YOCTO KNOWLEDGE SHARING

Renesas Design Vietnam Co., Ltd.
Khoa Thien Tran, PFSolution Team, Software Department

April 04, 2013 Rev. 1.00

Outline

- Overview
  - Challenges Developing Today's Embedded Software
  - Yocto Project Introduction
  - Whole Image
  - Benefits of Yocto Project
- Yocto Build System
  - Workflow
  - Configuration
  - Recipe
  - Layer
- Structure Of Directory
- Use Cases
  - Armadillo800EVA BSP Layer
  - Add Gstreamer

Overview

- ***It's not an embedded Linux distribution***
- ***It creates a custom one for you.***

yocto PROJECT

Source: www.yoctoproject.org

Challenges Developing Today's Embedded Software

- SoCs have brought fantastic capabilities to embedded products.
- But... it means that rapidly developed application software is a key part of the solution.
- Application developer is quite often a different role/skillset than the system developer.

- ***Challenges:***
  - Support the app developer role with their own environment.
  - Quickly roll out new applications that utilize features in the silicon and meet time to market (TTM) demands is paramount

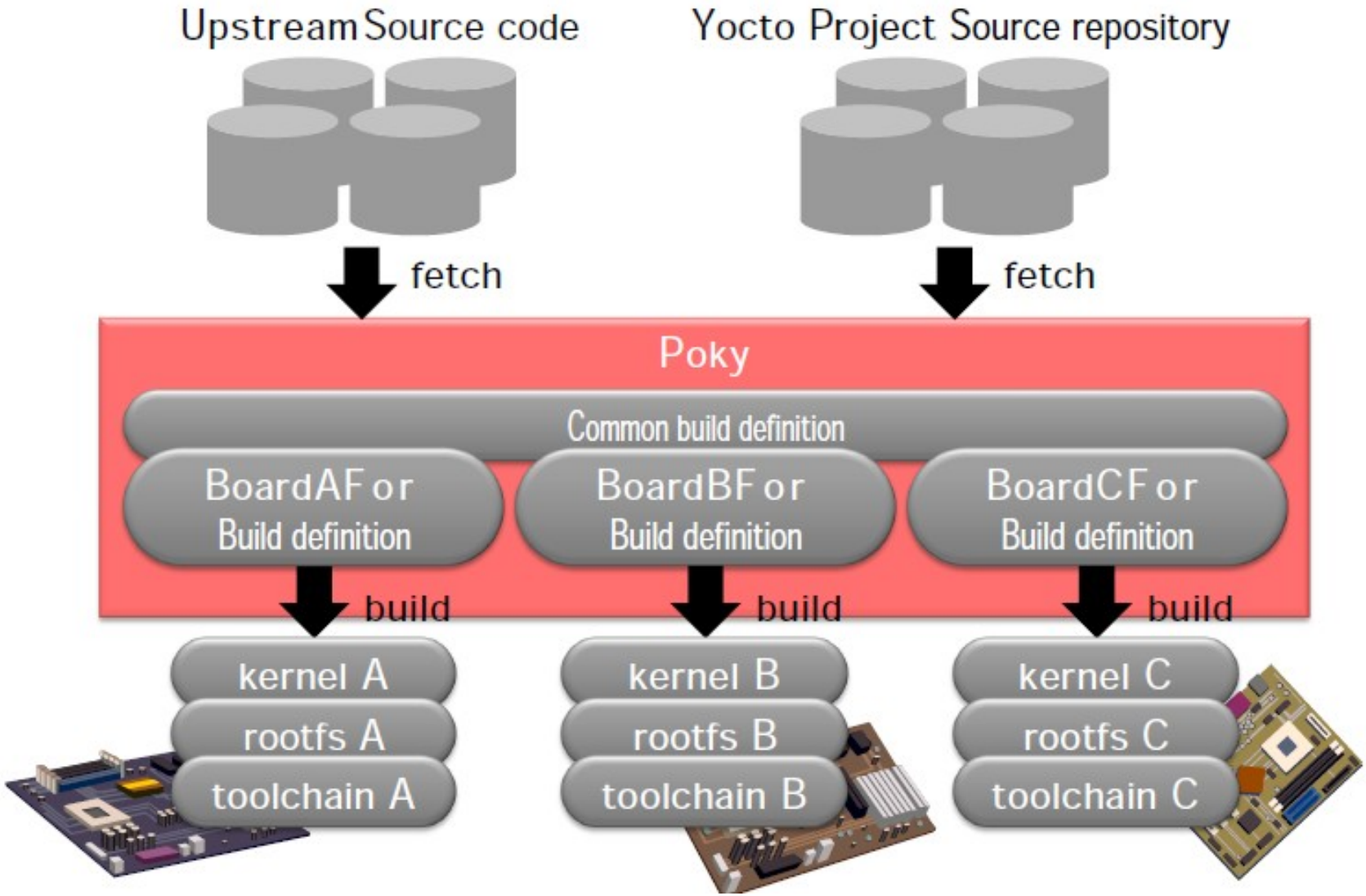
Source: "The Answer to Effective Embedded Application Development" - David Stewart

Yocto Project Introduction

- The Yocto Project* is an open source collaboration project
  - Provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of hardware architecture.
- Focused resources for system application developers who need to customize a Linux distribution for a device.
- Validated and tested BSPs in a common format.
- Automatically creates an application development SDK customized for each specific device.
- Supported by embedded industry leaders across multiple architectures (IA, ARM, PowerPC, MIPS, etc).

Source: Intel Corporation

Whole Image



Source: Developing Embedded Linux by Poky - HAYASHI Kazuhiro

Benefits of Yocto Project

- Linux is becoming increasingly popular for Embedded.
- Non-commercial and commercial embedded Linux has many distros
  - Developers spend lots of time porting or making build systems
  - Leaves less time/money to develop interesting software features
- The industry needs a common build system and core technology.
- Industry leaders have joined together to form the Yocto Project, the benefit of doing so is:
  - Less time spent on things which don't add value (build system, core

