Create a custom BSP layer for Intel SoC or FPGA

It is possible to customize existing meta layers by creating an own meta layer with bitbake append files. In this description the meta layer meta-altera for Intel FPGA devices is used as reference.

Table of contents

For more informations about the OS used and the sources and versions of yocto project or meta-altera layer, see Yocto KICKstart#Used source files.

Initialize the environment

Before starting to create a custom meta layer, navigate to the yocto poky directory and initialize the environment with the following commands:

```
cd path/to/yocto/poky
source oe-init-build-env
```

Download required sources

Since the created custom layer should depend on meta-altera, these sources files must be downloaded and added to the yocto project.

1. Navigate to yocto poky directory and download meta-altera layer:

   ```
   cd path/to/yocto/poky
   git clone -b dunfell https://github.com/kraj/meta-altera.git
   ```

2. Add meta-altera to yocto project with:

   ```
   cd path/to/yocto/poky/build
   bitbake-layers add-layer ./meta-altera
   ```

Create BSP layer

In this example, the created layer is named meta-custom. If you want to use a different name, just replace "custom".

1. Create own layer "meta-custom" with:

   ```
   bitbake-layers create-layer path/to/yocto/poky/meta-custom
   ```

2. The created layer contains a sample recipes folder, that can be removed:

   ```
   rm -rf path/to/yocto/poky/meta-custom/recipes-example
   ```

3. Make meta-custom dependent on meta-altera:

```
sed -i '/LAYERDEPENDS_meta-custom/s/core/meta-altera/g' path/to/yocto
/poky/meta-custom/conf/layer.conf
```

4. Add the meta-custom layer to the yocto project with:

```
bitbake-layers add-layer path/to/yocto/poky/meta-custom
```

Configure BSP Machine

In the machine configuration file, you can store general settings for the custom BSP layer, such as the version of u-boot or the linux kernel. Here, we will use the cyclone5 machine configuration file from the meta-altera layer as a template. For other machine configurations in the meta-altera layer, see meta-altera/conf/machine.

1. Download the cyclone5 machine configuration file to the meta-custom layer and rename it:

```
mkdir -p path/to/yocto/poky/meta-custom/conf/machine
cd path/to/yocto/poky/meta-custom/conf/machine
wget https://github.com/kraj/meta-altera/raw/dunfell/conf/machine
/cyclone5.conf
mv cyclone5.conf custom.conf
```

2. Make the following changes to the content of custom.conf:
    a. Change the name in line 2 from 'cyclone5' to 'custom' and customize the variables UBOOT_EXTLINUX_DEFAULT_LABEL and UBOOT_EXTLINUX_MENU_DESCRIPTION_default:

    ```
    sed -i '/^#@NAME:/s/.*/#@NAME: custom/g' path/to/yocto/poky
    /meta-custom/conf/machine/custom.conf
    sed -i '/^UBOOT_EXTLINUX_DEFAULT_LABEL/s/".*"/"Cyclone5
    custom SDMMC"/g' path/to/yocto/poky/meta-custom/conf/machine
    /custom.conf
    sed -i '/^UBOOT_EXTLINUX_MENU_DESCRIPTION_default/s/".*"/"
    Cyclone5 custom SDMMC"/g' path/to/yocto/poky/meta-custom/conf
    /machine/custom.conf
    ```

    b. The KMACHINE variable defines the MACHINE name, which is referenced in the build /conf/local.conf file. Redefine KMACHINE with your own MACHINE name:

    ```
    sed -i '/^KMACHINE/s/".*"/"custom"/g' path/to/yocto/poky/meta-
    custom/conf/machine/custom.conf
    ```

    c. For the linux kernel in the meta-altera layer it's required to define a the preferred provider and version in custom.conf (see Yocto KICKstart#Used source files for more information about the version used). Add the provider and version for linux kernel:

```
echo -e '\nPREFERRED_PROVIDER_virtual/kernel = "linux-altera-
lts"' >> path/to/yocto/poky/meta-custom/conf/machine/custom.
conf
echo -e 'PREFERRED_VERSION_linux-altera-lts = "5.4%"' >> path
/to/yocto/poky/meta-custom/conf/machine/custom.conf
```

d. Set the preferred u-boot version for meta-altera in custom.conf (see Yocto KICKstart#Used source files for more information about the version used):

```
echo -e '\nPREFERRED_VERSION_u-boot-socfpga = "v2021.04%"' >>
path/to/yocto/poky/meta-custom/conf/machine/custom.conf
```

Other variables in the custom.conf file are changed or added at other points in this guide.

