AN13138

在 Layerscape 平台上部署 Xen

第 0 版 — 2021 年 3 月

应用笔记

目录

1 引言

本文档旨在为硬件、软件和系统工程师提供必要的信息，以在恩智浦 Layerscape 平台上运行 Xen 管理程序。在此操作中，我们考虑基本场景，如基本域管理、dom0less 和设备透传。

2 操作

Xen 项目管理程序可以直接在硬件上或在 QEMU 下运行。在本应用笔记中，我们只考虑第一种情况。以下说明为您提供了镜像的位置以及启动管理程序和一个或多个域所需的步骤。

下面的构建说明已经在运行 Ubuntu 18.04 的 x86_64 机器上测试，使用的是具有 sudo 权限的非 root 用户。

2.1 构建

我们使用 Yocto 项目来构建 Linux 内核和 Xen 镜像，包括管理程序二进制文件和 Xen 项目工具栈和文件系统。

2.1.1 要求

确保构建机器满足以下要求：

- 50 GB 或更大的空闲磁盘空间
- Git 1.8.3.1 或更高版本
- tar 1.27 或更高版本
- Python 3.4.0 版或更高版本

2.1.2 依赖关系

在构建主机上安装以下软件包：

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath
socat cpio python python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping libsqlite3-dev
```

注意

软件包可能因构建系统而异。

2.1.3 下载并安装 Yocto 项目

1. 安装 repo 工具：

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

```
$ chmod a+x ~/bin/repo
$ PATH=${PATH}:~/bin
```

2. 下载 Yocto 元数据：

```
$ mkdir yocto-sdk
$ cd yocto-sdk
$ repo init -u https://source.codeaurora.org/external/qoriq/qoriq-components/yocto-sdk -b
<branch> #zeus and dunfell branches have been tested
$ repo sync --force-sync
```

3. 使用 poky 脚本为目标平台设置环境变量：

```
$ . ./setup-env -m <target_board>
```

4. 向 `conf/local.conf` 文件添加以下命令行：

```
IMAGE_FSTYPES_append = " ext2"
DISTRO_FEATURES_append = " xen"
IMAGE_INSTALL_append = " zlib-dev xen-base"  #zeus branch
IMAGE_INSTALL_append = " zlib-dev xen-tools" #dunfell branch
```

5. 确保启用了以下内核选项：

```
CONFIG_XEN_DOM0=y
CONFIG_XEN=y
CONFIG_XEN_BLKDEV_FRONTEND=y
CONFIG_XEN_NETDEV_FRONTEND=y
CONFIG_INPUT_XEN_KBDDEV_FRONTEND=y
CONFIG_HVC_XEN=y
CONFIG_HVC_XEN_FRONTEND=y
CONFIG_XEN_FBDEV_FRONTEND=y
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES_DEFAULT=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XEN_BACKEND=y
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_SYS_HYPERVISOR=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_XEN_GNTDEV=y
CONFIG_XEN_GRANT_DEV_ALLOC=y
CONFIG_SWIOTLB_XEN=y
CONFIG_XEN_PRIVCMD=y
CONFIG_XEN_EFI=y
CONFIG_XEN_AUTO_XLATE=y
```

6. 如果需要启用某些内核选项，请使用以下命令：

```
$ bitbake -c menuconfig virtual/kernel
```

7. 启动构建：

```
$ bitbake fsl-image-networking
```

8. 在以下目录中查找生成的镜像：

/<workdir>/yocto-sdk/build_<target_board>/tmp/deploy/images/<target_board>

```
fsl-image-networking-<target_board>.rootfs.cpio.gz        #used as dom0-rfs and domu-rfs
fsl-image-networking-<target_board>.rootfs.ext2           #used as domu-rfs in section 2.2.2
fsl-<target_board>.dtb                                    #device tree blob used for Dom0
Image                                                     #used as dom0-image and domu-image
xen-<target_board>
```

9. Image 表示用于 Dom0 的内核镜像。它也可以用于非权限域。如果需要不同的内核配置，请运行以下命令以使用所需选项重建内核镜像：

