Application Note

AWR294x 安全诊断库 (SDL) 使用参考

Allen Yin *China Central FAE*

摘要

AWR2944 是德州仪器推出的高性能毫米波雷达芯片，集成了射频前端、信号处理单元和微控制器。该芯片采用双核架构，包含 ARM Cortex-R5F 主控子系统（MSS）和 C66x DSP 信号处理子系统（DSS），支持 76-81GHz 频段，广泛应用于汽车高级驾驶辅助（ADAS）等关键领域。

本文系统介绍了 AWR2944 的安全诊断库（SDL）架构与应用。SDL 实现芯片级故障检测与诊断，包括：数据移植层（DPL）提供硬件抽象；内核比较模块（CCM）实现 CPU 锁步监测；双时钟比较器（DCC）检测时钟漂移；ECC 模块提供存储器错误校正；总线安全模块保护互联总线；存储器循环冗余校验（MCRC）实现数据完整性校验；奇偶校验检测单比特错误；可编程内建自测试（PBIST）执行存储器自测；复位管理追踪系统状态；ROM 校验和验证固件完整性；窗口看门狗（RTI）监控执行流；自测控制器（STC）进行逻辑门电路测试。所有模块通过 ESM（错误信号模块）统一上报安全事件。本文提供 SDL 的模块、API 说明、测试用例和配置流程，为开发符合功能安全要求的雷达应用提供完整安全诊断技术参考。

内容

插图清单

表格清单

商标

所有商标均为其各自所有者的财产。

1 AWR294x 及安全模块介绍

汽车高级驾驶辅助系统利用了各种传感器（毫米波雷达、激光雷达、摄像头以及卫星导航），在汽车行驶过程中随时感知周边环境，进行障碍物体的识别与追踪，通过系统运算分析提供给驾驶员辅助决策信息。其中毫米波雷达具有全天候优秀的测距测速性能，较强抗干扰能力，在 ADAS 系统内得到了广泛应用。

TI 基于 RFCMOS 工艺将毫米波雷达数据链路的中射频电路、压控振荡器（VCO）、模数转换器（ADC）、微控制器（MCU）和数字信号处理器（DSP）集成在单颗芯片内，极大降低了汽车雷达的成本和开发难度。