Add required parameters to local.conf

Before you start configuring the meta-custom layer, the MACHINE variable in the path/to/yocto/poky/build/conf/local.conf file must be set to the correct machine. The correct machine name is defined in the meta-custom/conf/machine/custom.conf file with the variable KMACHINE.
Remove the default value of the MACHINE variable in local.conf and add the new MACHINE 'custom':

```
sed -i '/^MACHINE/s/MACHINE/#MACHINE/g' path/to/yocto/poky/build/conf
/local.conf
echo -e '\nMACHINE = "custom"' >> path/to/yocto/poky/build/conf/local.conf
```

Create workspace

The following command creates the workspace directory and adds it to bblayers.conf:

```
devtool create-workspace
```

Configure u-boot

Prepare u-boot configuration in workspace

To prepare the u-boot sources for modify the default configuration and create patches, do the following steps:

1. To set the correct default defconfig file for the u-boot, it is required to set the UBOOT_MACHINE variable in the meta-custom/conf/machine/custom.conf file. The defconfig file 'socfpga_cylcone5_defconfig' will be used as default u-boot configuration (see meta-altera/recipes-bsp/u-boot/u-boot-socfpga-common.inc for other Intel fpga defconfig files). If UBOOT_MACHINE has been defined, UBOOT_CONFIG is no longer needded.
Remove the variable UBOOT_CONFIG and add UBOOT_MACHINE to meta-custom/conf/machine/custom.conf:

```
sed -i '/^UBOOT_CONFIG/d' path/to/yocto/poky/meta-custom/conf/machine
/custom.conf
echo -e '\nUBOOT_MACHINE = "socfpga_cyclone5_defconfig"' >> path/to
/yocto/poky/meta-custom/conf/machine/custom.conf
```

2. Run following command to prepare the workspace and fetch and unpack the sources:

```
devtool modify u-boot-socfpga -O
```

Configure u-boot with menuconfig

To make changes to u-boot configuration, follow these steps:

1. Launch menuconfig with following command and make your changes:

```
devtool menuconfig u-boot-socfpga
```

2. Exit and save your configuration. devtool-fragment.cfg will be created automatically.

Create patches for u-boot source files

Do following steps to create patches for u-boot source files:

1. Navigate to u-boot-socfpga in the workspace and add, modify or remove files:

```
cd path/to/yocto/poky/build/workspace/sources/u-boot-socfpga
```

2. Run a build to test your changes:

```
devtool build u-boot-socfpga
```

3. Add the changes made with:

```
git add .
```

4. Commit your changes and add a short message. The message is used to name the patch file:

```
git commit -m "<message>"
```

   If you get the following error message: "*** Please tell me who you are. Run ...", run this commands to define a name and email:

```
git config --local user.email "oe.patch@oe"
git config --local user.name "OpenEmbedded"
```

5. If you want to generate more patches, repeat steps 1 - 4, otherwise navigate back to the poky directory:

```
cd path/to/yocto/poky
```

Create and update u-boot recipe bbappend

Add the devtool-fragment.cfg and the patches to the meta-custom layer with:

```
devtool update-recipe u-boot-socfpga -a path/to/yocto/poky/meta-custom -O
```

Configure linux kernel

Prepare linux kernel configuration in workspace

To prepare the linux kernel sources for modify the default configuration and create patches, do the following steps:

1. Run following command to prepare the workspace and fetch and unpack the sources:

```
devtool modify linux-altera-lts -O
```

Configure linux kernel with menuconfig

To make changes to linux kernel configuration, follow these steps:

1. Launch menuconfig with following command and make your changes:

```
devtool menuconfig linux-altera-lts
```

2. Exit and save your configuration. devtool-fragment.cfg will be created automatically.

Create patches for linux kernel source files

Do following steps to create patches for u-boot source files:

1. Navigate to linux-altera-lts in the workspace and add, modify or remove files:

```
cd path/to/yocto/poky/build/workspace/sources/linux-altera-lts
```

2. Run a build to test your changes:

```
devtool build linux-altera-lts
```

3. Add the changes made with:

```
git add .
```

4. Commit your changes and add a short message. The message is used to name the patch file:

```
git commit -m "<message>"
```

5. If you want to generate more patches, repeat step 1 - 4, otherwise navigate back to the poky directory:

```
cd path/to/yocto/poky
```

Create and update linux kernel recipe bbappend

Add the devtool-fragment.cfg and the patches to the meta-custom layer with:

```
devtool update-recipe linux-altera-lts -a path/to/yocto/poky/meta-custom -O
```

Remove workspace

Remove the workspace from bblayers.conf and delete the workspace folder:

```
cd path/to/yocto/poky/build
bitbake-layers remove-layer workspace
rm -rf path/to/yocto/poky/build/workspace
```

Add custom files to meta layer

Add handoff files to u-boot

If you create a u-boot file for an Intel Cyclone V / Arria V with quartus version 20.1 you have to convert the handoff data into source code.

1. compile your quartus project
2. run the embedded command shell (Intel SoC EDS):

```
~/intelFPGA_lite/20.1/embedded/embedded_command_shell.sh
```

3. navigate to your quartus project directory and convert the handoff data into source code:

```
cd path/to/quartus/project
mkdir -p build
bsp-create-settings --type spl --bsp-dir build \
--preloader-settings-dir hps_isw_handoff/* \
--settings build/settings.bsp
```

Next you have to run the qts_filter to take the sources from the handoff folder, format them and copy them to u-boot source code.

1. Fetch qts_filter.sh from github:

```
wget https://raw.githubusercontent.com/altera-opensource/u-boot-socfpga/socfpga_v2021.04/arch/arm/mach-socfpga/qts-filter.sh
```

2. Run qts_filter for cyclone5 to format and copy the handoff files to custom meta layer:

```
mkdir -p path/to/yocto/poky/meta-custom/recipes-bsp/u-boot/u-boot-socfpga/qts
./qts-filter.sh cyclone5 \
                                path/to/quartus/project \
                                path/to/quartus/project/build \
                                path/to/yocto/poky/meta-custom
/recipes-bsp/u-boot/u-boot-socfpga/qts
```

3. Add the generated files to the u-boot bbappend file:

```
cd  path/to/yocto/poky/meta-custom/recipes-bsp/u-boot
gedit u-boot-socfpga*.bbappend
```

   Add qts_filter files and the following task to copy the qts_filter files to the correct place in u-boot sources during build image (if your are using another board, you have to adjust the file path):

```
SRC_URI += "file://qts/iocsr_config.h \
                        file://qts/pinmux_config.h \
                        file://qts/pll_config.h \
                        file://qts/sdram_config.h \
                        "

do_configure_prepend () {
        install -m 0644 ${WORKDIR}/qts/iocsr_config.h ${S}/board
/altera/cyclone5-socdk/qts
        install -m 0644 ${WORKDIR}/qts/pinmux_config.h ${S}/board
/altera/cyclone5-socdk/qts
        install -m 0644 ${WORKDIR}/qts/pll_config.h ${S}/board/altera
/cyclone5-socdk/qts
        install -m 0644 ${WORKDIR}/qts/sdram_config.h ${S}/board
/altera/cyclone5-socdk/qts
}
```

For more information see u-boot-socfpga/doc/README.socfpga.