```
$ bitbake -c menuconfig virtual/kernel
$ bitbake -c compile -f virtual/kernel
```

注意

如果使用较新版本的 GCC（例如 GCC 9），您将遇到 address-of-packed-member 类型的警告，这将导致构建失败。使用 https://patchwork.kernel.org/patch/11265041/ 修补程序修复。Xen 源位于 /<workdir>/yocto-sdk/build_<target_board>/tmp/work/aarch64-fsl-linux/xen/<version>目录中。

2.2 部署

在执行以下步骤之前，请确保在构建了镜像的机器上启动并运行 TFTP 服务器。

2.2.1 运行 Xen 和 Dom0

选择合适的闪存库，在 U-Boot 中运行以下指令。如果不熟悉 Layerscape 平台，请参见 https://lsdk.github.io/document.html 以确定合适的库。必须选择以下指令中使用的地址，使其符合平台的内存布局。

1. 将 Xen 镜像加载到$xen_addr 地址：

```
u-boot> tftp $xen_addr $path/xen-<target_board>
```

2. 将 Dom0 内核镜像加载到$kernel_addr 地址：

```
u-boot> tftp $kernel_addr $path/Image
```

3. 将 Dom0 设备树 blob 加载到$fdt_addr 地址：

```
u-boot> tftp $fdt_addr $path/fsl-<target_board>.dtb
```

在本文档中，我们提供了如何为 Dom0 使用 RAM 磁盘或使用存储在大容量存储设备（例如 SD 卡）上的文件系统的说明。但是，可以使用其他设备进行存储。

4. 如果选择使用 RAM 磁盘，请运行以下命令，将其加载到$rootfs_addr 地址：

```
u-boot> tftp $rootfs_addr $path/dom0-rfs
```

您也可以将 rootfs 写入 SD 卡。

5. 添加包含 dom0 内核的启动模块：

```
u-boot> fdt addr $fdt_addr
u-boot> fdt resize 1024
u-boot> fdt set /chosen \#address-cells <0x2>
u-boot> fdt set /chosen \#size-cells <0x2>
u-boot> fdt set /chosen xen,xen-bootargs "console=dtuart dtuart=serial0 dom0_mem=2G
dom0_max_vcpus=1 bootscrub=0 vwfi=native sched=null earlycon=xen loglvl=all guest_loglvl=all"
```

```
#Xen command line
u-boot> fdt mknod /chosen dom0
u-boot> fdt set /chosen/dom0 compatible "xen,linux-zimage" "xen,multiboot-module"
u-boot> fdt set /chosen/dom0 reg <0x0 <$kernel_addr> 0x0 <$kernel_size>>
```

6. 如果需要，添加包含 dom0 ramdisk 的启动模块：

```
u-boot> fdt set /chosen xen,dom0-bootargs "console=hvc0 earlycon=xen earlyprintk=xen
clk_ignore_unused root=/dev/ram0" #Dom0 command line
u-boot> fdt mknod /chosen dom0-ramdisk
u-boot> fdt set /chosen/dom0-ramdisk compatible "xen,linux-initrd" "xen,multiboot-module"
u-boot> fdt set /chosen/dom0-ramdisk reg <0x0 <$rootfs_addr> 0x0 <$rootfs_size>>
```

7. 如果文件系统存储在 SD 卡上，请使用以下 dom0 命令行：

```
u-boot> fdt set /chosen xen,dom0-bootargs "console=hvc0 earlycon=xen earlyprintk=xen
clk_ignore_unused root=/dev/mmcblk0p1 rw rootwait"
```

注意

此信息可以直接添加到设备树源文件中。

8. 启动：

```
u-boot> booti $xen_addr - $fdt_addr
```

9. 打印 Xen 主机的信息：

```
# xl info
host                   : ls1046ardb
release                : 4.19.46-g4ddb5d78dd74-dirty
version                : #8 SMP PREEMPT Thu Jul 9 11:18:53 EEST 2020
machine                : aarch64
nr_cpus                : 4
max_cpu_id             : 3
nr_nodes               : 1
cores_per_socket       : 1
threads_per_core       : 1
cpu_mhz                : 25.000
hw_caps                : 00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000
virt_caps              :
total_memory           : 8126
free_memory            : 186
sharing_freed_memory   : 0
sharing_used_memory    : 0
outstanding_claims     : 0
free_cpus              : 0
xen_major              : 4
xen_minor              : 12
xen_extra              : .0
xen_version            : 4.12.0
xen_caps               : xen-3.0-aarch64 xen-3.0-armv7l
xen_scheduler          : credit2
xen_pagesize           : 4096
platform_params        : virt_start=0x200000
xen_changeset          :
xen_commandline        : console=dtuart dtuart=serial0 dom0_mem=7500M loglvl=all
guest_loglvl=all sync_console
cc_compiler            : aarch64-fsl-linux-gcc (GCC) 9.2.0
cc_compile_by          : nxf41040
```