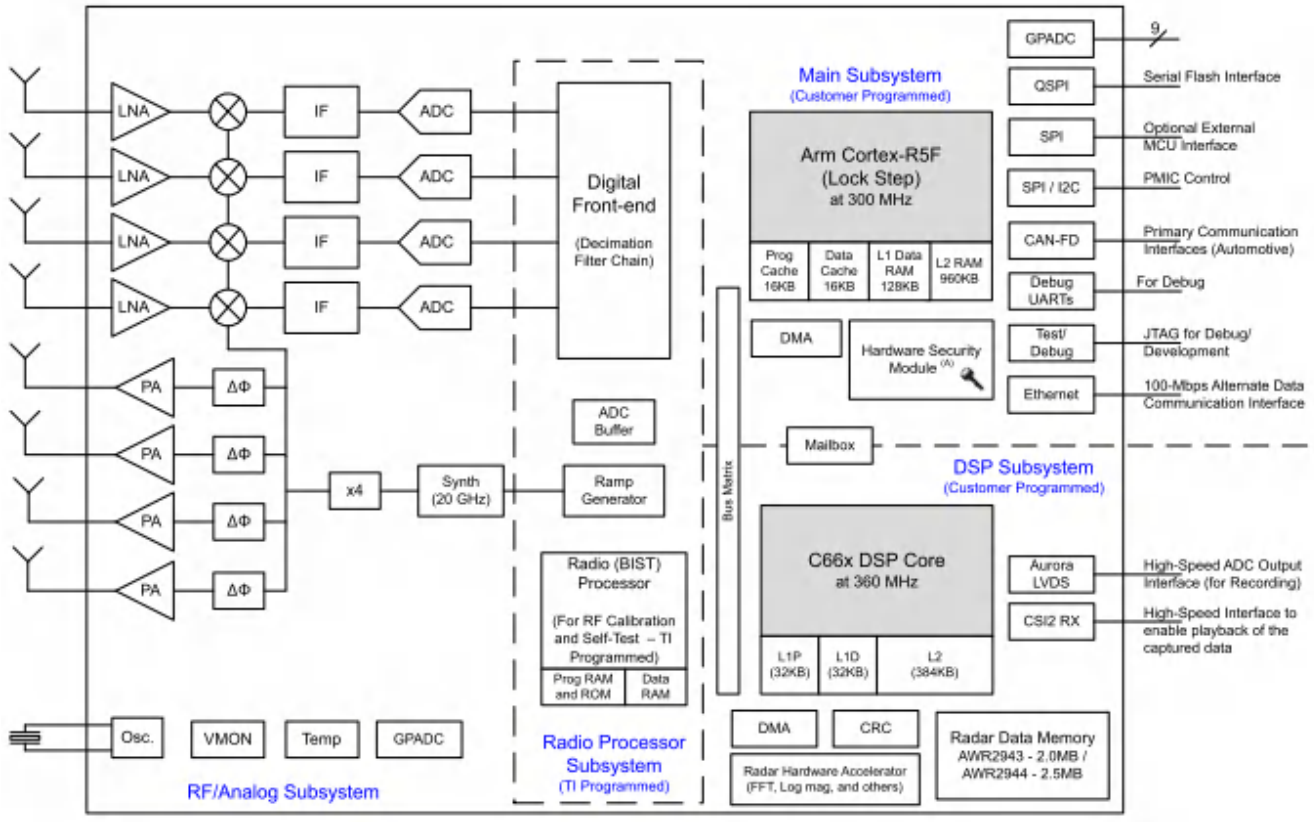

图 1-1. AWR2943/4 系统框图

汽车毫米波雷达芯片的功能安全是构建汽车自动驾驶与主动安全系统的核心基石。作为车载核心传感器的控制中枢，毫米波雷达芯片的输出直接驱动自动紧急制动（AEB）、自适应巡航（ACC）等关乎人身安全的决策控制系统。一旦芯片因随机硬件失效、环境电磁干扰或系统性缺陷导致漏报或误报，将直接引发灾难性的交通碰撞或非预期制动。因此，严格遵循汽车功能安全标准（如 ISO26262），通过在芯片内部部署冗余架构、实时自检及故障容错机制，是确保雷达系统在极端或失效工况下仍能实现安全受控、保障行车生命安全的关键。

AWR2944 的安全模块是芯片功能安全架构的核心，通过软件诊断库（SDL）提供标准化接口，帮助开发者实现符合功能安全标准的应用。在安全系统中，检测随机硬件故障并将系统引导至安全状态至关重要。安全模块集成了多种硬件安全机制，包括存储器 ECC 错误检测与纠正、ESM（错误信号模块）统一监控安全事件、CCM 核心锁步比较、DCC 时钟监测、PBIST 存储器自测、RTI 看门狗等诊断模块。SDL 通过 API 接口封装这些硬件机制，提供错误捕获和安全诊断功能，而错误响应策略则由应用层根据器件安全手册要求实现。该架构支持裸机或 RTOS 环境，系统集成商可利用 SDL 提供的 API 配置 ECC、设置中断处理、执行诊断测试，构建满足 ISO26262 等功能安全标准的雷达应用系统，实现高诊断覆盖率和故障安全响应能力。

2 AWR294x 安全诊断库详解

AWR294x 系列 SoC 为功能安全应用提供了完整的硬件安全机制和软件诊断库（SDL）支持。SDL 作为标准化接口层，封装了芯片内置的各类安全功能，帮助开发者构建符合功能安全标准的应用系统。

在安全关键应用中，及时检测随机硬件故障并将系统引导至安全状态是核心要求。SDL 通过错误捕获模块和安全机制模块实现这一目标，提供的典型硬件安全机制包括：存储器 ECC 错误检测与纠正、ESM（错误信号模块）统一监控安全事件、核心锁步比较、时钟监测、存储器自测等。SDL 以软件 API 形式提供配置接口，例如 ECC 配置 API 和中断设置参考示例，系统集成商可利用这些接口实现软件诊断，满足安全系统目标。

SDL 架构按设备运行模式划分为两大阶段。启动阶段执行自测诊断以验证器件正常运行，并通过 SDL API 初始化连续诊断功能；运行阶段通过 ESM 处理器实时接收连续诊断检测到的错误通知，同时调用 SDL API 执行周期性诊断任务。错误响应策略由应用层根据器件安全手册要求灵活管理。

SDL 采用模块化设计，支持裸机或 RTOS 环境部署，覆盖多个 IP 核的诊断需求。应用程序通过标准 API 与 SDL 交互，并可结合外部模块扩展功能。这种分层架构为开发符合 ISO26262 等功能安全标准的毫米波雷达应用提供了完整、灵活的软件诊断解决方案。(注意： 当前 mmWave Radar SDK 4.7 版本不包含 SDL 及相关代码，需下载 4.6 版本)。

