相互干扰，对于存在耦合干扰的信号可通过调整管脚位置、增加地线隔离等方式处理；

d) 上电过程中如信号对状态有确定要求，宜通过外部上、下拉确定。

附 录 A
（资料性附录）
可编程逻辑器件软件安全性分析方法

A.1 概述

常用的可编程逻辑器件软件安全性分析方法包括 FMEA、FTA 和 BDA。这三种方法各有特点，在实际应用中可根据需要，综合应用这三种方法，更加全面、有效地分析可编程逻辑器件软件安全性。

A.2 可编程逻辑器件软件 FMEA

A.2.1 概述

可编程逻辑器件软件 FMEA 分析方法通过识别可编程逻辑器件软件故障模式，分析造成的后果，研究分析各种故障模式产生的原因，寻找消除和减少其有害后果的方法，以尽早发现潜在的问题，并采取相应的措施，从而提高可编程逻辑器件软件的安全性。

A.2.2 可编程逻辑器件软件故障模式、故障影响及严酷度

A.2.2.1 可编程逻辑器件软件故障模式

可编程逻辑器件软件故障模式指可编程逻辑器件软件故障发生的不同方式，例如功能异常、接口异常等。其中可编程逻辑器件软件故障指可编程逻辑器件软件在运行中丧失了全部或部分功能、出现偏离预期的正常状态的事件。可编程逻辑器件软件故障归根结底是由可编程逻辑器件软件中潜藏的缺陷引起的。

可编程逻辑器件软件故障模式的分析是进行可编程逻辑器件软件 FMEA 的基础。只有将被分析对象的所有可能的故障模式尽可能全面地分析出来，才能采取相应措施改进设计，防止故障发生。因此故障模式的分析是否全面合理决定了可编程逻辑器件软件 FMEA 的分析效果，是整个分析过程中最为关键的一步。

在进行可编程逻辑器件软件 FMEA 时，宜根据被分析可编程逻辑器件软件的不同特点，选择合适的故障模式。可编程逻辑器件软件设计中常见的典型故障模式，见表 A.1。

表 A.1 可编程逻辑器件软件设计中常见的典型故障模式

序号	类别	具体故障模式
1	典型功能类	串口通信功能典型故障模式
		存储控制功能典型故障模式
		复位功能典型故障模式
		FLASH 存储器控制功能典型故障模式
		数模转换控制功能典型故障模式

表 A.1（续）

序号	类别	具体故障模式
2	典型接口类	中央处理器接口典型故障模式
		低电压差分信号接口典型故障模式
		串行外设接口典型故障模式
		控制器局域网络总线接口典型故障模式
3	典型硬件环境类	模拟单粒子翻转故障模式
	……	……

A.2.2.2 可编程逻辑器件软件故障影响及严酷度

可编程逻辑器件软件故障影响指可编程逻辑器件软件故障模式对可编程逻辑器件软件系统的运行、功能或状态等造成的后果。例如可编程逻辑器件软件故障会影响任务的完成或造成设备的损坏等。

可编程逻辑器件软件故障影响严酷度指可编程逻辑器件软件故障模式所产生的后果的严重程度。最严重的后果可能是导致人员死亡、对环境造成灾难性破坏，而轻微的后果仅降低使用的舒适性、方便性等。

可编程逻辑器件软件故障影响及其严酷度的确定可以为故障模式改进措施的制定提供依据。对于产生后果严重的故障模式，采取有效措施加以改进，以避免危险事件的发生。

A.2.3 可编程逻辑器件软件 FMEA 分析步骤

A.2.3.1 概述

可编程逻辑器件软件 FMEA 分析一般包括系统定义、可编程逻辑器件软件故障模式分析、可编程逻辑器件软件故障原因分析、可编程逻辑器件软件故障模式影响及严酷度分析、制定改进措施等几个步骤，见图 A.1。