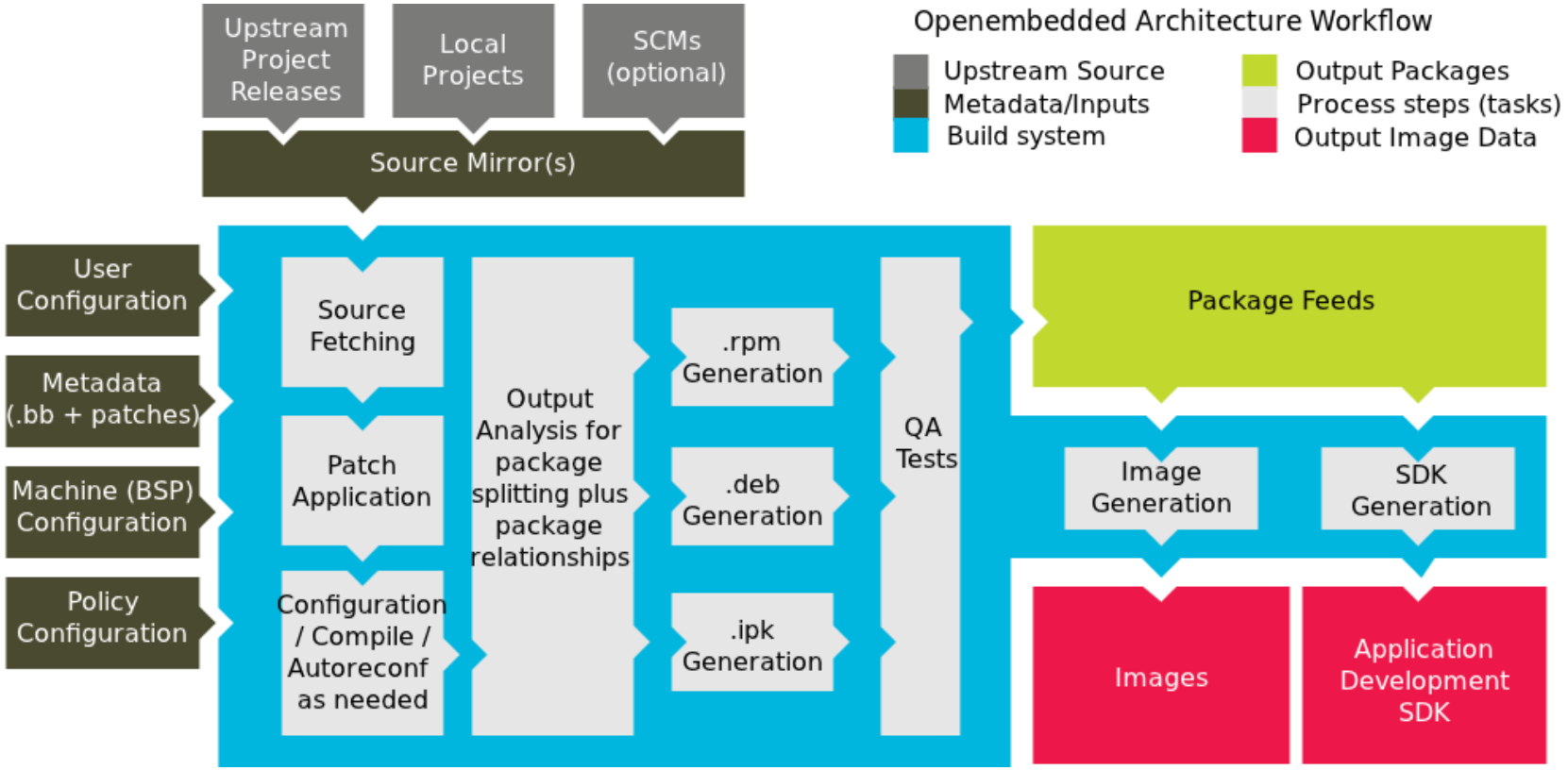
    Linux components)
  - Increased ability to enable key silicon features
  - Linux grows more in embedded

Source: Intel Corporation

Yocto Build System

- ***It's not an embedded Linux distribution***
- ***It creates a custom one for you.***

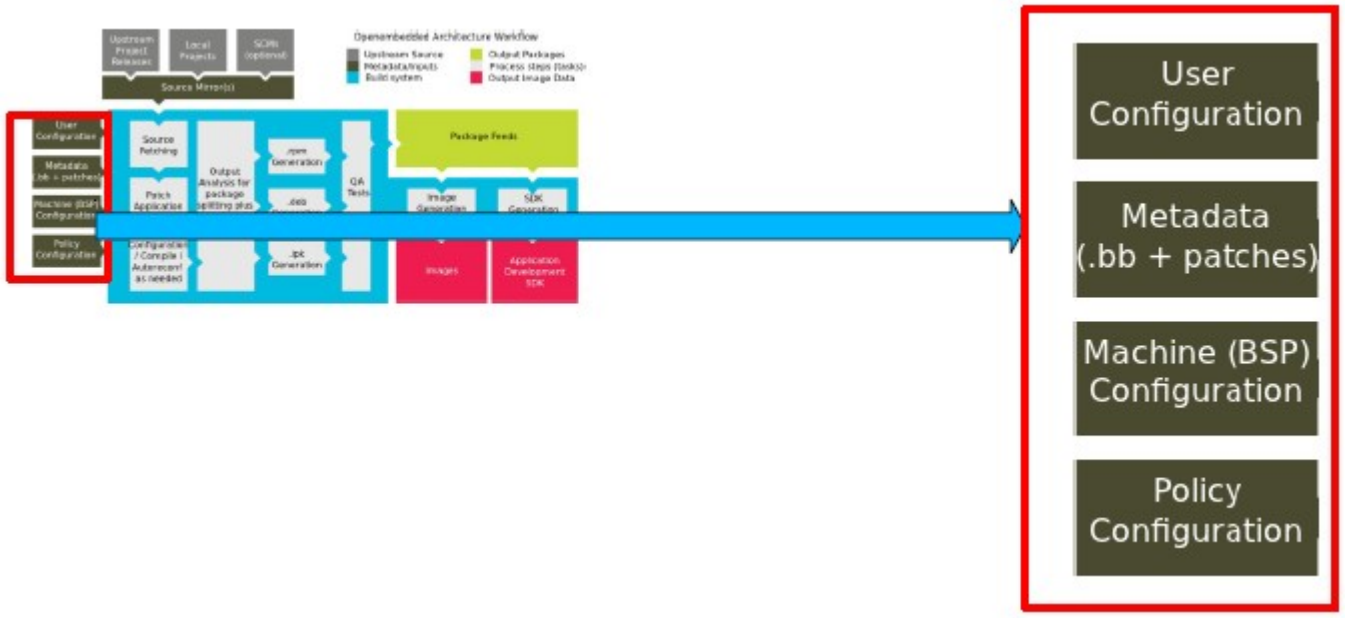
Workflow



- bitbake + metadata (Poky)
  - Bitbake – a task executor and scheduler
  - metadata – task definitions in recipes, classes + config

Configuration

- Configuration (*.conf) – global definition of variables
  - build/conf/local.conf
    - Local user-defined variables
  - meta-renesas/conf/machine/armadillo800eva.conf
    - Machine specific variables
  - build/conf/bblayers.conf
    - Layer building declaration