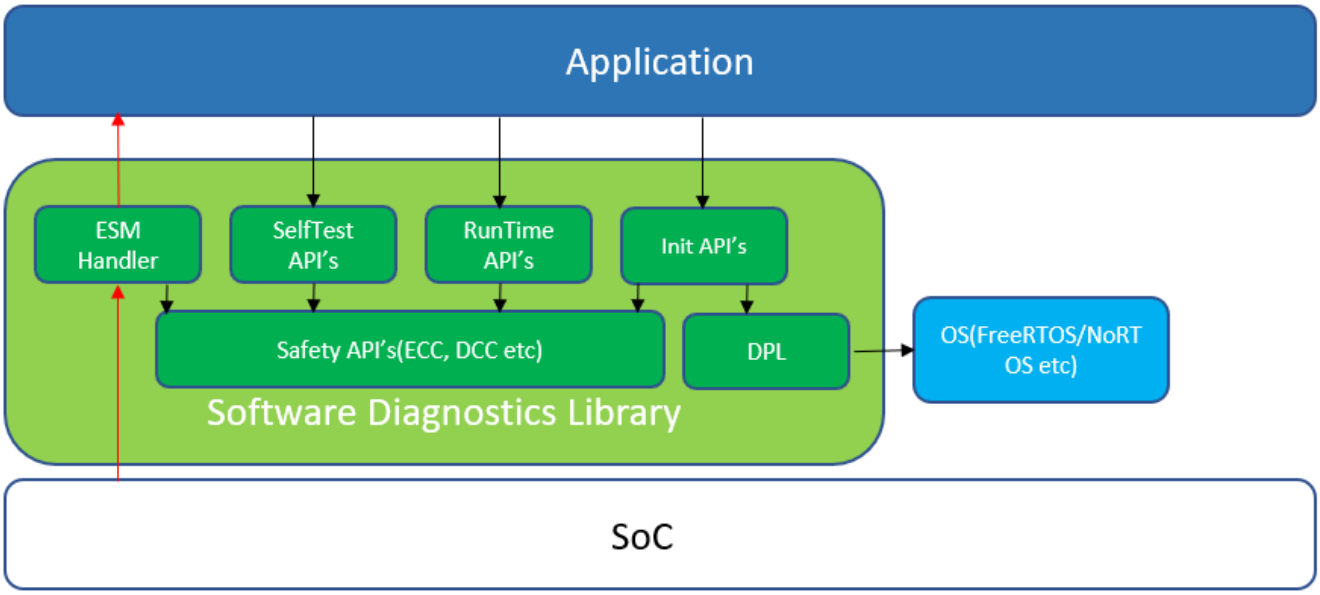

图 2-1. AWR294x SDL 架构

2.1 DPL 数据移植层

DPL（Data Porting Layer）是 SDL 的底层抽象层，屏蔽不同芯片平台和不同处理器核心在中断管理、延时控制和地址映射上的差异。所有上层 SDL 模块通过 DPL 接口访问底层硬件资源。

设计目的是解耦 SDL 模块与具体硬件实现，使得同一套 SDL 代码可以移植到 AM263x、AM273x、AWR2944 等不同芯片上。DPL 提供两组主要能力：中断管理（注册/注销/使能/禁用）和基础工具（延时/地址转换）。在 AWR2944 上同时支持 R5F 和 C66x DSP 两种核心的中断管理。

DPL 接口以函数指针结构体形式 SDL_DPL_Interface 注册到 SDL 库，包含接口如下：

表 2-1. DPL 接口函数

接口	对应函数	说明
enableInterrupt	SDL_TEST_enableInterrupt	使能指定中断号
disableInterrupt	SDL_TEST_disableInterrupt	禁用指定中断号
registerInterrupt	SDL_TEST_registerInterrupt	注册中断回调
deregisterInterrupt	SDL_TEST_deregisterInterrupt	注销中断回调

表 2-1. DPL 接口函数（续）

接口	对应函数	说明
delay	ClockP_sleep	延时（μs 级）
addrTranslate	SDL_TEST_addrTranslate	物理地址转本地地址

DPL 本身不提供独立测试，它作为所有 SDL 模块的前置条件。使用方法是在应用入口处先调 SDL_TEST_dplInit() 初始化 DPL，再初始化具体 SDL 模块。注意 DPL 必须在所有 SDL 模块之前初始化。

2.2 CCM 内核比较模块

CCM（Core Compare Module）实现 CPU 锁步（Lockstep）机制。AWR2944 中两个 R5F 核可配置为锁步模式，CCM 在每个时钟周期比对两个内核的输出（指令、地址、数据），当检测到不一致时判定 CPU 发生了硬件故障。CCM 支持两类监测：

- 输出比较块监测（SDL_CCM_MONITOR_TYPE_OUTPUT_COMPARE_BLOCK），比对主核和从核的指令级执行输出；
- VIM 比较监测（SDL_CCM_MONITOR_TYPE_VIM），Lockstep（锁步）模式让 CPU1 作为 CPU0 的诊断镜像，两者执行相同的中断处理。中断信号同时送入 CPU0 和经过延迟电路的 CPU1，各自的输出通过专用总线送到 CCMR5 模块比较。CPU0 的输出也被延迟以对齐时序。CCMR5 比对两路信号，若不一致则产生 MSS_CCMR5_ERR 错误中断，从而检测 CPU 核的硬件故障，错误通过 ESM 模块上报。