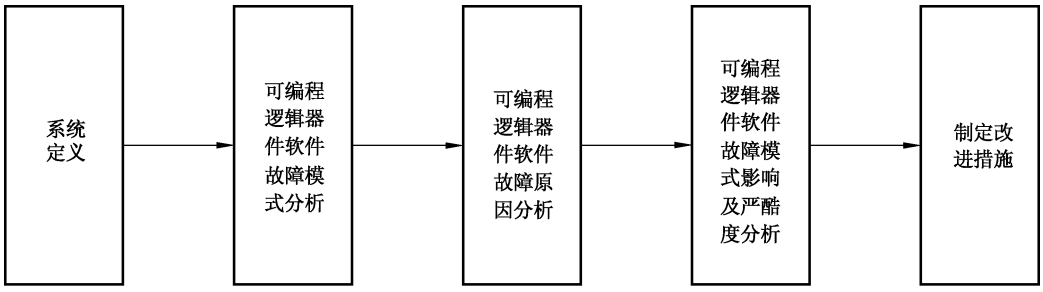

图 A.1 可编程逻辑器件软件 FMEA 分析步骤

A.2.3.2 系统定义

系统定义的主要目的为确定可编程逻辑器件软件 FMEA 的分析级别和分析对象，以确定分析的重点。在系统定义中首先说明系统的主要功能、次要功能、用途、系统的约束条件和故障判据等。系统定义还包括系统工作的各种模式的说明、系统的环境条件，以及软、硬件配置等。如果受时间或经费等因素的影响无法对整个可编程逻辑器件软件系统进行全面的分析，可在分析前确定分析的重点。通过识别对系统功能和安全性影响较大的危险事件，确定对上述危险事件的出现有直接或间接关系的功能模

块、可编程逻辑器件软件部件等，作为可编程逻辑器件软件 FMEA 分析的重点。

A.2.3.3 可编程逻辑器件软件故障模式分析

可编程逻辑器件软件故障模式为可编程逻辑器件软件故障的表现形式。可编程逻辑器件软件故障模式分析的目的为针对每个被分析的可编程逻辑器件软件单元，找出其所有可能的故障模式；针对每个分析对象，确定其潜在的故障模式。

A.2.3.4 可编程逻辑器件软件故障原因分析

针对每个故障模式，分析其所有可能原因。可编程逻辑器件软件故障的原因是可编程逻辑器件软件中潜藏的缺陷，一个可编程逻辑器件软件故障的产生可能是由一个缺陷引起的，也可能是由多个缺陷共同作用引起的。在进行故障原因分析时尽可能全面地分析所有可能的可编程逻辑器件软件缺陷，为制定改进措施提供依据。

A.2.3.5 可编程逻辑器件软件故障模式影响及严酷度分析

分析每个故障模式对局部、高一层次，直至整个系统的影响，以及故障影响的严重性。分析故障影响及其严重性的目的为识别可编程逻辑器件软件故障所造成的后果的严重程度，以便按照优先级为不同严重等级的故障制定改进措施。

可参照表 A.2 所示的方法确定故障影响的严酷度类别。需要指出的是，严酷度类别仅是按故障模式造成的最坏的潜在后果，且仅对系统层次的影响程度进行确定的。严酷度类别划分有多种方法，但对同一可编程逻辑器件软件进行 FMEA 时，其定义保持一致。

A.2.3.6 制定改进措施

根据上述分析得到的故障产生的原因及影响的严重性等，确定出需要采取的改进措施。改进措施主要有两种途径，一是修改可编程逻辑器件软件需求、设计或编码中的缺陷，增加软件防护措施；二是增加硬件防护措施。

进行可编程逻辑器件软件 FMEA 时，填写 FMEA 表。FMEA 表宜完整地体现分析的目的和取得的成果。表 A.2 是一张 FMEA 表的示例，表中记录了 FMEA 的分析结果。

表 A.2 FMEA 表示例

序号	单元	功能	故障模式	故障原因	故障影响		严酷度	备注
					局部影响	最终影响		
单元序号	功能模块名称	单元执行的主要功能	与功能、性能有关的所有故障模式	导致故障模式发生的可能原因	根据故障影响分析结果，依次填写 FPGA 故障模式的局部影响和最终影响		按故障最终影响严重程度确定	记录对其他栏的注释和补充说明

A.3 可编程逻辑器件软件 FTA

A.3.1 概述

可编程逻辑器件软件 FTA 是一种自顶而下的可编程逻辑器件软件安全性分析方法，即从可编程逻辑器件软件系统不希望发生的事件（顶事件），特别是对人员和设备的安全产生重大影响的事件开始，向下逐步追查导致顶事件发生的原因，直至基本事件（底事件）。可编程逻辑器件软件 FTA 的分析结

果可以用来指导可编程逻辑器件软件安全性设计，确定可编程逻辑器件软件测试的重点和内容，使可编程逻辑器件软件安全性得到更充分的保证。

A.3.2 可编程逻辑器件软件故障树的建立

A.3.2.1 概述

建立可编程逻辑器件软件故障树的目的为找出导致顶事件发生的直接原因，直至最底层的根本原因。只有建立了故障树，才能在此基础上进行定性、定量分析，故障树的建立是可编程逻辑器件软件FTA中最基本同时也是最关键的一项工作，故障树建立的完善程度直接影响定性分析和定量分析的准确性。建立可编程逻辑器件软件故障树通常采用演绎法，见图A.2。