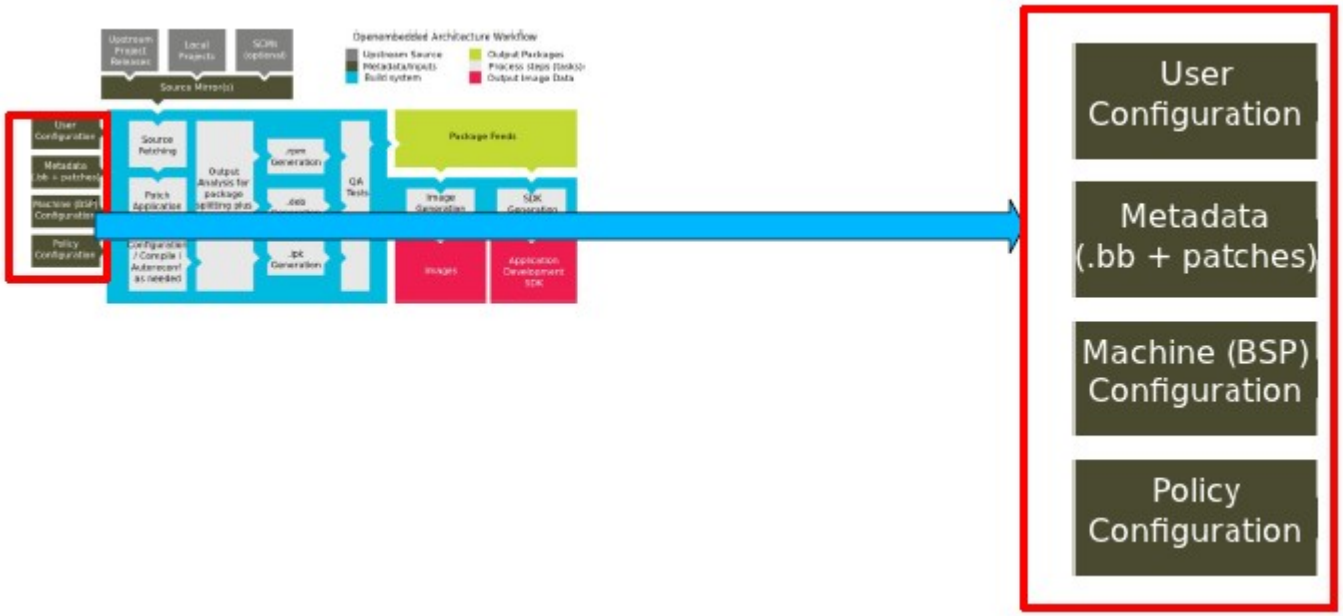


User Configuration

- BB_NUMBER_THREADS
  - Determines how many tasks bitbake should run in parallel
- PARALLEL_MAKE
  - Controls how many processes make should run in parallel when running compile tasks
- MACHINE
  - Determines CPU architecture or hardware board target machine
- IMAGE_INSTALL_append
  - Adds modules, features

Recipe

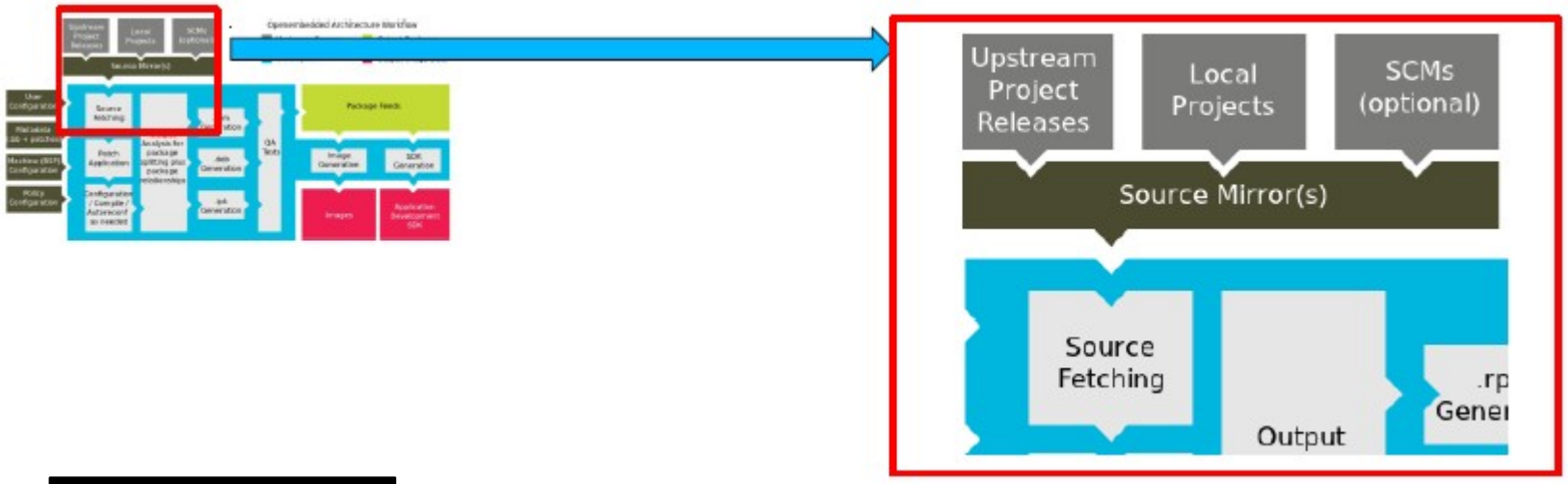
- A recipe(*.bb) is a set of instructions for building packages, including:
  - Where to obtain the upstream sources and which patches to apply
  - Dependencies (on libraries or other recipes)
  - Configuration/compilation options
  - Define what files go into what output packages



Standard Recipe Build Steps

- Building recipes involves executing the following functions, which can be overridden when needed for customizations
  - do_fetch
    - Download archive of source code from server(git, svn, website, local...)
  - do_unpack
    - Unpack the archive of source code
  - do_patch
    - Apply the patch file to source code
  - do_configure
    - Configure (create make files, configuration files for building)
  - do_compile
    - Make (Build programs, libraries, documentation,...)
  - do_install
    - Make install (Install what needs to be installed)
  - do_package
    - Create output package

Source Fetching

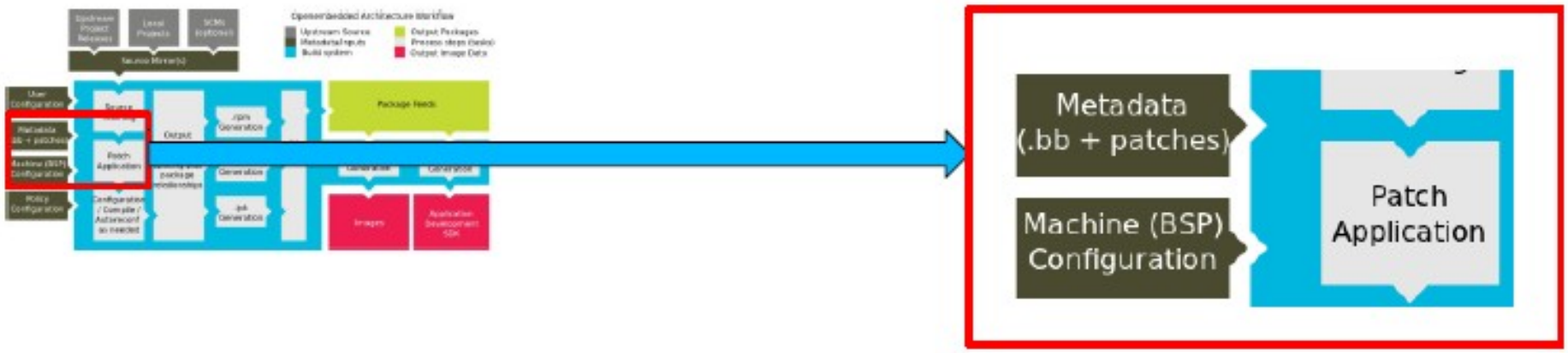


gstreamer_0.10.36.bb

```
SRC_URI = "http://gstreamer.freedesktop.org/src/gstreamer/gstreamer-${PV}.tar.bz2 \
           file://check_fix.patch \
           file://gst-inspect-check-error.patch"
```

- Sources will be downloaded based on the links that are described by macro "*SRC_URI*".
- "*SRC_URI*" can be local or in the network.
- Bitbake can fetch from various types
  - git, svn, bzr, from tarballs, and many, many more*

Patching

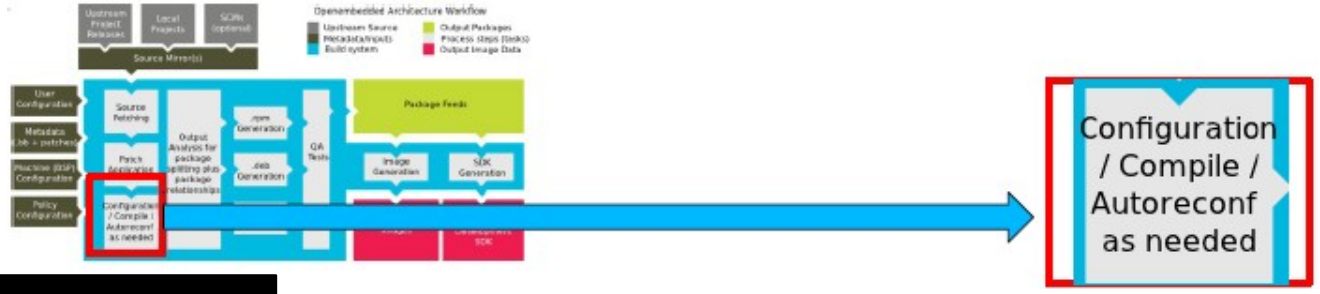


gstreamer_0.10.36.bb

```
SRC_URI = "http://gstreamer.freedesktop.org/src/gstreamer/gstreamer-${PV}.tar.bz2 \
           file://check_fix.patch \
           file://gst-inspect-check-error.patch"
```

- When sources are unpacked, the patches are applied.
- The patch files are described via "*SRC_URI*".