参考文献[1]中描述了该模块的底层机制：锁步比较器持续监视 CPU 的指令和数据总线，一旦发现偏差立即触发 ESM 安全响应。

CCM 功能测试（sdl_ccm_example）包含三个子项：

1. 正常自检：SDL_CCM_selfTest() 验证输出比较块基本功能。
2. 错误强制自检：以 SDL_CCM_SELFTEST_TYPE_ERROR_FORCING 模式调用 SDL_CCM_selfTest()，强制注入错误验证检测路径。
3. 错误注入：SDL_CCM_injectError() 注入比较错误，验证 ESM 中断响应和清除流程。

VIM 测试（sdl_ccm_vim）针对 VIM 路径比较错误，流程类似。

表 2-2. CCM 模块函数说明

关键函数	功能
SDL_CCM_init	初始化 CCM 模块
SDL_CCM_verifyConfig	验证 CCM 配置
SDL_CCM_selfTest	执行自检（正常/错误强制两种模式）
SDL_CCM_injectError	注入 CCM 错误
SDL_CCM_getErrorType	获取错误类型
SDL_CCM_clearError	清除 CCM 错误

2.3 DCC 双时钟比较器

DCC（Dual Clock Comparator）用一路已知参考时钟测量另一路时钟信号的频率，检测时钟漂移是否超出允许范围。参考文献[1]给出了详细工作原理：

DCC 内部包含两个递减计数器 counter0 和 counter1，分别由 clock0 和 clock1 驱动。应用程序为两个计数器设置种子值，并为 clock0 配置容差窗口计数器 valid0。启动后两个计数器同时递减。当 counter0 减到 0 时触发 valid0 开始递减。如果 counter1 在 valid0 递减期间也减到 0，则认为频率比正常，所有计数器重载种子值继续监测（连续模式）。

两种可能的错误条件：

1. counter1 在 counter0 之前减到 0 → clock1 过快或 clock0 过慢（含 clock0 卡死）。
2. counter1 在 counter0 和 valid0 都减到 0 后仍未到 0 → clock1 过慢或 clock1 卡死。

任一错误都会冻结计数器，CPU 可读取计数器值分析故障原因。

连续模式下 DCC 持续运行，无需软件干预；单次模式下完成一次测量后自动停止，产生完成中断。

AWR2944 中 MSS 含 4 个 DCC 实例，DSS 含 2 个：

表 2-3. MSS 和 DSS 的 DCC 实例

实例	子系统	参考时钟源	测试时钟源示例
MSS_DCCA	MSS	XTALCLK/RCCLK10	PLL_CORE / CR5F_CLK / SYS_CLK / QSPI / RTIA / SPIA / MCANA / SCIA
MSS_DCCB	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_CORE / CR5F_CLK / XTALCLK / CPTS / RTIB / SPIB
MSS_DCCC	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_DSS / GPIO[0-2] / WDT_CLK / XTALCLK
MSS_DCCD	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_PER / WDT_CLK / MCANA / GPIO[8-10,0-1]
DSS_DCCA	DSS	XTALCLK/RCCLK/SYS_CLK	SCIA / SYS_CLK / HWA_CLK / RTIA / WDT / DSP_CLK / PLL_DSP
DSS_DCCB	DSS	XTALCLK/RCCLK/RSS_CLK	HWA_CLK / CSI2_RX / WDT / RSS_CLK / PLL_PER

DCC 测试代码基本流程如下：

表 2-4. DCC 测试流程

步骤	操作	说明
1	SDL_TEST_dplInit	初始化 DPL
2	SDL_DCCA_clockInit	配置目标时钟源
3	SDL_ESM_init	注册 DCC 错误事件
4	SDL_DCCAppGetClkRatio SDL_DCCAppSetSeedVals	计算时钟比率和种子值，确定参考/测试时钟频率比和容差窗口
5	SDL_DCC_configure	配置 DCC 参数（种子值/容差/时钟选择）
6	SDL_DCC_verifyConfig	验证配置正确性
7	SDL_DCC_enableIntr	使能 DCC 完成/错误中断
8	SDL_DCC_enable	启动 DCC 监测
9	SDL_DCCAppWaitForCompletion	等待完成中断或错误中断

测试用例中的典型时钟组合：

表 2-5. DCC 测试用例典型时钟组合

用例	参考时钟	测试时钟	模式	预期
UC-1	XTAL_CLK(40MHz)	MSS_MCANA_CLK(30MHz)	连续	频率比偏差，产生错误
UC-2	XTAL_CLK(40MHz)	MSS_MCANA_CLK(30MHz)	单次	正常完成
UC-3	XTAL_CLK(40MHz)	SYS_CLK(150MHz)	连续	正常运行

DCC 模块提供的关键函数列表

表 2-6. DCC 模块关键函数说明

关键函数	功能
SDL_DCC_configure	配置时钟源和种子值/容差窗口
SDL_DCC_verifyConfig	验证配置正确性
SDL_DCC_enable	启动 DCC 监测

表 2-6. DCC 模块关键函数说明（续）

关键函数	功能
SDL_DCC_disable	停止 DCC
SDL_DCC_enableIntr	使能完成/错误中断
SDL_DCC_clearIntr	清除中断标志
SDL_DCC_getStatus	获取完成/错误状态