```
cc_compile_domain      :
cc_compile_date        : Thu Jul  9 08:49:59 UTC 2020
build_id               : 6f101bbbf7a8f3f35404b08fe936718a0af2adda
xend_config_format     : 4
```

10. 打印有关现有域的信息：

```
# xl list
Name                                ID   Mem VCPUs State Time(s)
Domain-0                             0  2048     1    r------- 28.7
```

2.2.2 运行 DomU

Dom0 启动后，您可以使用 Xen 管理工具栈 `xl` 创建和访问非权限域。

1. 挂载客机文件系统：

```
# losetup /dev/loop0 $path/domu-rfs
```

其中：`domu-rfs` 是 `fsl-image-networking-<target_board>.rootfs.ext2`

2. 创建配置文件 `domu.cfg` 并添加以下命令行：

```
name = "domu"
vcpus = 1
memory = 1024
kernel = "$path/domu-image"
disk = [ 'phy:/dev/loop0,xvda,w' ]
extra = "root=/dev/xvda rw earlyprintk=xenboot console=hvc0"
```

如需了解更多信息，请参见 https://xenbits.xen.org/docs/unstable/man/xl.cfg.5.html。

注意

您可以对 Dom0 和 DomU 使用相同的内核，但这不是强制性的。如果您希望对非权限域使用另一种内核配置，请按照下载并安装 Yocto 项目一节末尾列出的指令来构建不同的内核镜像。

3. 创建域：

```
# xl create domu.cfg
```

4. 验证域是否已创建：

```
$: xl list
Name                                ID   Mem VCPUs  State     Time(s)
Domain-0                             0  4096     4  r-----      28.7
domu                                 1  1024     1  r-----       2.7
```

5. 启动域：

```
# xl console domu
```

6. 以 `root` 用户身份登录。

以下命令类似于运行 `xl create domu.cfg && xl console domu`：

```
# xl create -c domu.cfg
```

7. 使用<CTRL>']'从域控制台分离。请注意，telnet 也使用此转义字符。您可以使用-E 选项连接到电路板控制台，以阻止任何字符被用作转义字符：

```
# telnet -E <ip> <port>
```

8. 销毁域：

```
# xl destroy domu
```

注意

非权限域无法访问系统硬件或设备驱动程序。如果需要，可以将硬件透传到这些域，或者使用可用的半虚拟化设备驱动程序。

2.2.3 Dom0less

有一种部署 Xen 域的替代解决方案，即 dom0less。它不依赖于控制域，而是允许用户在启动时通过设备树模块向 Xen 传递所需信息来创建域。

1. 加载 Xen 和 Dom0 镜像后，加载 DomU 内核和文件系统：

```
u-boot> tftp $domu_kernel_addr $path/domu-image
u-boot> tftp $domu_rootfs_addr $path/domu-rfs
```

注意

这些指令假设您正在使用下载和安装 Yocto 项目一节中描述的命令来获取非权限域的内核镜像和文件系统。

2. 编辑设备树 blob：

```
u-boot> fdt set /chosen/domU0 compatible "xen,domain"
u-boot> fdt set /chosen/domU0 \#address-cells <0x2>
u-boot> fdt set /chosen/domU0 \#size-cells <0x2>
u-boot> fdt set /chosen/domU0 memory <0x0 0x100000>
u-boot> fdt mknod /chosen domU0
u-boot> fdt set /chosen/domU0 cpus <0x1>
u-boot> fdt set /chosen/domU0 vpl011 <0x1>
u-boot> fdt mknod /chosen/domU0 module$domu_kernel_addr
u-boot> fdt set /chosen/domU0/module$domu_kernel_addr compatible "multiboot,kernel"
"multiboot,module"
u-boot> fdt set /chosen/domU0/module$domu_kernel_addr reg <0x0 $domu_kernel_addr 0x0
$domu_kernel_size>
u-boot> fdt set /chosen/domU0/module$domu_kernel_addr bootargs "console=ttyAMA0"
u-boot> fdt mknod /chosen/domU0 module$domu_rootfs_addr
u-boot> fdt set /chosen/domU0/module$domu_rootfs_addr compatible "multiboot,ramdisk"
"multiboot,module"
u-boot> fdt set /chosen/domU0/module$domu_roootfs_addr reg <0x0 $domu_rootfs_addr 0x0
$domu_rootfs_size>
```

注意

此信息可以直接添加到设备树源文件中。

有关详细信息，请参见 https://xenbits.xenproject.org/docs/unstable/features/dom0less.html。

2.2.4 ImageBuilder

您还可以使用 ImageBuilder（一组用于构建自动化的脚本）在 Layerscape 平台上部署 Xen。

以下配置文件已在 LS2088ARDB 上测试：