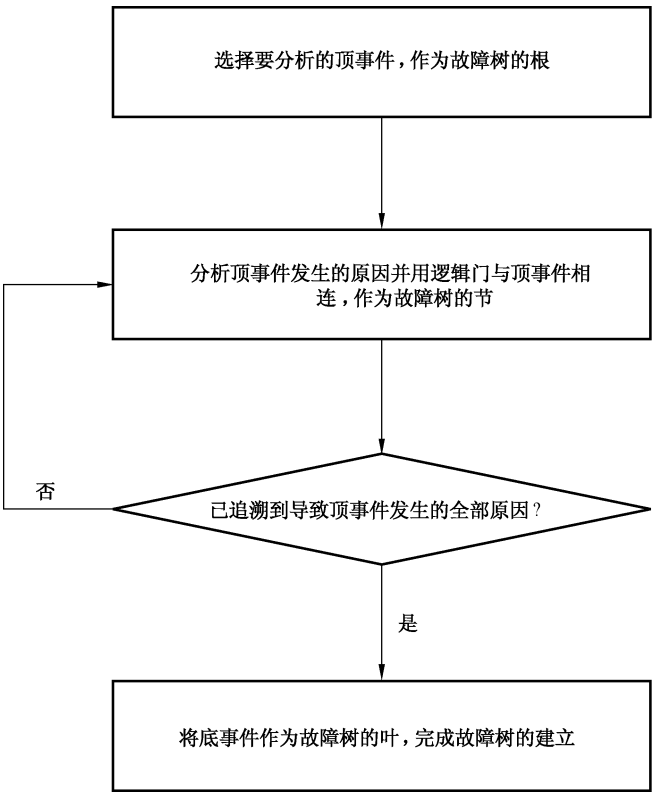

图 A.2 可编程逻辑器件软件故障树建立方法

A.3.2.2 顶事件的确定

在故障树最顶层的顶事件是系统不期望的故障事件，可依据以下信息确定可编程逻辑器件软件故障树的顶事件：

a) 可编程逻辑器件软件危险分析提供的信息。危险分析是指对系统设计、使用、维修及与环境有关的所有危险进行系统化分析，以判别和评价危险或潜在的危险状态、可能相关的危险事件及其后果的危害性。在可编程逻辑器件软件开发的各个过程，通过危险分析可识别出对系统可能造成的潜在危险，可以此作为故障树的顶事件进行可编程逻辑器件软件FTA。

b) 可编程逻辑器件软件FMEA的分析结果。在可编程逻辑器件软件FMEA中，通过对故障模式的影响分析可以得出对可编程逻辑器件软件系统产生的不利影响。在进行可编程逻辑器件软件FTA时，可选取影响严重性较大的故障事件作为可编程逻辑器件软件故障树的顶事件进行分析。

c） 可编程逻辑器件软件需求规格说明等文档中提出要求等。

A.3.2.3 底事件的确定

在故障树最底层的底事件是导致顶事件发生的根本原因。可编程逻辑器件软件故障树分析的目的就是要采取措施避免底事件的发生，从而降低顶事件的发生概率。有些底事件可以独立地引发顶事件，有些底事件按照一定的逻辑关系共同引发顶事件。

可编程逻辑器件软件故障树的建立可以伴随着可编程逻辑器件软件开发过程而逐步深入。在可编程逻辑器件软件需求分析时，可以利用故障树分析可编程逻辑器件软件需求中可能导致顶事件发生的原因，即需求的不完善之处。随着开发过程的深入，可以通过对可编程逻辑器件软件设计的进一步分析，对可编程逻辑器件软件故障树加以扩展，直至最底层的可编程逻辑器件软件单元，甚至可以分析到可编程逻辑器件软件代码语句级。

A.3.3 可编程逻辑器件软件故障树的定性分析

可编程逻辑器件软件故障树定性分析的目的为找出关键性的导致顶事件发生的原因，指导可编程逻辑器件软件安全性设计以及可编程逻辑器件软件测试，从而提高可编程逻辑器件软件的安全性。可编程逻辑器件软件故障树定性分析的常用方法是识别所有最小割集，并对最小割集进行定性比较，对最小割集及底事件的重要性进行排序。

割集是指能引起顶事件发生的底事件的集合，最小割集是指不包含任何冗余因素的割集。如果去掉最小割集中的任何事件或条件，它就不再成为割集。最小割集的求解方法通常是：遇到“与门”增加割集的阶数（割集所含底事件数目），遇到“或门”增加割集的个数。下面以图 A.3 所示的故障树为例，说明进行最小割集求解的方法。