2.4 ECC 错误校正码

ECC 模块通过 ECC Aggregator（ECC 聚合器）和 ECC Wrapper（ECC 包装器）为片内存储器提供单错误校正（SEC）和双错误检测（DED）。参考文献[1]说明 ECC Aggregator 通过串行接口与各存储器的 ECC Wrapper 通信，Wrapper 负责实际的 ECC 检测与校正。AWR2944 芯片中包含 5 个 ECC Aggregator：

- MSS_ECC_AGG_R5A（R5F 核心 A 的 Cache/TCM/VIM）
- MSS_ECC_AGG_R5B（R5F 核心 B 的 Cache/TCM/VIM）
- MSS_ECC_AGG_MSS（L2/MBOX/RETRAM/GPADC/TPTC）
- DSS_ECC_AGG（L3/MBOX/TPTC）
- HSM_ECC_AGGR (HWA)

每个 Aggregator 管理多个 RAM ID，每个 ID 对应一个受 ECC 保护的存储器模块。

ECC Aggregator 的核心特性：

- 通过 ECC 减少存储器软错误（SEC 可校正，DED 可检测不可校正）
- 支持串行 FIFO 接口上的奇偶校验
- 控制位具有冗余保护，单比特翻转可触发中断
- 支持 Inject Only 模式，用于诊断测试
- 提供软件可读的 SEC/DED 错误状态，包含行地址和翻转比特位信息

ECC 测试覆盖 11 个存储器区域（r5_atcm0、r5_btcm、mss_l2、mss_tptc、mcan、dss_ecc_hwa、dss_l3_ecc、dss_mailbox、dss_tptc、dsp_edc、dss_l2_edc）。

通用测试流程：

1. R5F 核心执行 SDL_EXCEPTION_init 注册异常处理函数。
2. SDL_ECC_initMemory 初始化目标 RAM 的 ECC 保护。
3. 初始化 ESM 并配置高低优先级中断。
4. SDL_ECC_init 初始化 ECC 聚合器（指定 RAM ID 列表）。
5. SDL_ECC_injectError 注入单/双比特错误，等待 ESM 中断。
6. 在 ESM 回调中调用 SDL_ECC_getESMErrorInfo 和 SDL_ECC_getErrorInfo 获取错误详情，然后清除中断。

表 2-7. ECC 模块函数说明

关键函数	功能
SDL_ECC_init	初始化 ECC 聚合器，配置 RAM 子类型列表
SDL_ECC_initMemory	初始化指定 RAM 的 ECC 保护
SDL_ECC_injectError	向指定存储器注入 SEC/DED 错误
SDL_ECC_getESMErrorInfo	从 ESM 中断源解析 ECC 错误类型
SDL_ECC_getErrorInfo	获取详细 ECC 信息（地址、翻转比特等）
SDL_ECC_clearNIntrPending	清除中断挂起
SDL_ECC_ackIntr	应答 ECC 中断

小心

R5F 上需先初始化异常处理器（SDL_EXCEPTION_init），否则 DED 可能无法正确上报。

2.5 ECC 总线安全

ECC Bus Safety 将 ECC 保护从存储器扩展到总线互联模块。参考文献[1]指出，系统互联总线上的 command/address 总线具有奇偶校验和冗余保护，data 总线具有 SEC 和 DED 保护。该模块覆盖 MSS 和 DSS 两个子系统中的约 60 个总线节点。

测试针对每个总线节点分三类：单比特错误校正（SEC）、双比特错误检测（DED）、冗余错误检测（RED）。覆盖范围包括：TPTC 读写通道、DMA 控制器、Mailbox、FIFO、CPSW、QSPI、GPADC、MCRC 等模块。

测试方法：通过 EDMA 传输触发目标总线节点上的数据读写，注入 ECC 错误后验证 ESM 响应。

表 2-8. ECC 总线安全模块函数说明

关键函数	功能
SDL_ECC_BUS_SAFETY_*_SEC_Test	各节点的单比特错误校正测试
SDL_ECC_BUS_SAFETY_*_DED_Test	各节点的双比特错误检测测试
SDL_ECC_BUS_SAFETY_*_RED_Test	各节点的冗余错误检测测试

2.6 HWA 硬件加速器安全

HWA（Hardware Accelerator）是 AWR2944 中的雷达信号处理硬件加速器。 HWA 内部包含多个 DMEM（Data Memory）和一个 Window RAM，这些存储器受奇偶校验保护。HWA 安全模块检测 DMEM 的奇偶校验错误以及有限状态机（FSM）的锁步运行状态。

HWA 属于 DSS 子系统，ESM 实例为 SDL_ESM_INST_DSS_ESM。DSS ECC Aggregator 中 RAM ID 22 对应 HWA PARAM RAM。

测试覆盖 DMA0 和 DMA1 各 8 个 DMEM 以及 Window RAM，共 18 个奇偶校验测试项，加 1 个 FSM 锁步测试。每个测试调用 SDL_HWA_memParityExecute 或 SDL_HWA_fsmLockStepExecute。