Configure/Compile



gstreamer_0.10.36.bb

```
SUMMARY = "GStreamer multimedia framework"
DESCRIPTION = "GStreamer is a multimedia framework for encoding and decoding video and sound. \
It supports a wide range of formats including mp3, ogg, avi, mpeg and quicktime."
HOMEPAGE = "http://gstreamer.freedesktop.org/"
BUGTRACKER = "https://bugzilla.gnome.org/enter_bug.cgi?product=Gstreamer"
SECTION = "multimedia"
LICENSE = "LGPLv2+"

DEPENDS = "glib-2.0 libxml2 bison-native flex-native"

inherit autotools pkgconfig gettext

GSTREAMER_DEBUG ?= "--disable-debug"
EXTRA_OECONF = "--disable-nls --disable-static --disable-loadsave"
```

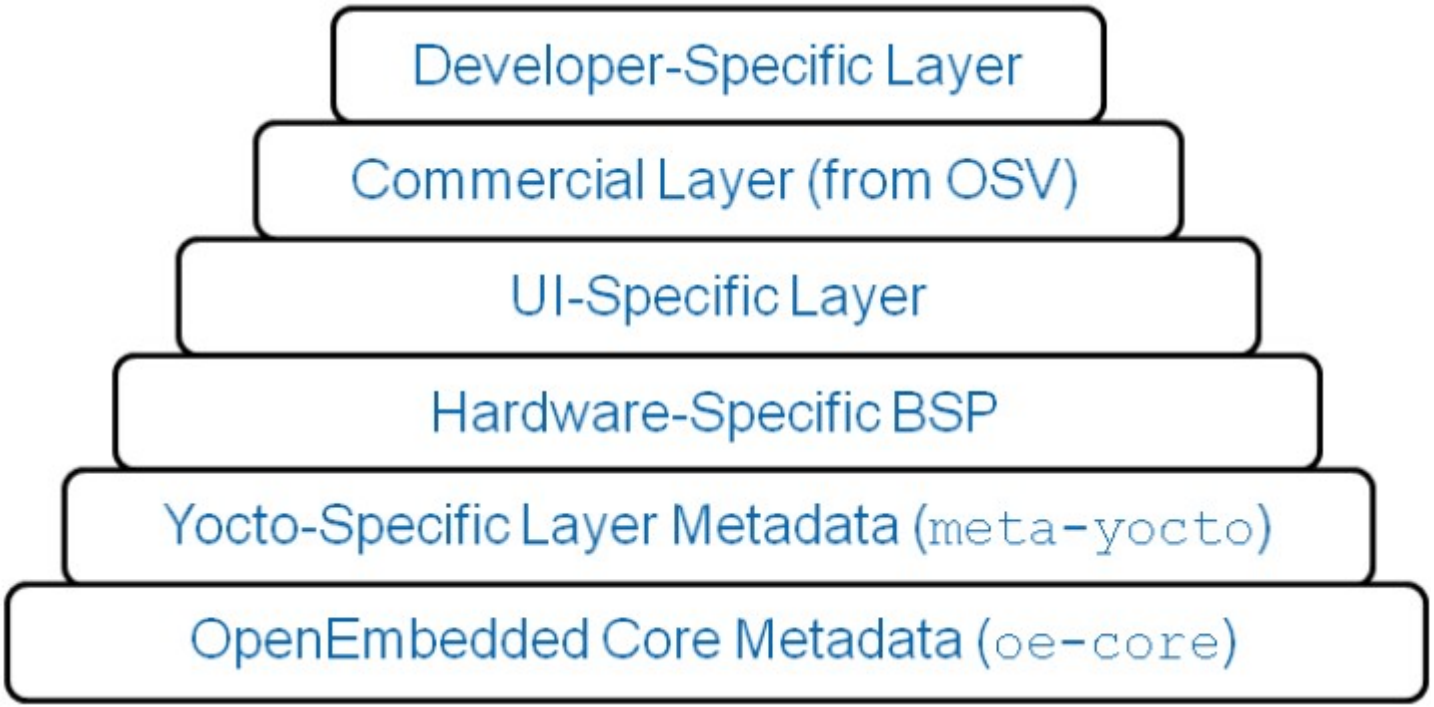
- After the patching operation is finished, configure/compile operation will be executed.
- "*DEPENDS*" the list of recipes which should be done beforehand.
- configure/compile operation are based on the declaration of configuration tools, build flags, build options.

Packaging

- After configure/compile/install operation are completed, packaging operation is started.
- The most popular package formats are supported: RPM, Debian, and ipk.
  - Set PACKAGE_CLASSES in conf/local.conf

Layer

- The Yocto Project build system is composed of layers.
- A layer is a logical collection of recipes representing the core, a Board Support Package (BSP), or an application stack.
- Layer have a configuration file, class directory, recipes directory, priority information.



Using Layers

- Folders that are layers begin with the string meta.
  - For example: meta, meta-hob, meta-skeleton, meta-yocto, and meta-yocto-bsp
- Layers are added to your build by editing the build/conf/bblayers.conf file:

```
BBLAYERS = " \
    /data/poky/meta \ # core system
    /data/poky/meta-yocto \ # yocto config and recipes
    /data/meta-renesas \ # machine BSP layer
"
```


BSP Layer

- BSPs are layers to enable support for specific hardware platforms.
- Add machine settings and recipes.
- Machine settings are specified in a layer's conf/machine/xxx.conf file(s)
  - Example:

    meta-renesas/conf/machine/armadillo800eva.conf
- A BSP developer's task is to create a machine layer.
  - Define a machine configuration to match hardware.
  - Add machine-specific recipes or extend existing recipes.