```
MEMORY_START="0x80000000"
MEMORY_END="0xFBE00000"

DEVICE_TREE="$path/fsl-ls2088a-rdb.dtb"
XEN="$path/xen-ls2088ardb"
DOM0_KERNEL="$path/dom0-img"
DOM0_RAMDISK="$path/dom0-rfs"

NUM_DOMUS=1
DOMU_KERNEL[0]="$path/domu-img"
DOMU_RAMDISK[0]="$path/domu-rfs"

UBOOT_SOURCE="boot.source"
UBOOT_SCRIPT="boot.scr"
```

使用 `bdinfo` U-Boot 命令获取内存起始地址和大小。

1. 调用 uboot-script-gen：

```
$ bash ./scripts/uboot-script-gen -c config -d . – tftp
Generated uboot script boot.scr, to be loaded at address 0x80200000:
tftpb 0x80200000 boot.scr; source 0x80200000
```

注意

`$path` 相对于作为参数传递给`-d` 选项的目录。

2. 使用 `ImageBuilder` 打印的指令在 U-Boot 中加载脚本并将其作为源代码。

   `ImageBuilder` 中有一些默认选项，例如 `dom0-bootargs`。如果需要，请使用以下指令执行更改：

   a. 编辑 `boot.source`

   b. 重新生成 `boot.scr`

```
$ mkimage -A arm64 -T script -C none -a 0x80200000 -d boot.source boot.scr
```

2.2.2 启用 SMMU 支持

在 Xen `arm-smmu` 驱动程序中，页表遍历的起始级别被硬编码为 1，这在级别具有不同值的情况下会导致错误，就像 Layerscape 平台的情况一样。

如果要在启用了 SMMU 的 Layerscape 平台上运行 Xen 管理程序：

1. 打上上游发送的以下补丁：

```
From xen-devel  Fri Oct 02 10:33:44 2020
From: laurentiu.tudor () nxp ! com
Date: Fri, 02 Oct 2020 10:33:44 +0000
To: xen-devel
Subject: [PATCH v3] arm,smmu: match start level of page table walk with P2M
Message-Id: <20201002103344.13015-1-laurentiu.tudor () nxp ! com>
X-MARC-Message: https://marc.info/?l=xen-devel&m=160163484031524

From: Laurentiu Tudor <laurentiu.tudor@nxp.com>

Don't hardcode the lookup start level of the page table walk to 1
and instead match the one used in P2M. This should fix scenarios
```

```
involving SMMU where the start level is different than 1.
In order for the SMMU driver to also compile on arm32 move the
P2M_ROOT_LEVEL in the p2m header file (while at it, for
consistency also P2M_ROOT_ORDER) and use the macro in the smmu
driver.

Signed-off-by: Laurentiu Tudor <laurentiu.tudor@nxp.com>
---
Changes in v3:
 - also export 'p2m_root_order'
 - moved variables in their rightful #ifdef block

Changes in v2:
 - made smmu driver compile on arm32

 xen/arch/arm/p2m.c                  |  9 ++-------
 xen/drivers/passthrough/arm/smmu.c |  2 +-
 xen/include/asm-arm/p2m.h          | 11 ++++++++++
 3 files changed, 14 insertions(+), 8 deletions(-)

diff --git a/xen/arch/arm/p2m.c b/xen/arch/arm/p2m.c
index ce59f2b503..4eeb867ca1 100644
--- a/xen/arch/arm/p2m.c
+++ b/xen/arch/arm/p2m.c
@@ -17,17 +17,12 @@
 #define INVALID_VMID 0 /* VMID 0 is reserved */