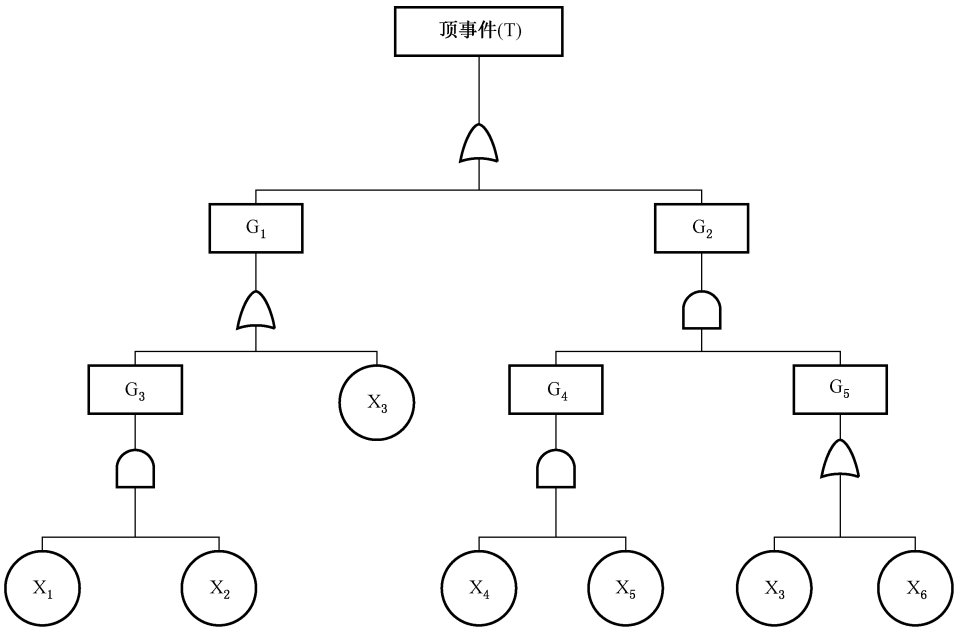

图 A.3 故障树定性分析示例

根据最小割集求解规则，其分析过程见表 A.3。

表 A.3　最小割集求解过程

步骤	1	2	3	4	5	6
过程	G_1	G_3	X_1，X_2	X_1，X_2	X_1，X_2	X_1，X_2
	G_2	X_3	X_3	X_3	X_3	X_3
		G_2	G_2	G_4，G_5	X_4，X_5，G_5	X_4，X_5，X_3
						X_4，X_5，X_6

由表 A.3 最后得出的割集有{X_1，X_2}、{X_3}、{X_4，X_5，X_3}及{X_4，X_5，X_6}。根据最小割集的定义，可得最小割集有 3 个：{X_1，X_2}、{X_3}和{X_4，X_5，X_6}。对最小割集进行定性比较，宜根据最小割集所包含的底事件数目（阶数）排序，在各底事件发生概率比较小，其差别不大的条件下，宜遵循以下原则：

a）　阶数越小的最小割集越重要；

b）　低阶最小割集所包含的底事件比高阶最小割集所包含的底事件重要；

c）　在不同的最小割集中重复出现次数越多的底事件越重要。

在上面的例子中，{X_3}是最重要的最小割集，X_3 是最重要的底事件。

A.3.4　可编程逻辑器件软件故障树的定量分析

可编程逻辑器件软件故障树定量分析的目的为计算或估算故障树顶事件发生的概率。故障树顶事件发生概率可由最小割集的集合来确定，即一次取出一个割集的概率和，减去一次取出两个割集的交集的概率和，加上一次取出三个割集的交集的概率和，依此类推。

注：应用 FTA 时，由于定量分析需要知道每个底事件的发生概率，这在实际应用中往往难以做到。而可编程逻辑器件软件故障树分析的重点在于定性分析，而不是定量分析。可编程逻辑器件软件故障树分析的最重要意义在于，根据分析的结果找出关键性的安全事故发生的原因，指导可编程逻辑器件软件安全性设计以及可编程逻辑器件软件测试，从而提高可编程逻辑器件软件的安全性。