Example layout of a BSP layer

./COPYING.MIT
./README
./classes/
./classes/sdcard_image.bbclass
./conf/
./conf/layer.conf
./conf/machine/armadillo800eva.conf
./files
./files/device_table_add-rmobile.txt
./recipes-bsp
./recipes-bsp/u-boot
./recipes-bsp/u-boot/u-boot_git.bb
./recipes-kernel
./recipes-kernel/linux/linux-yocto
./recipes-kernel/linux/linux-yocto/armadillo800eva
./recipes-kernel/linux/linux-yocto/armadillo800eva/0001-ARM-shmobile-add-common-DMAEngine-definitions.patch
./recipes-kernel/linux/linux-yocto/r8a7740
./recipes-kernel/linux/linux-yocto/r8a7740/defconfig
./recipes-kernel/linux/linux-yocto/sh73a0
./recipes-kernel/linux/linux-yocto/sh73a0/defconfig
./recipes-kernel/linux/linux-yocto_3.4.bbappend
./recipes-kernel/linux/linux-libc-headers

Layer config file

Machine config file

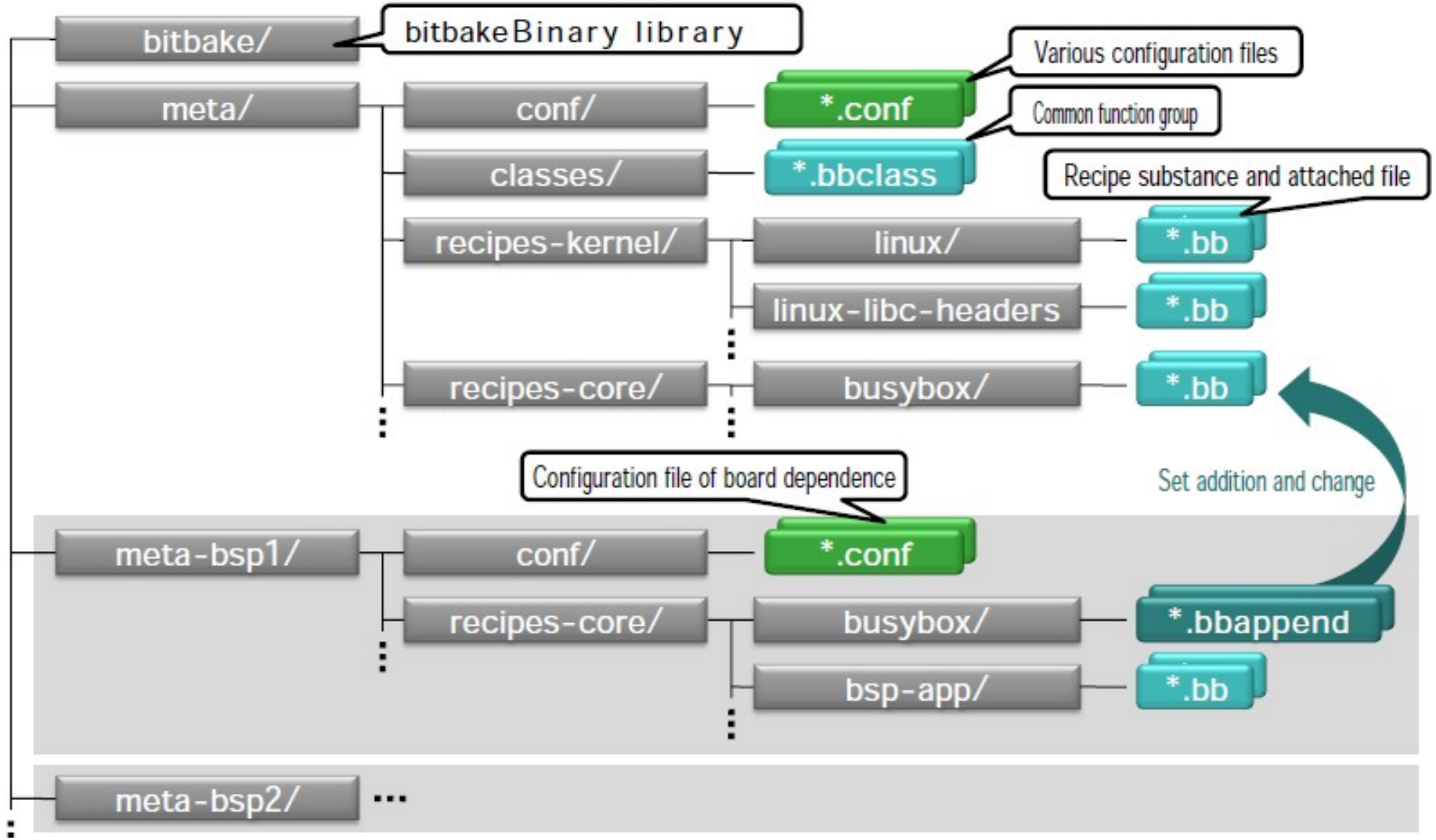
Uboot recipe

Kernel 3.4 recipe append

Structure Of Directory

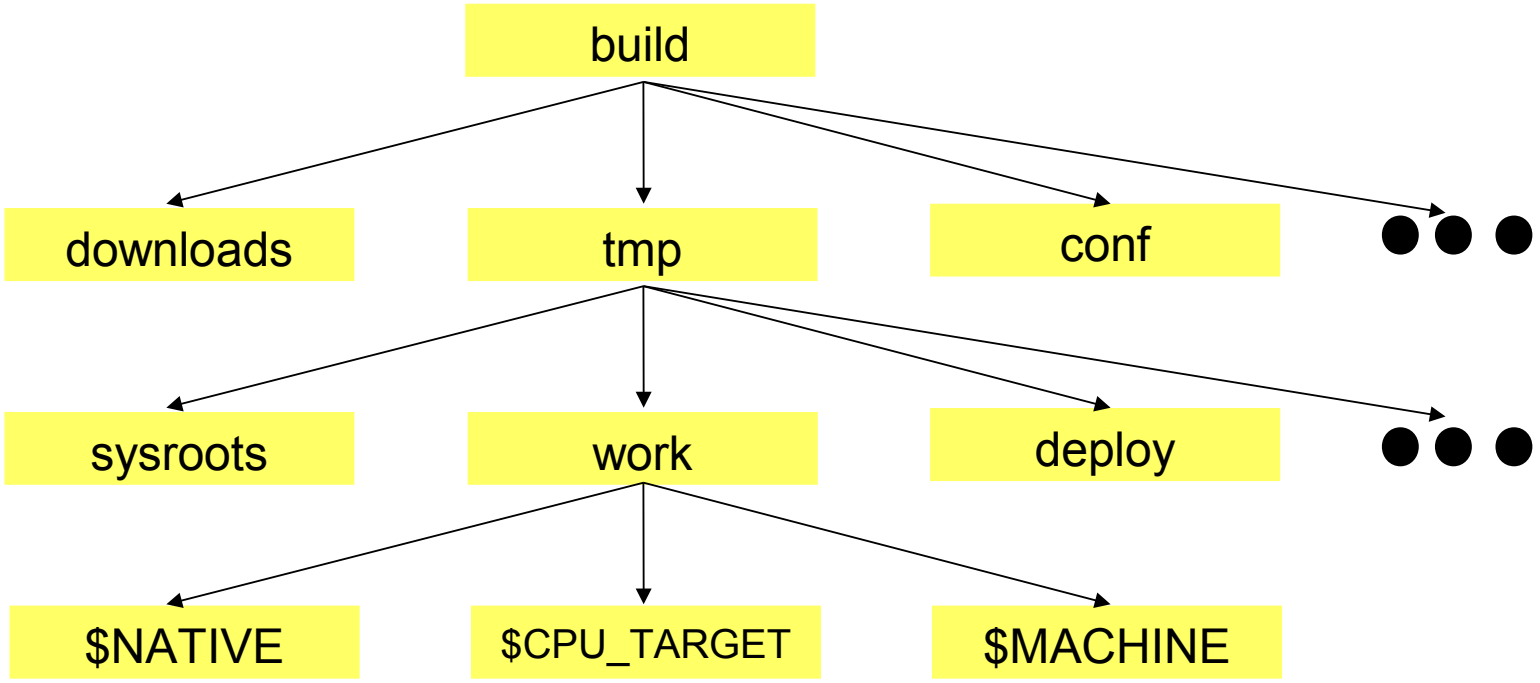
- ***It's not an embedded Linux distribution***
- ***It creates a custom one for you.***