 #ifdef CONFIG_ARM_64
-static unsigned int   read_mostly p2m_root_order;
-static unsigned int   read_mostly p2m_root_level;
-#define P2M_ROOT_ORDER    p2m_root_order
-#define P2M_ROOT_LEVEL p2m_root_level
+unsigned int   read_mostly p2m_root_order;
+unsigned int   read_mostly p2m_root_level;
 static unsigned int   read_mostly max_vmid = MAX_VMID_8_BIT;
 /* VMID is by default 8 bit width on AArch64 */
 #define MAX_VMID       max_vmid
 #else
-/* First level P2M is always 2 consecutive pages */
-#define P2M_ROOT_LEVEL 1
-#define P2M_ROOT_ORDER    1
 /* VMID is always 8 bit width on AArch32 */
 #define MAX_VMID        MAX_VMID_8_BIT
 #endif
diff --git a/xen/drivers/passthrough/arm/smmu.c b/xen/drivers/passthrough/arm/smmu.c
index 94662a8501..4ba6d3ab94 100644
--- a/xen/drivers/passthrough/arm/smmu.c
+++ b/xen/drivers/passthrough/arm/smmu.c
@@ -1152,7 +1152,7 @@ static void arm_smmu_init_context_bank(struct arm_smmu_domain *smmu_domain)
           (TTBCR_RGN_WBWA << TTBCR_IRGN0_SHIFT);

     if (!stage1)
-        reg |= (TTBCR_SL0_LVL_1 << TTBCR_SL0_SHIFT);
+        reg |= (2 - P2M_ROOT_LEVEL) << TTBCR_SL0_SHIFT;

     writel_relaxed(reg, cb_base + ARM_SMMU_CB_TTBCR);

diff --git a/xen/include/asm-arm/p2m.h b/xen/include/asm-arm/p2m.h
index 5fdb6e8183..28ca9a838e 100644
--- a/xen/include/asm-arm/p2m.h
```

```
+++ b/xen/include/asm-arm/p2m.h
@@ -13,6 +13,17 @@
 /* Holds the bit size of IPAs in p2m tables.  */
 extern unsigned int p2m_ipa_bits;

+#ifdef CONFIG_ARM_64
+extern unsigned int p2m_root_order;
+extern unsigned int p2m_root_level;
+#define P2M_ROOT_ORDER    p2m_root_order
+#define P2M_ROOT_LEVEL p2m_root_level
+#else
+/* First level P2M is always 2 consecutive pages */
+#define P2M_ROOT_ORDER    1
+#define P2M_ROOT_LEVEL 1
+#endif
+
 struct domain;

 extern void memory_type_changed(struct domain *);
--
2.17.1
```

注意

有关两级表访问的更多详细信息，请参见 *https://developer.arm.com/ documentation/100940/0101*。

Xen `arm-smmu` 驱动程序使用传统设备树绑定，因此需要对 Dom0 使用的设备树进行额外更改。例如，`mmu-masters` 属性已被弃用，因此不包含在本机设备树源码中。

2. 将 `mmu-masters` 属性添加到 `smmu` 节点，并为每个主机指定 `StreamID`：

```
smmu: iommu@9000000 {
    /* Other properties */
    mmu-masters = <&eshdc 0xc04>, <&sata 0xc05> …
};
```

3. 向 `mmu-masters` 中列出的每个节点添加`#stream-id-cells`：

```
esdhc: esdhc@1560000 {
    /* Other properties */
    #stream-id-cells = <1>;
};

sata: sata@3200000 {
    /* Other properties */
    #stream-id-cells = <1>;
};
```

如果配置错误，将触发全局故障，外围设备将无法工作。例如，在以下示例中 `StreamID` 0x420 被 SMMU 接收，但无法被识别。

```
(XEN) smmu: /soc/iommu@5000000: Unexpected global fault, this could be serious
(XEN) smmu: /soc/iommu@5000000:  GFSR 0x80000002, GFSYNR0 0x00000000, GFSYNR1 0x00000420, GFSYNR2
0x00000000
```

2.2.3 设备透传

您可以通过设备透传将资源分配给没有权限的访客。以下说明提供了如何透传 USB 设备的指导。可以执行相同的步骤来透传 PCIe 控制器。有关详细信息，请参见 https://xenbits.xen.org/docs/4.9-testing/misc/arm/passthrough.txt。

USB 透传

1. 为 DomU 创建设备树 blob：