A.4　可编程逻辑器件软件 BDA

A.4.1　概述

可编程逻辑器件软件 FMEA 与可编程逻辑器件软件 FTA 两种技术在单独应用进行可编程逻辑器件软件安全性分析时各有其不足：可编程逻辑器件软件 FMEA 是一种自底而上的单因素故障分析方法，无法完善的表达故障原因之间的各种逻辑关系。此外，其分析结果以表格方式列出，不如故障树的图形化表达方式直观。可编程逻辑器件软件 FTA 是一种自顶而下依照树状结构倒推故障原因的方法，选取顶事件时，可能会遗漏潜在的顶层故障事件，有时候这种影响是关键的。另外，可编程逻辑器件软件 FTA 在分析故障原因时也会有所遗漏，这会影响到底事件的重要度排序，从而影响实施改进措施时轻重缓急的判断。树形结构在描述分析结果方面不如 FMEA 信息详尽。

将这两种分析方法综合应用，形成可编程逻辑器件软件 FMEA 和可编程逻辑器件软件 FTA 相结合的可编程逻辑器件软件 BDA，可以起到优势互补的作用。

根据可编程逻辑器件软件 FMEA 与可编程逻辑器件软件 FTA 综合分析的方向，可将其分为正向综合分析和逆向综合分析两类。正向综合分析是以可编程逻辑器件软件 FMEA 方法为主，辅以可编程逻辑器件软件 FTA；而逆向综合分析则正相反，以可编程逻辑器件软件 FTA 方法为主，辅以可编程逻辑器件软件 FMEA。

A.4.2　正向分析方法

可编程逻辑器件软件 FMEA 与 FTA 正向综合分析原理见图 A.4。从分析对象的故障模式出发，

先由可编程逻辑器件软件 FMEA 分析得到顶层故障影响及其严酷度类别，故障影响可作为可编程逻辑器件软件 FTA 顶事件的来源，并可根据严酷度类别确定 FTA 的分析优先级。也可以分析对象的故障模式作为中间事件，进行可编程逻辑器件软件 FTA 向下分析故障原因。这样，在可编程逻辑器件软件 FMEA 基础上，补充使用故障树分析故障原因，以树形结构图可以更加直观地表达各故障原因之间的逻辑关系，使故障原因的分析更加彻底。

