LSDK INTRODUCTION

XIE XIAOBO
SOFTWARE MANAGER

AUGUST 2017
AGENDA

• What is LSDK
• Why change to LSDK
• LSDK in detail
• How to use LSDK
• Plan & Roadmap
• Summary
WHAT IS LSDK
Overview - Keywords

LSDK is NXP new generation of SDK for Layerscape productions

• **Disaggregation**
  The concept that many software components are available individually. This enables customers and 3rd parties to access them individually so they can integrate them into Linux distributions or systems by themselves.

• **LTS**
  Long Term Support, used to describe a kernel or Linux distribution that will be formally supported with prompt bug fixes, security updates, and limited feature additions for a defined time period.

• **Linux Distribution**
  A complete Linux kit from a specific provider. Includes kernel, tools, user space, etc.

• **Upstreaming**
  The process of adding support for NXP-specific hardware or features to a community (non-NXP) software repository.

You can get the source code and related documents from:

https://lsdk.github.io/
Overview - Evolution

Yocto-Based SDK Only

Disaggregation then supporting many distributions
WHY CHANG TO LSDK
Changing Requirements for Linux Distributions

• Requirements are changing due to technology shifts and convergences
  - SDN/NFV
  - Appliance / Server convergence

• Demand broadens away from embedded (and NXP SDK) towards enterprise and some enterprise-derived special purpose distributions.

• Biggest reason is convergence of network appliances & servers but also
  • Server ecosystem dominance in ARMv8
  • More powerful SoCs
  • Intel encroachment (WB switches)
  • Ease of use in some cases
  • Standardization
Details of the Changes

- **Server ODMs, OEMs and operators require a single stable image for consolidating all of their server equipment**
  - One unified asset to deploy to all of their equipment (of the same type)
  - Control OPEX related to validation of platforms and management of equipment
- **Embedded Solutions (Yocto, Linaro, Enea) insufficient (for this usage)**
  - Server users (e.g. carrier operators) use automated provisioning to perform a one-time install of a certified Enterprise image to white box & NFV servers
  - Requires inclusion of QorIQ platform support and drivers (esp. net driver)
- **Rely on commercial distributions**
  - Linux kernel and suite of server applications certified to work against it
  - Kitchen sink approach: distribution contains all the platform software the operator may conceivably need, pre-built (i.e. in binary form) and pre-tested
  - Long-term support provided against a stable (i.e. well-tested in field) configuration
- **Limited set of vendors:**
  - Red Hat (primary)
  - Canonical (new entrant – largest platform vendor for OpenStack cloud): Ubuntu
  - SUSE (predominantly Europe)
Changes for DN SW

Applications Solutions Kit
Applications Development Kit
Software Development Kit (Yocto based)

Solution
Distros and Platform
IP Enablement

github
Social Coding
How do DN SDK Respond?

How do we respond?

• Make better use of ecosystem.
  • “disaggregate” NXP-specific SW components so that
    • They fit into multiple distributions.
    • NXP can integrate them
    • 3rd parties can use it
    • Customers can use

“Upstream” SW root stocks

Community “best effort” Support

Special Purpose Community

General Purpose Commercial

Demand for embedded still exists

Demand for embedded still exists

SW component sources, not Linux distributions

Linux distributions

kernel.org
openssl.org
denx.de
Etc.

Yocto / Open Embedded
Fedora
Debian
CentOS
ONL
ONIE
White box switches

wireless ap

FSL SDK
Wind River
Red Hat
Ubuntu

FSD
distributions

OPEN

Embedded

FSF

Linaro/ODP

Etc.
Brush Up on Overview

Yocto-Based SDK Only

Disaggregation then supporting many distributions

- All-in-one SDK
- User Land
- ASK, ADK
- Tools
- Patch
- 3rd party
- Tool-chain

NXP-specific Kernel, Drivers, Patches, Fixes
Tools, Boot-firmware, Cross Compiler
Acceleration Development Toolkit, Applications
NXP
The Benefit of New SDK - LSDK

Flexibility
Customers need to be able to load whatever distro or run whatever open-source components

Scalability
Customers need to run the same software for low-end and high-end deployments

Stability
Customers need to base their development on most recent LTS kernel versions

Consistency
Customers need to be able to move freely between different architectures, x86 or ARM

MARKET DEMANDS OPEN PLATFORM AND STANDARD API
LSDK IN DETAIL
Basic Elements of LSDK

- **Two key components**
  - Linux kernel – standardized to a stable configuration / revision level
  - Root file system – containing user space applications and dynamically loadable kernel modules for standard drivers

- **Commercial distributions usually rely on kernels and user space packages derived from an upstream community-driven distribution (feeder)**
  - E.g. Debian, Slackware, Gentoo
  - Generally share build tools, package management system, etc. with progenitor
  - Frequent cross-pollination between feeder and derivative (i.e. not strictly a *fork*); e.g. derivative re-bases off new feeder releases, bug fixes, enhancements submitted upstream
  - Some commercial distros sponsor community distros; e.g. Red Hat → Fedora, CentOS