```
/dts-v1/;

/ {
    #address-cells = <2>;
    #size-cells = <2>;

    passthrough {
        compatible = "simple-bus";
        ranges;
        #address-cells = <2>;
        #size-cells = <2>;
        usb0: usb3@3100000 {
                  status = "okay";
                  #stream-id-cells = <0x1>;
                  compatible = "snps,dwc3";
                  reg = <0x0 0x3100000 0x0 0x10000>;
                  interrupts = <0 80 0x4>;
                  dr_mode = "host";
                  snps,quirk-frame-length-adjustment = <0x20>;
                  snps,dis_rxdet_inp3_quirk;
                  snps,incr-burst-type-adjustment = <1>, <4>, <8>, <16>;
                  snps,host-vbus-glitches;
              };
    };
};
```

2. 在 U-Boot 中运行以下命令以标记设备，以便 Xen 知道它将用于透传：

```
u-boot> fdt set /soc/usb3@3100000 "xen,passthrough"
```

3. 将以下命令行添加到 DomU `domu.cfg` 配置文件：

```
device_tree = "$path/domu.dtb"
irqs = [ 112 ]
iomem = [ "0x03100,0x10" ]
```

PCIe 控制器透传（LS1046ARDB）

此场景与前一个场景类似，但也需要 MSI 控制器的透传，因为它们是由控制器使用的。

> **注意**
>
> 在这个平台上，MSI 控制器是恩智浦自定义的，Xen 将其视为常规设备，默认分配给 Dom0。Dom0 接收常规中断，设备驱动程序将其转换为 MSI。因此，即使 Xen 中缺少 MSI 支持，也可以将 PCIe 与 MSI 一起使用。

3 已知限制

不支持 DPAA2 功能，因为 Xen 中缺少对 MC 总线和驱动程序的支持。

不支持需要时钟源的设备透传。

Xen 中缺少 MSI 支持。

4 故障排除

4.1 共享中断

在 Layerscape 平台上，串行设备共享相同的中断，如以下设备树片段所示：

```
duart0: serial@21c0500 {
            compatible = "fsl,ns16550", "ns16550a";
            reg = <0x00 0x21c0500 0x0 0x100>;
            interrupts = <GIC_SPI 54 IRQ_TYPE_LEVEL_HIGH>;
            clocks = <&clockgen 4 1>;
};

duart1: serial@21c0600 {
            compatible = "fsl,ns16550", "ns16550a";
            reg = <0x00 0x21c0600 0x0 0x100>;
            interrupts = <GIC_SPI 54 IRQ_TYPE_LEVEL_HIGH>;
            clocks = <&clockgen 4 1>;
};
```

Xen 不允许在它和其他域之间共享中断。在运行 Xen 和 Dom0 的基本场景中，一个 UART 由管理程序使用，第二个 UART 在创建时分配给域。一种解决方法是从 Dom0 设备树中删除 `duart1` 节点。

4.2 SPI 分配

因为 Xen 只为其域分配 SPI，必须从 dts 中删除使用其他类型中断的设备（例如，在 LS1028ARDB 上，PMU 使用 PPI 类型中断）。否则，域 0 无法正确设置并触发 panic 事件。

4.3 LS1028ARDB 启动问题

在 LS1028ARDB 平台上，GPU 节点保留了整个低内存库。这会在 Xen 场景中触发 panic 事件。一种可行的解决方法是从 dts 中删除 `gpu` 节点，或者保留第二个内存库而不是第一个。

4.4 重复的流 ID

在当前 SMMU 驱动程序实现中，不支持多个主设备共享相同流 ID 的场景。为了解决这个问题，只需在 `mmu-masters` 中列出一个主设备。

5 参考资料

- https://wiki.xenproject.org
- https://xenbits.xen.org/docs/

6 修订历史

表 1 总结了自首次发布以来对本文档所做的更改。

表 1. 修订历史

版本号	日期	实质性变更
0	2021 年 3 月	初版发布

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 03/2021

Document identifier: AN13138

arm