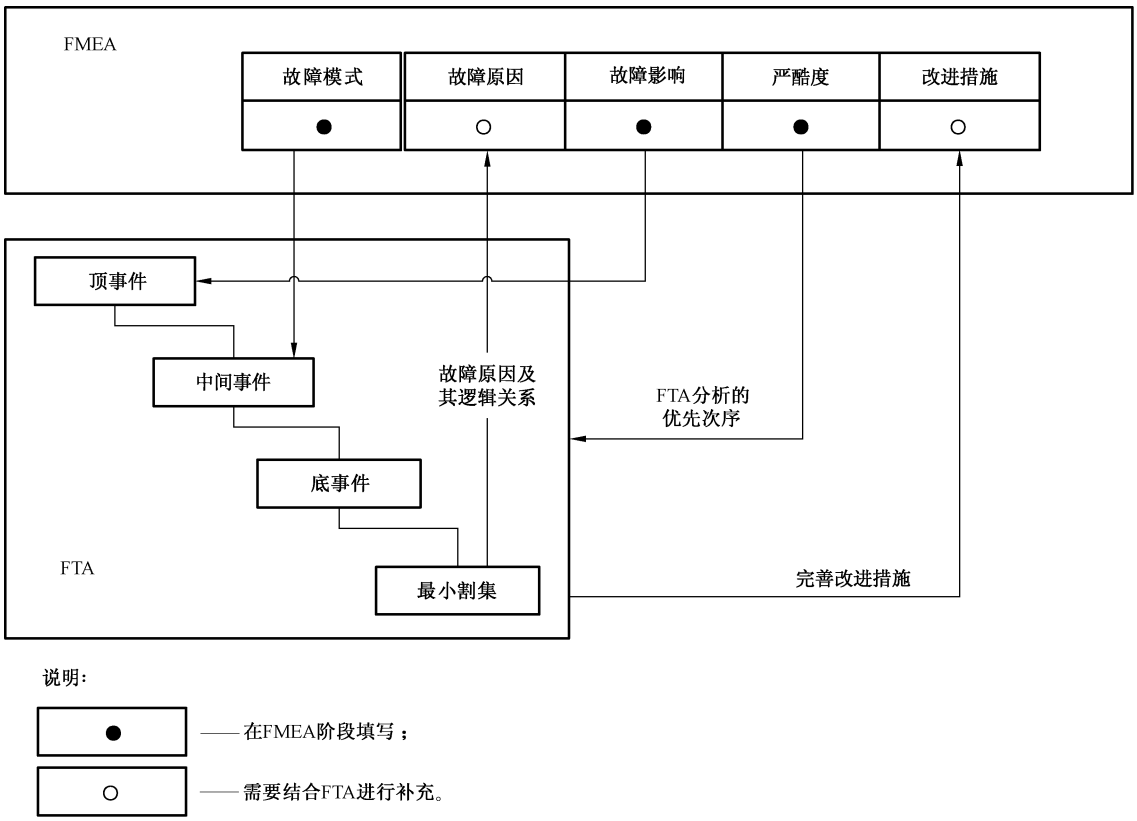

图 A.4　正向综合分析方法原理图

对比单一的可编程逻辑器件软件 FMEA 技术，这种以可编程逻辑器件软件 FTA 为补充的综合分析方法可以更全面直观地分析故障原因，从而在制定改进措施时，能够考虑到多点故障的逻辑关联，提出更为合理的改进建议。例如，对于单点故障需要一一改进，而对于以与门连接的多点故障，只需要选取其中一点改进即可。另外，如果分析过程受到时间和资源的限制，需要适当裁剪、突出重点，对所有故障影响进行可编程逻辑器件软件 FTA 显然不现实。通常，根据故障影响的严酷度高低排序决定实施可编程逻辑器件软件 FTA 的优先次序，同时，改进措施的制定也可以此为依据。由此，综合分析方法可以根据工程需要灵活的放大或缩小实施的工作量。

可编程逻辑器件软件 FMEA 与 FTA 正向综合分析的实施步骤如下：

a）　对所选定的分析对象进行可编程逻辑器件软件 FMEA；

b）　选取严酷度等级高的故障影响作为顶事件进行可编程逻辑器件软件 FTA，对关键的故障影响进行更加深入全面的原因分析；

c）　或者选取故障模式作为中间事件，进行可编程逻辑器件软件 FTA，分析对应故障模式的故障原因；

d）　根据可编程逻辑器件软件 FTA 得出的故障原因及其逻辑关系补充完善可编程逻辑器件软件 FMEA 表格，从而制定更为合理的改进措施。

A.4.3 逆向综合分析

可编程逻辑器件软件 FMEA 与 FTA 逆向综合分析的原理见图 A.5，该方法以可编程逻辑器件软件 FTA 为主，由可编程逻辑器件软件 FTA 识别出导致顶事件发生的根本原因，即底事件，并进行定性定量分析。在此基础上，选取重要的底事件作为分析对象，对其进行可编程逻辑器件软件 FMEA，逐步向上分析可能导致的故障影响。