Structure Of Directory

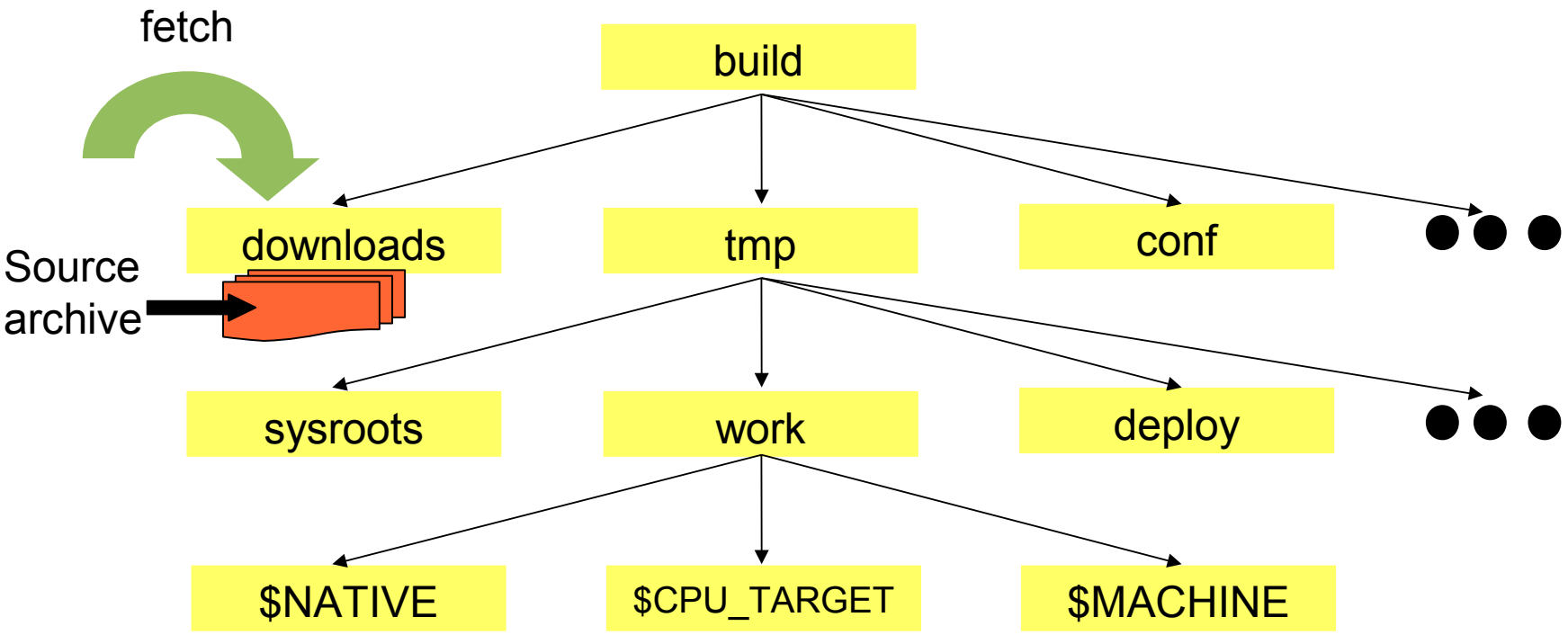


Source: Developing Embedded Linux by Poky - HAYASHI Kazuhiro

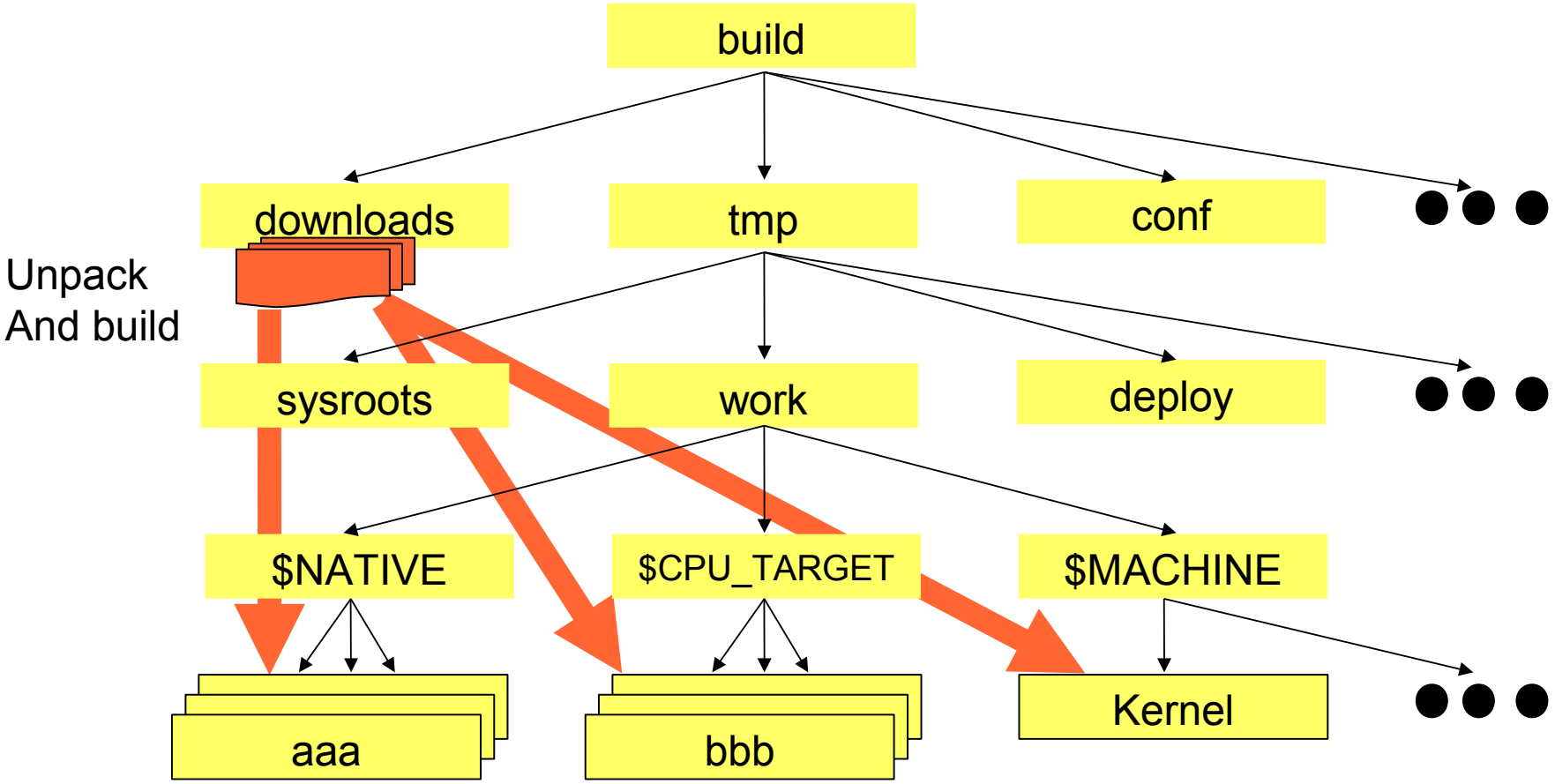
Structure Of Build Directory



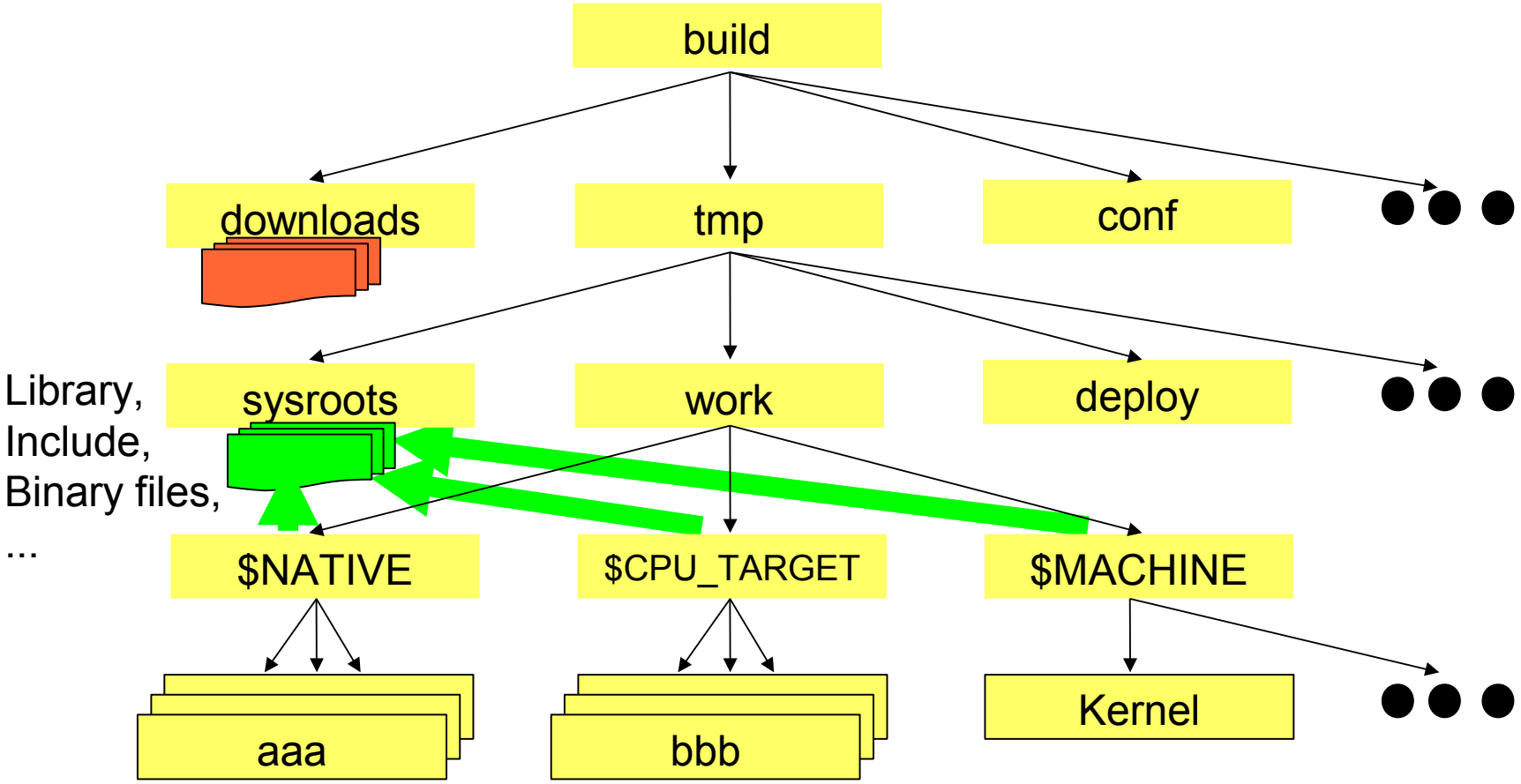
Structure Of Build Directory



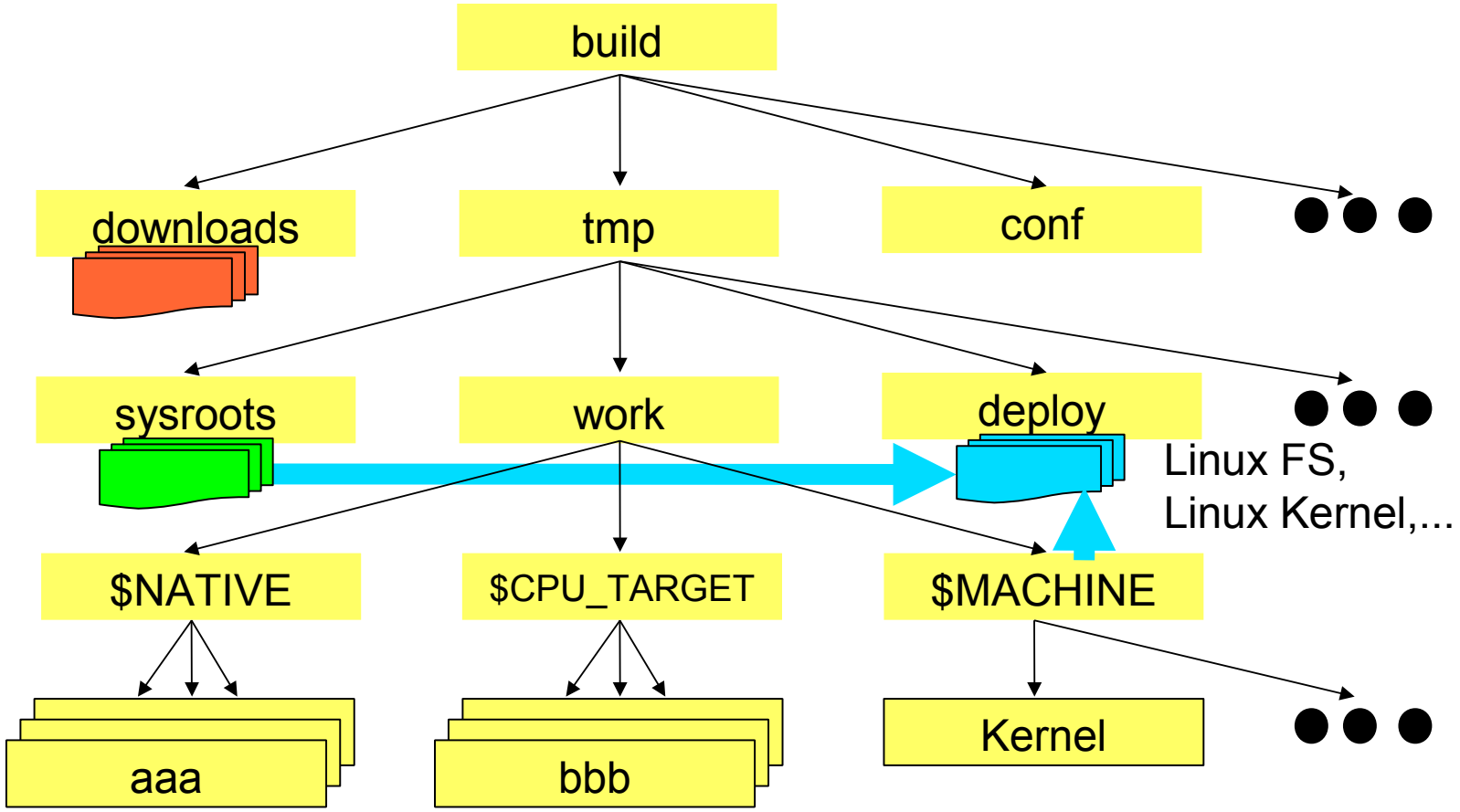
Structure Of Build Directory



Structure Of Build Directory



Structure Of Build Directory



Armadillo800EVA BSP Layer

- *It's not an embedded Linux distribution*
- *It creates a custom one for you.*

Layout of Armadillo800EVA BSP layer

./COPYING.MIT
./README
./classes/
./classes/sdcard_image.bbclass
./conf/
./conf/layer.conf
./conf/machine/armadillo800eva.conf
./files
./files/device_table_add-rmobile.txt
./recipes-bsp
./recipes-bsp/u-boot
./recipes-bsp/u-boot/u-boot_git.bb
./recipes-kernel
./recipes-kernel/linux/linux-yocto
./recipes-kernel/linux/linux-yocto/armadillo800eva
./recipes-kernel/linux/linux-yocto/armadillo800eva/0001-ARM-shmobile-add-common-DMAEngine-definitions.patch
./recipes-kernel/linux/linux-yocto/r8a7740
./recipes-kernel/linux/linux-yocto/r8a7740/defconfig
./recipes-kernel/linux/linux-yocto/sh73a0
./recipes-kernel/linux/linux-yocto/sh73a0/defconfig
./recipes-kernel/linux/linux-yocto_3.4.bbappend
./recipes-kernel/linux-libc-headers
./recipes-kernel/linux-libc-headers/linux-libc-headers-rmobile_git.bb

Layer Configuration

Meta-renesas/conf/layer.conf

```
# We have a conf and classes directory, append to BBPATH
BBPATH .= ":${LAYERDIR}"

# We have a recipes directory, add to BBFILES
BBFILES += "${LAYERDIR}/recipes*/*/*.bb ${LAYERDIR}/recipes*/*/*.bbappend"

BBFILE_COLLECTIONS += "meta-renesas"
BBFILE_PATTERN_meta-renesas := "^${LAYERDIR}/"
BBFILE_PRIORITY_meta-renesas = "5"
```

- BBPATH .= ":${LAYERDIR}"
  - Add layer to poky build system
- BBFILES += "${LAYERDIR}/recipes*/*/*.bb ${LAYERDIR}/recipes*/*/*.bbappend"
  - Add new recipe and recipe extention
- BBFILE_COLLECTIONS += "meta-renesas"
  - BSP name

Machine Configuration