- **All derive from a release branch of the mainline kernel.org Linux kernel development tree**
  - Often distinguished by how closely they track to kernel.org releases
  - Community distributions typically released more frequently and closer to kernel.org releases
  - Enterprise distros focus on stability with less frequent releases – based on long-term support “branches”
LTS Kernel Development

DEVELOPMENT FLOW

- Upstream
- Rebase
- Backport
- Publish to Yocto
- Publish to OpenWrt

LTS n-1

LTS n

Kernel.org

NXP IP Owner

BU Developers

Core Team
**RELEASE CYCLE**

1. **Kernel.Org**
   - Merge LSK kernel, apply bug fixes

2. **Staging Repo**
   - Backport NXP and other patches

3. **LSDK Release Cycle**
   - Develop and integrate (rootfs, recipes, ...)

4. **Test and create tags**

5. **Publish, make official release**
WHAT’S ON THE BOARD

• Boards are shipped with:
  • NOR image in Bank0/Bank4
  • Boot image and rootfs on SD card
• NOR Image consists of:
  • Boot firmware (u-boot)
  • RCW
  • PHY firmware
  • DPAA firmware (fman, MC)
• Minimal busybox rootfs
• Installer to install Ubuntu rootfs on SD consists of:
  • Standard distro rootfs
  • NXP specific user space: restool, aiop_tool, fmc

BETTER CUSTOMER EXPERIENCE

Majority Customers
• Self-sufficient
• Download toolchain from Linaro, distros from somewhere, NXP components from GIT, ...

Curious Customers
• Demand integrated vertical solution software
• Solution team delivers package on target hybrid distro

Old-School Customers
• NPI or Support team produces the YOCTO-based SDK
<table>
<thead>
<tr>
<th>Region 1 (4KB)</th>
<th>Region 2 (64MB)</th>
<th>Region 3 (20MB)</th>
<th>Region 4 (300MB)</th>
<th>Region 5 (remaining space of disk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBR/GPT</td>
<td>Firmware</td>
<td>Partition 1 (FAT32) EFI</td>
<td>Partition 2 (EXT4) Boot Partition</td>
<td>Partition 3 (EXT4) rootfs</td>
</tr>
<tr>
<td></td>
<td>RCW</td>
<td>BOOTAA64.EFI grub.cfg</td>
<td>Kernel image</td>
<td>Ubuntu</td>
</tr>
<tr>
<td></td>
<td>U-boot or UEFI</td>
<td></td>
<td>DTBs</td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>Eth PHY firmware</td>
<td></td>
<td>Flex Installer_&lt;arch&gt;.itb</td>
<td>Ubuntu-Core</td>
</tr>
<tr>
<td></td>
<td>QE/uQE firmware</td>
<td></td>
<td>Distro boot scripts</td>
<td>CentOS</td>
</tr>
<tr>
<td></td>
<td>FMan firmware</td>
<td></td>
<td>Secure headers</td>
<td>Debian</td>
</tr>
<tr>
<td></td>
<td>MC firmware</td>
<td></td>
<td>Other</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PPA firmware</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kernel image</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DTB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ramdisk RFS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ubuntu
Ubuntu-Core
CentOS
Debian
## Layerscape SDK vs. QorIQ SDK

<table>
<thead>
<tr>
<th>Feature</th>
<th>QorIQ SDK</th>
<th>Layerscape SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
<td>LTS kernel, platform drivers, tools</td>
<td>Choice of 2 LTS kernels, platform drivers, tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Available as components too.</td>
</tr>
<tr>
<td>User-space</td>
<td>Yocto</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>Build-tool</td>
<td>Yocto</td>
<td>Ubuntu, make, flexbuild</td>
</tr>
<tr>
<td>Build Environment</td>
<td>Host</td>
<td>Host, Target</td>
</tr>
<tr>
<td>Boot/recovery options</td>
<td>Flash, network</td>
<td>Flash, network, SD card, HDD</td>
</tr>
<tr>
<td>Package Installation</td>
<td>Integrate into Yocto, build image, re-flash board.</td>
<td>Apt-get over network</td>
</tr>
<tr>
<td>Downloadable</td>
<td>Giant ISO with sources and binaries for all platforms</td>
<td>Individual Binaries, Individual components source</td>
</tr>
</tbody>
</table>

Layerscape SDK **provides more, not less.** Layerscape SDK is **Easier to Use**.
HOW TO USE LSDK
GitHub Access for LSDK

LSDK Open Source

- Enable the QorIQ® Layerscape Processors Based on ARM Technology.
- Disaggregated components of LSDK are available in github.
- Ubuntu-based userland for ARMv7 and ARMv8 targets.
- By downloading the LSDK components, you agree to the EULA (End User License Agreement).
- Flexbuild scripts and prebuilt images need to be downloaded from www.nxp.com.
Download Components from GitHub

[Image showing the LSDK Open Source website with components listed and links to GitHub]
Clone Linux from GitHub

No description, website, or topics provided.