表 2-9. HWA 安全模块函数说明

关键函数	功能
SDL_HWA_memParityExecute	执行指定存储器的奇偶校验测试
SDL_HWA_fsmLockStepExecute	执行 FSM 锁步测试
SDL_HWA_ESM_CallbackFunction	HWA ESM 错误回调

2.7 MCRC 循环冗余校验

MCRC（Memory Cyclic Redundancy Check）通过硬件加速的 CRC 计算验证存储器数据完整性。该模块使用最大长度 64 位多项式 $f(x) = x^{64} + x^4 + x^3 + x + 1$ 构建 PSA（Parallel Signature Analysis）寄存器，对 8/16/32/64 位数据宽度进行计算。

MCRC 支持两个独立通道，可并行对多个存储器区域进行 CRC 计算并支持三种模式：

自动模式：DMA 自动搬运数据做 CRC，无需 CPU 参与，匹配失败时产生中断，硬件提供 16 位扇区 ID 定位错误扇区。

半 CPU 模式：DMA 仍负责数据搬运，但每个扇区压缩完成后触发中断，由 CPU 读取签名寄存器并手动比对预置值。

全 CPU 模式：CPU 全程负责——既搬运数据到签名寄存器，又读取结果做比对。无中断、无 DMA，所有计数器关闭。

测试全 CPU 模式（mcrc_full_cpu）：

表 2-10. MCRC 全 CPU 模式流程

步骤	操作	说明
1	SDL_MCRC_init	初始化 MCRC 通道
2	SDL_MCRC_channelReset	复位 CRC 通道
3	SDL_MCRC_config	配置模式/模式大小/扇区数

表 2-10. MCRC 全 CPU 模式流程（续）

步骤	操作	说明
4	Input raw data	写入已知数据模式
5	SDL_MCRC_computeSignCPUmode	计算 CRC 签名
6	Compare signature	与参考签名比较

半 CPU 模式（mcrc_semi_cpu）：

表 2-11. MCRC 半 CPU 模式流程

步骤	操作	说明
1	SDL_MCRC_init	初始化 MCRC 通道
2	SDL_MCRC_channelReset	复位 CRC 通道
3	SDL_MCRC_config	配置通道为 Semi-CPU 模式
4	EDMA operation	将数据搬入 PSA 寄存器
5	Waiting for interrupt	MCRC 完成一个扇区压缩
6	读取 PSA Sector Signature	与参考签名比较

自动模式（mcrc_auto_cpu）：

表 2-12. MCRC 自动模式流程

步骤	操作	说明
1	SDL_MCRC_init SDL_MCRC_channelReset	初始化 并复位 MCRC 通道
2	SDL_MCRC_config	设置 SDL_MCRC_OPERATION_MODE_AUTO
3	EDMA transfer	同时搬运数据和预计算签名到 CRC Value Register
4	MCRC 自动比较	硬件完成签名比较
5	中断获取结果	CRC Fail 中断或完成通知

关键函数功能说明：

表 2-13. MCRC 模块函数说明

关键函数	功能
SDL_MCRC_init	初始化 MCRC 通道（看门狗/块预载值）
SDL_MCRC_channelReset	复位 CRC 通道
SDL_MCRC_config	配置模式、模式大小、扇区数
SDL_MCRC_configCRCType	配置 CRC 多项式
SDL_MCRC_computeSignCPUmode	全 CPU 模式计算 CRC 签名
SDL_MCRC_getPSASigRegAddr	获取 PSA 签名寄存器地址
SDL_MCRC_getPSASectorSig	获取扇区签名值
SDL_MCRC_getPSASig	读取 PSA 签名值
SDL_MCRC_enableIntr	使能 MCRC 中断

2.8 Parity 奇偶校验

奇偶校验检测存储器中的单比特翻转错误，仅能检测奇数个比特错误，不能校正。在 AWR2944 中主要用于 EDMA（MSS 与 DSS）的 TPCC 存储器，R5F TCM 存储器，DSS 的 L1 和 L2 存储器。错误通过 ESM 上报，软件需通过写控制寄存器对应 RAM 的奇偶校验来清除错误状态。

DMA TPCC Parity 测试（sdl_dma_parity）：

表 2-14. DMA TPCC Parity 校验流程

步骤	操作	说明
1	SDL_TEST_dplInit	初始化 DPL
2	SDL_ESM_init	注册 DMA 奇偶校验事件
3	Parity_funcTest	执行奇偶校验功能测试
4	ESM 回调处理	检测到错误后禁用对应 RAM 奇偶校验并清除状态

TCM Parity 测试和 DSS L1/L2 Parity 测试流程类似，分别针对 R5F TCM 和 DSS L1/L2 存储器。

表 2-15. Parity 模块函数说明

关键函数	功能
Parity_funcTest	执行奇偶校验功能测试
SDL_ESM_applicationCallbackFunction	ESM 回调函数处理奇偶校验错误