Meta-renesas/conf/machine/armadillo800eva.conf

```
require conf/machine/include/r8a7740.inc
UBOOT_MACHINE = "armadillo-800eva_config"
KERNEL_IMAGETYPE = "zImage"
KERNEL_DEVICETREE = "${S}/arch/arm/boot/dts/r8a7740-armadillo800eva.dts"
PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto"
PREFERRED_VERSION_linux-yocto ?= "3.4%"
IMAGE_FSTYPES += "tar.bz2"
SERIAL_CONSOLE = "115200 ttySC1"
MACHINE_EXTRA_RRECOMMENDS = "kernel-modules"
MACHINE_FEATURES = "serial mmc alsa ext2 touchscreen usbhost vfat ethernet"
```

- **CPU setting**
  - conf/machine/include/r8a7740.inc
- **Kernel setting**
  - KERNEL_IMAGETYPE = "zImage"
  - KERNEL_DEVICETREE = "${S}/arch/arm/boot/dts/r8a7740-armadillo800eva.dts"
  - PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto"
  - PREFERRED_VERSION_linux-yocto ?= "3.4%"
  - SERIAL_CONSOLE = "115200 ttySC1"
  - MACHINE_EXTRA_RRECOMMENDS = "kernel-modules"
  - MACHINE_FEATURES = "serial mmc alsa ext2 touchscreen usbhost vfat ethernet"
- **RootFS setting**
  - IMAGE_FSTYPES += "tar.bz2"

Kernel Customization

Meta-renesas/recipes-kernel/linux/linux-yocto_3.4.bbappend

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

COMPATIBLE_MACHINE_armadillo800eva = "armadillo800eva"
KBRANCH_DEFAULT_armadillo800eva = "standard/armadillo800eva"
KBRANCH_armadillo800eva = "${KBRANCH_DEFAULT}"
KMACHINE_armadillo800eva = "armadillo800eva"

SRC_URI_append_armadillo800eva = " \
file://0001-sh-clkfwk-Support-variable-size-accesses-for-MSTP-cl.patch \
file://0002-sh-clkfwk-Support-variable-size-accesses-for-div4-di.patch \
file://0003-sh-clkfwk-Move-to-common-clk_div_table-accessors-for.patch \
"
```

- Machine branch
  - KMACHINE_armadillo800eva = "armadillo800eva"
- Declare Armadillo patch files
  - SRC_URI_append_armadillo800eva = " \
  - file://0001-sh-clkfwk-Support-variable-size-accesses-for-MSTP-cl.patch \
  - file://0002-sh-clkfwk-Support-variable-size-accesses-for-div4-di.patch \
  - file://0003-sh-clkfwk-Move-to-common-clk_div_table-accessors-for.patch \"