- 568,026 commits
- 3 branches
- 3 releases
- 6,479 contributors

sudeep-holla committed with Xie Xiaobo
irqchip/gicsv: Remove disabling redistributor and group1 non-secure l...
Latest commit ec61f52 on 17 Aug 2016

- Documentation
- arch
  - arm64: dts: update the cpu idle node 24 days ago
Flexbuild is an integrated build system with flexible system build and distro installation.

The LSDK build system includes three major components: package builder, rootfs maker and image installer.

The utility can run on x86 host of Ubuntu 16.04, arm targets and docker container.
Build LSDK using Flexbuild

- **General build command**
  - $ tar xvzf flexbuild_<version>.tgz
  - $ cd flexbuild
  - $ source setup.env
  - $ flex-builder -i repo-fetch
  - $ flex-builder -i repo-tag (check out tags specified in file build_lsdk1706.cfg)

- **Build custom kernel and update the boot partition**
  - $ flex-builder -c linux -B menuconfig
  - $ flex-builder -i uimg
  - $ flex-builder -i mkbootpartition
  - $ cd build/qoriq-linux/kernel/arm64/lib && tar cvzf modules.tgz modules

- **Build custom u-boot or application**
  - $ flex-builder -c uboot -m <machine> -b <boottype> #build uboot for <machine> to generate specified nor/sd/qspi boot image
  - $ flex-builder -c <component> -a <arch> #build single application component for specified <arch>
Deploy LSDK Images on the target board

**Deploy LSDK images from Linux Host**

- $ wget http://www.nxp.com/lgfiles/sdk/lsdk1706/firmware_ls1088ardb_uboot_sdboot.img
- Or $ flex-builder -i mkrsfs -a <arch> -B additional_packages_list_full
- $ flex-installer --bootpart=bootpartition_arm64.tgz --rootfs=build/images/ubuntu_xenial_arm64_rootfs.d --firmware=firmware_ls1088ardb_uboot_sdboot.img --machine=ls1088ardb --device=/dev/sdX

**Deploy LSDK images from Target board**

- Download LSDK composite firmware from NXP website
  - E.g. $ wget http://www.nxp.com/lgfiles/sdk/lsdk1706/firmware_ls2088ardb_uboot_norboot.img
- Put LSDK composite firmware to a TFTP server, then download the firmware via TFTP to the target board under the U-Boot prompt
- Reset the board and deploy boot partition and Ubuntu 16.04 userland to SD/USB/SATA.
  - Enable network connection to download LSDK images
  - Use flex-installer to create and format partitions
  - $ flex-installer -i install --bootpart=bootpartition_arm64.tgz --rootfs=ubuntu_xenial_arm64_rootfs.tgz --machine=ls2088ardb --device=usb
Add a Package using Flexbuild

- **How to add a package not officially supported by Ubuntu user land during build stage**
  
  - add extrinsic package name to extrinsic_packages_list in packages/apt-packages/additional_packages_list
  
  - put custom script of extrinsic package to packages/apt-packages/extrinsic-pkg (e.g. refer to nginx.sh)
  
  - run flex-builder -i mkrfs -a <arch> to generate new Ubuntu rootfs
  
  - install the new Ubuntu rootfs to target machine via flex-installer
PLAN AND ROADMAP
Layerscape SDK Roadmap

LSDK-17.03
31-Mar

- LTS 4.4
- Ubuntu user land and toolchain gcc-5.4
- Installer of packages
- Github release
- PoC, limited support on LS1043A, LS2088A

LSDK-17.06
28-July

- LTS 4.4, base off LSK 4.4, incl. kaslr
- U-boot 2017.03
- UEFI
- ARM v7 and v8 platforms (including LS1088A), except for LS1012A rev1.0.
- MC updates 10.2.2
- ODP, DPDK and user space packages
- New Infocenter structure

LSDK-17.09
29-Sep

- LTS 4.9
- LS1012A rev 1.0 & 2.0, LS2088A rev1.1
- ODP on DPAA1
- Cherry picks the fixes from the latest Linaro LTS
- Additional functionality of IP
- Additional errata software workarounds and software fixes

LSDK-17.12
29-Dec

- Additional functionality of IP
- Additional errata software workarounds and software fixes
- More…

Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Summary

- LSDK is a new form of Linux from NXP DN, and consist of a set of disaggregated components based on Linux distributions.
- Meet market demand to more Linux distributions of more types, and satisfy the requirement from a wide variety of customers.
- We can use Flexbuild to build all packages from LSDK, make Root filesystem and generate the installer.

All you need to get your product to market faster!
SECURE CONNECTIONS FOR A SMarter WORLD

In case of any question’s please contact,
Xie Xiaobo at Xiebo.xie@nxp.com
Cynthia Fu at cynthia.fu@nxp.com