2.9 PBIST 可编程内置自测

PBIST（Programmable Built-In Self-Test）是一个专用协处理器，具有针对存储器测试的专用指令集。PBIST 从片上 ROM 读取测试算法和存储器配置信息，通过专用数据路径直接对目标存储器执行测试，无需 CPU 干预。

相比 CPU 运行软件测试，PBIST 的优势在于：专用路径绕过 CPU 的访问路径、专用指令集更高效、代码体积更小、硬件面积开销更低。AWR2944 包含两个 PBIST 实例：

- TOP_PBIST（覆盖 MSS 子系统全部存储器）
- DSS_DSP_PBIST（覆盖 DSS 子系统全部存储器）

推荐的测试算法为 March13N，可检测地址译码故障、Stuck-At 故障、耦合故障、状态耦合故障、参数故障、写恢复故障、读/写逻辑故障等。

PBIST 测试流程（以 TOP PBIST 为例）：

表 2-16. TOP PBIST 测试流程

步骤	操作	说明
1	SDL_TEST_dplInit	初始化 DPL
2	SDL_PBIST_configNeg	执行负测试，配置错误算法参数验证 PBIST 检测能力
3	禁用 Cache	禁用 L1P/L1D Cache，防止干扰测试结果
4	SDL_PBIST_selfTest	执行正常 PBIST，运行 March 算法检测存储器故障
5	等待完成/超时	等待 PBIST 完成中断或超时
6	获取结果	读取测试状态（通过/失败）

注意： PBIST 期间需禁用 Cache，测试耗时较长。

表 2-17. PBIST 模块函数说明

关键函数	功能
SDL_PBIST_selfTest	执行 PBIST 自测
SDL_PBIST_configNeg	PBIST 负测试配置（故障注入）

2.10 Reset 复位管理

Reset 模块提供复位原因查询和软件复位生成。通过读取复位状态寄存器，软件可判断上次复位原因（上电复位/看门狗复位/软件复位/外部引脚复位/HSM 看门狗复位），从而在启动时选择不同初始化策略。

Reset 测试流程：

表 2-18. RESET 模块测试流程

步骤	操作	说明
1	SDL_getWarmResetCause	读取暖复位原因
2	分类处理	根据复位原因选择初始化策略
3	SDL_r5fGetResetCause	读取 R5F 核心复位原因
4	SDL_generateSwWarmReset	若为上电复位，触发软件暖复位验证路径
5	SDL_rcmDspLocalReset	执行 DSP 本地复位

表 2-19. RESET 模块函数说明

关键函数	功能
SDL_getWarmResetCause	获取暖复位原因
SDL_r5fGetResetCause	获取 R5F 核心复位原因
SDL_generateSwWarmReset	触发软件暖复位
SDL_rcmDspLocalReset	触发 DSP 本地复位

备注

复位状态寄存器读取后自动清除；软件热复位前需保存关键数据。

2.11 ROM 校验和

ROM Checksum 对 MCU ROM 内容计算校验和并与预置参考值比较，检测 ROM 内容的意外修改或损坏。这是系统启动时建立信任的手段之一。

ROM Checksum 测试流程：

表 2-20. ROM 内容校验流程

步骤	操作	说明
1	SDL_TEST_dpllnit	初始化 DPL
2	SDL_ROM_Checksum_compute	计算 ROM 校验和
3	比较结果	与预置参考值比较

返回值有三种情况：SDL_PASS（通过）、SDL_EBADARGS（参数错误）、其他（完整性校验失败）。

表 2-21. ROM 模块函数说明

关键函数	功能
SDL_ROM_Checksum_compute	计算 ROM 校验和并验证完整性

2.12 RTI 实时窗口看门狗

RTI DWWD（Digital Windowed Watchdog，数字窗口看门狗）要求喂狗必须在特定时间窗口内完成，过早或过晚都会触发安全响应。有效检测 CPU 执行流异常（跑飞、死锁、执行延迟）。

窗口大小比例配置：

表 2-22. RTI 窗口比例配置

窗口大小	打开窗口比例	关闭窗口比例
100%	100%	0%（传统看门狗行为）
50%	50%	50%
25%	25%	75%
12.5%	12.5%	87.5%
6.25%	6.25%	93.75%
3.125%	3.125%	96.875%

测试流程覆盖四种使用场景（UC1-UC4）：

表 2-23. RTI 模块测试流程

步骤	操作	说明
1	配置 RTI 参数	设置窗口大小和预载值
2	启动看门狗	调用 SDL_RTI_start
3	UC1：正确服务	在打开窗口内喂狗，无错误中断
4	UC2：永久故障	不喂狗触发超时，ESM 中断触发
5	UC3：过早服务	在关闭窗口内喂狗，ESM 中断触发
6	UC4：窗口切换	配置不同窗口比例，验证服务时机合法性