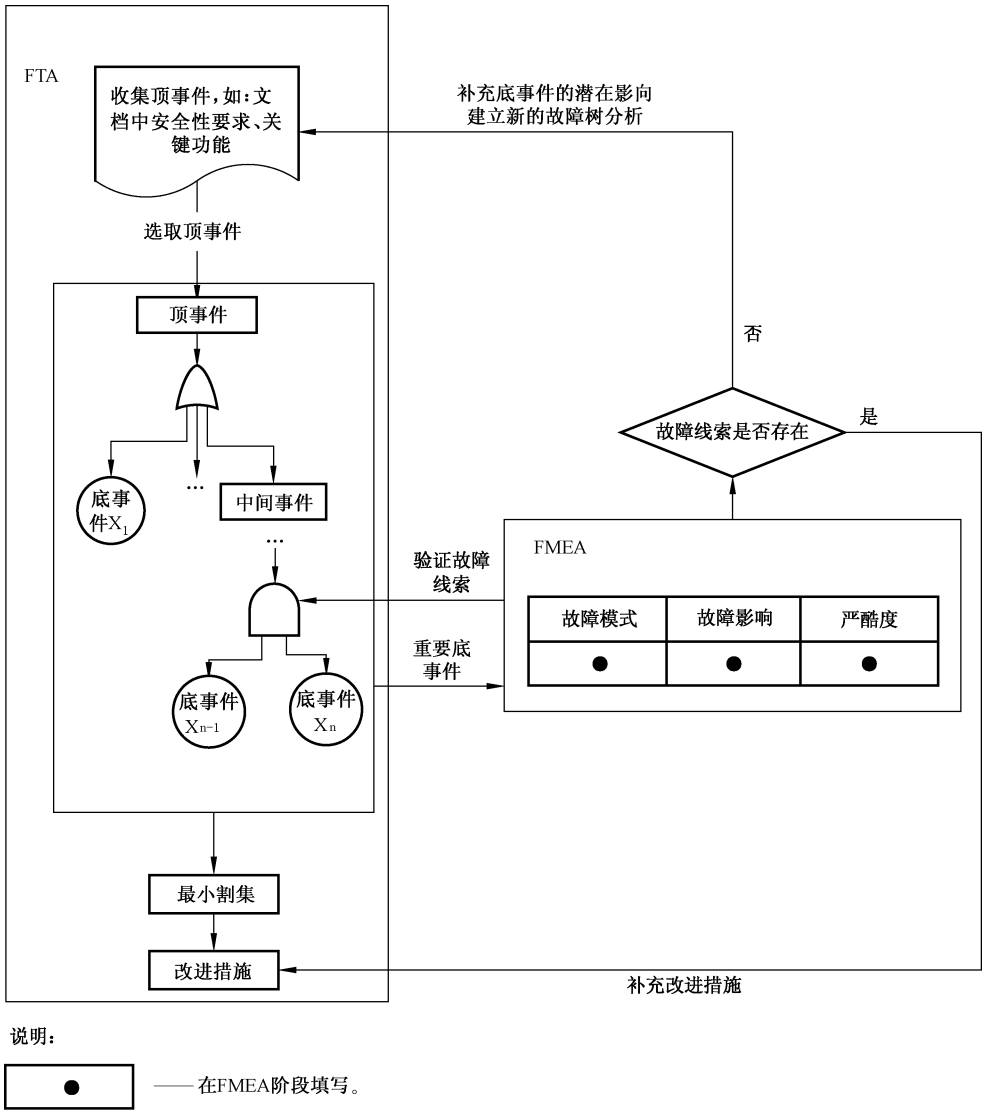

图 A.5 逆向综合分析方法原理图

对比单一的可编程逻辑器件软件 FTA 技术，补充进行的可编程逻辑器件软件 FMEA 能够验证故障线索，即由底事件导致顶事件发生的故障影响传递路径。另一方面，通过可编程逻辑器件软件 FMEA 可能识别出新的潜在故障影响，可增加故障树的顶事件，建立新树进行分析。

可编程逻辑器件软件 FMEA 与 FTA 逆向综合分析的实施步骤如下：

a) 根据可编程逻辑器件软件 FTA 实施步骤选取故障树的顶事件，建立故障树并进行定性定量分析；

b) 根据定性分析结果，选取重要的底事件进行可编程逻辑器件软件 FMEA；

c) 根据可编程逻辑器件软件 FMEA 的分析结果修正故障树，补充完善改进措施；

d) 如果由可编程逻辑器件软件 FMEA 得到的新的顶层故障影响，其严酷度等级较高，则需要以此作为顶事件建立一棵新的故障树进行分析，识别出导致该顶事件的所有可能原因，并制定相应的改进措施。

A.4.4 通信类可编程逻辑器件软件安全性分析实例

为确保安全性分析的完整性，需要对影响可编程逻辑器件软件安全性的关键事件进行重点分析，选用适合可编程逻辑器件软件 FMEA 与 FTA 逆向综合分析的方法：先进行 FTA，由 FTA 识别出导致串口通信故障的根本原因，即底事件。在此基础上，选取这些底事件作为分析对象，对其进行 FMEA，向上分析其可能导致的故障影响。

选取“串口通信异常”作为可编程逻辑器件软件故障树的顶事件进行分析。根据串口通信异常的故障机理，可以分析出导致串口通信异常的直接原因为：可编程逻辑器件串口接收功能模块异常、串口芯片异常或串口输入管脚异常。将它们作为中间事件，继续向下分析，直至识别出导致故障的根本原因为止，得到的故障树见图 A.6。