Build Armadillo800EVA BSP Layer

- Open “bblayers.conf ” file in “build/conf/” and add Armadillo BSP information:

```
BBLAYERS = " \
    /data/poky/meta \ # core system
    /data/poky/meta-yocto \ # yocto config and recipes
    /data/meta-renesas \ # machine BSP layer
"
```

- Open “local.conf ” file in “build/conf/” and add Armadillo machine information:

```
#
# Machine Selection
#
MACHINE ??= "armadillo800eva"
```


Build Armadillo800EVA BSP Layer

- cd ~/poky-danny-8.0/
- source oe-init-build-env
  - Sets up important environment variables
- bitbake -c fetchall core-image-minimal
  - Download all necessary Yocto packages
- bitbake -k core-image-minimal
  - Builds a minimal Linux image for the Armadillo800EVA target

The output that contain Linux Kernel and Linux Root FS will be stored at:
~/poky-danny-8.0/build/tmp/deploy/images/

Add Gstreamer

- *It's not an embedded Linux distribution*
- *It creates a custom one for you.*

Gstreamer In Yocto

- Gstreamer is located at:

~/poky-danny-8.0/meta/recipes-multimedia/gstreamer

- It contents some plugins as below:
  - gst-ffmpeg_0.10.13.bb
  - gst-plugins-bad_0.10.23.bb
  - gst-plugins-base_0.10.36.bb
  - gst-fluendo-mpegdemux_0.10.71.bb
  - gst-plugins-good_0.10.31.bb
  - gst-meta-base_0.10.bb
  - gst-plugins-ugly_0.10.19.bb
  - gst-openmax_0.10.1.bb
  - gst-fluendo-mp3_0.10.19.bb
  - gstreamer_0.10.36.bb

Building Gstreamer

- By default, core-image-minimal is not enable Gstreamer.

- Core-image-minimal must be customized to add Gstreamer recipe.
  - Open *~/poky-danny-8.0/build/conf/local.conf*

    At the end of file, add below information:

    ```
    IMAGE_INSTALL_append = " gst-meta-base"
    ```

- Rebuild core-image-minimal

Reference

- Yocto project website
  www.yoctoproject.org/

- Yocto Project Development Manual
  http://www.yoctoproject.org/docs/1.3/dev-manual/dev-manual.html

- Wiki main page
  https://wiki.yoctoproject.org/wiki/Main_Page

- RVC Yocto Setup Guideline

Renesas Design Vietnam Co., Ltd.