表 2-24. RTI 模块函数说明

关键函数	功能
SDL_RTI_config	配置窗口看门狗参数（窗口大小/预载值）
SDL_RTI_verifyConfig	验证 RTI 配置
SDL_RTI_start	启动窗口看门狗
SDL_RTI_service	服务（喂狗）
SDL_RTI_writeWinSz	动态修改窗口大小
SDL_RTI_getStatus	获取中断状态
SDL_RTI_clearStatus	清除中断状态

2.13 STC 自测控制器

STC（Self-Test Controller）基于 OPMISR（On-Product Multiple Input Signature Register）对 CPU 内核逻辑门电路进行 逻辑自测（Logic Built-In Self-Test）。STC 主要包含 ROM 接口、FSM 序列控制器、寄存器文件、Bypass/ATE 接口和 VBUSP 接口。

STC 从片上 ROM 读取测试码（含 Segment ID、Pattern Count、Golden MISR 值），将扫描数据移入被测逻辑，最后将 OPMISR 控制器产生的 896 位 MISR 值与 ROM 中的 Golden MISR 值比较，结果写入状态寄存器。支持 Stuck-at 故障模型和 Launch-on-System-Clock 转换延迟方法。

AWR2944 中 MSS 和 DSS 的 STC 实例独立运行，可分别触发。

STC 模块测试流程：

表 2-25. STC 模块测试流程

步骤	操作	说明
1	SDL_TEST_dplInit	初始化 DPL
2	SDL_ESM_init	注册 STC 事件
3	SDL_STC_config	配置 STC 参数（测试时长/间隔/算法）
4	SDL_STC_selfTest	执行 STC 自测
5	等待完成/超时	等待测试完成或超时
6	SDL_STC_getResult	读取测试结果
7	清除中断	清除 STC 中断标志

表 2-26. STC 模块函数说明

关键函数	功能
SDL_STC_selfTest	执行 STC LBIST 自测
SDL_STC_config	配置 STC 参数
SDL_STC_getResult	获取 STC 测试结果

备注

STC 期间 CPU 暂停执行用户代码，需在诊断覆盖率和系统可用性之间平衡。

3 总结

SDL 为 AWR294x 系列提供了一个即插即用、符合功能安全标准的诊断解决方案，，通过标准化 API 降低了安全机制集成的复杂度。其模块化设计使系统集成商能够快速构建功能安全应用，显著缩短开发周期。SDL 的分层架构实现了硬件诊断与应用逻辑的解耦，提高了代码可维护性和可移植性。作为 TI 官方提供的诊断库，SDL 已在汽车雷达等安全关键领域得到广泛验证，为开发者提供了可靠的功能安全诊断实现路径。

4 名词表

RBL - ROM Bootloader -	ROM 引导加载程序
SBL - Secondary Bootloader -	二级引导加载程序
ROM - Read Only Memory -	只读存储器
HWA - Hardware Accelerator -	硬件加速器
MSS - Main SubSystem (R5F) -	主子系统（R5F）
DSS - DSP SubSystem -	DSP 子系统
RSS - Radar SubSystem -	雷达子系统
BSS - BIST SubSystem -	内建自测试子系统
DSP - Digital Signal Processors -	数字信号处理器
DFE - Digital Front End -	数字前端
MPU - Memory Protection Unit -	内存保护单元
EDMA - Enhanced Direct Memory Access -	增强型直接内存访问
RTI - Real Time Interrupt -	实时中断
PBIST - Programmable Built-In Self-Test -	可编程内建自测试
ESM - Error State Machine -	错误状态机
MCRC - Memory Cyclic Redundancy Check -	存储器循环冗余校验
DCC - Dual Clock Comparator -	双时钟比较器
GPADC - General-Purpose Analog Digital Converter -	通用模数转换器
TCMA/TCMB - Tightly Coupled Memory -	紧耦合存储器
R5FA/R5FB - ARM Real Time Processor with Floating Point Unit -	ARM 实时处理器（带浮点单元）
TPTC - Third Party Transfer Controller -	传输控制器
TPCC - Third Party Channel Controller -	通道控制器
PRCM - Power, Reset and Clock Management -	电源、复位与时钟管理
STC - Self-Test Controller -	自测试控制器
WDT - Watchdog Timer -	看门狗定时器
WWDT - Windowed Watchdog Timer -	窗口看门狗定时器
VIM - Vectored Interrupt Manager -	向量中断管理器
IRQ - Interrupt Request -	中断请求
FIQ - Fast Interrupt Request -	快速中断请求
SECDEC - Single Error Correction Double Error Code -	单比特纠错双比特检错码
ECC - Error Correction Code -	错误纠正码

5 参考文献

- *AWR294x Technical Reference Manual (Rev. D)*
- *MMWAVE-MCUPLUS-SDK Software development kit (SDK)*
- *AWR2943/AWR2944 Single-Chip 76 to 81GHz FMCW Radar Sensor datasheet (Rev. E)*

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月