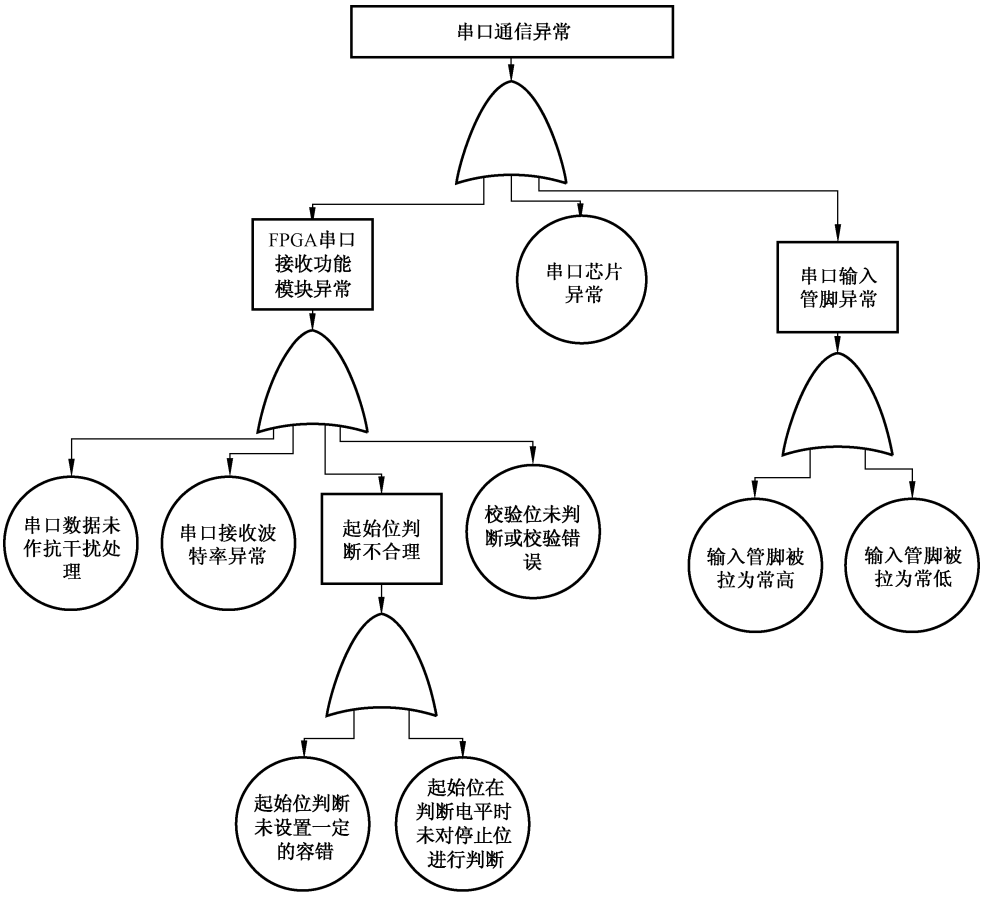

图 A.6 串口通信异常故障树

在对串口通信异常的故障树分析中，得到 8 个最小割集，分别是：{(串口数据未作抗干扰处理)}、{(串口接收波特率异常)}、{(起始位判断未设置一定的容错)}、{(起始位在判断电平时未对停止位进行判断)}、{(校验位未判断或校验错误)}、{(串口芯片异常)}、{(输入管脚被拉为常高)}、{(输入管脚被拉为常低)}，并且均为一阶最小割集，由于它们的故障都将导致顶事件“串口通信异常”，所以有必要对它们进行 FMEA 作进一步分析，以识别可能造成的其他影响。完成的可编程逻辑器件软件 FMEA 表格见表 A.4。

表 A.4　串口通信异常 FMEA 表

序号	单元	功能	故障模式	故障原因	故障影响		严酷度
					局部影响	最终影响	
U1	串口	实现串口数据接收	未接收到有效串口数据	串口输入端被拉为常高	无串口数据	通信无响应	…
				串口输入端被拉为常低	串口数据为常值	通信任务失败	…
				未对接收串口数据未进行抗干扰处理	串口数据无效	通信响应错误	…
			部分有效数据丢失	起始位判断太严格，未设置一定的容错	丢失有效串口数据	通信响应延迟或无响应	…
			接收到错误的串口数据	接收波特率不在协议要求范围内	接收到错误的串口数据	通信响应错误或无响应	…
				未对校验位进行判断或校验错误	接收到错误的串口数据	通信响应错误或无响应	…
				当起始位是判断电平时，未对停止位进行判断	接收到错误的串口数据	通信响应错误或无响应	…

通过对底事件进一步进行可编程逻辑器件软件 FMEA，不仅验证 FTA 中的故障线索（串口通信异常），而且识别出对系统可能导致的其他影响，即导致通信响应错误或无响应等。需要说明的是，在上面的 FMEA 表中，未确定故障影响的严酷度等级，是因为考虑到串口通信故障对系统影响的严重程度很大程度上取决于串口通信模块本身的安全性关键等级，不同安全性关键等级的串口通信模块发生的故障对系统造成的影响严酷度是不同的，需要在不同系统中具体分析。

将可编程逻辑器件软件 FTA 结果中的重要底事件作为可编程逻辑器件软件 FMEA 的分析对象，分析出其所有可能的故障模式，并分析其产生的故障影响，不仅使 FTA 中的故障线索得到验证，同时也识别出对系统可能造成的其他影响，使这些重要的底事件得到进一步深入的分析，避免该故障影响的发生，从而提高系统的安全性。