Add custom device tree

u-boot

To add an own device tree to u-boot do following steps:

1. Copy the .dts/.dtsi files to meta-custom/recipes-bsp/u-boot/u-boot-socfpga/dts:

```
mkdir -p path/to/yocto/poky/meta-custom/recipes-bsp/u-boot/u-boot-
socfpga/dts
cp custom.dts path/to/yocto/poky/meta-custom/recipes-bsp/u-boot/u-
boot-socfpga/dts/custom.dts
cp custom-u-boot.dtsi path/to/yocto/poky/meta-custom/recipes-bsp/u-
boot/u-boot-socfpga/dts/custom-u-boot.dtsi
```

2. Add the .dts/.dtsi files to the u-boot bbappend file:

```
cd  path/to/yocto/poky/meta-custom/recipes-bsp/u-boot
gedit u-boot-socfpga*.bbappend
```

   Copy the following code to the u-boot bbappend file to copy the files to the correct directory and add them to the correct Makefile:

```
SRC_URI += "[...]
                        file://dts/custom.dts \
                        file://dts/custom-u-boot.dtsi \
                        "

do_configure_prepend () {
        [...]
        install -m 0644 ${WORKDIR}/dts/custom.dts ${S}/arch/arm/dts
        install -m 0644 ${WORKDIR}/dts/custom-u-boot.dtsi ${S}/arch
/arm/dts
}
```

3. Redefine the variable UBOOT_EXTLINUX_FDT_default with the file name of your device tree in the machine configration file of meta-custom:

```
sed -i '/^UBOOT_EXTLINUX_FDT_default/s/".*"/"..\/custom.dtb"/g' path
/to/yocto/poky/meta-custom/conf/machine/custom.conf
```

4. Change in menuconfig the device tree to custom.dts:
   a. Run menuconfig as described here: Configure u-boot (Note: If a devtool-fragment.cfg file for u-boot already exists in the meta-custom layer, you should update it manually, otherwise the existing changes there will be deleted)
   b. Go to *Boot options* *Default fdt file* and change the value to 'custom.dtb'
   c. Go to *Device Tree Control* *Default Device Tree for DT control* and change the value to 'custom'

linux kernel

Adding your own device tree to the linux kernel is similar to the steps above:

1. Copy the .dts file to meta-custom/recipes-kernel/linux/linux-altera-lts/dts:

```
mkdir -p path/to/yocto/poky/meta-custom/recipes-kernel/linux/linux-
altera-lts/dts
cp custom.dts path/to/yocto/poky/meta-custom/recipes-kernel/linux
/linux-altera-lts/dts/custom.dts
```

2. Add the .dts file to the linux kernel bbappend file:

```
cd  path/to/yocto/poky/meta-custom/recipes-kernel/linux
gedit liunx-altera-lts*.bbappend
```

Copy the following code to the linux kernel bbappend file to copy the files to the correct directory during image generation:

```
SRC_URI += "[...]
                        file://dts/custom.dts \
                        "

do_configure_prepend () {
        [...]
        install -m 0644 ${WORKDIR}/dts/custom.dts ${S}/arch/${ARCH}
/boot/dts
}
```

3. Redefine the variable KERNEL_DEVICETREE with the file name of your device tree in the machine configration file of meta-custom.
    a. Open the custom.conf file:

```
gedit path/to/yocto/poky/meta-custom/conf/machine/custom.conf
```

    b. Delete the content of the variable KERNEL_DEVICETREE and add your customized device tree:

```
KERNEL_DEVICETREE = "custom.dtb"
```

(optional) Add fpga configuration file to meta layer

It is possible to add an fpga configuration file to the meta layer, which is used to configure the fpga from u-boot.

1. First generate an rbf file from the sof file generated by the quartus project:
    a. run the nios ii command shell (Intel Quartus):

```
~/intelFPGA_lite/20.1/nios2eds/nios2_command_shell.sh
```

    b. run following command to convert the sof file to rbf:

```
cd path/to/quartus/project/output_files
quartus_cpf -c -o bitstream_compression=on custom.sof
soc_system.rbf
```

2. Run following commands to create a recipe in the specified directory:

```
mkdir -p path/to/yocto/poky/meta-custom/recipes-bsp/rbf
cat > path/to/yocto/poky/meta-custom/recipes-bsp/rbf/custom-rbf.bb
<< EOF
DESCRIPTION = "Add fpga configuration file to image"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://\${COMMON_LICENSE_DIR}/MIT;
md5=0835ade698e0bcf8506ecda2f7b4f302"

PACKAGE_ARCH = "\${MACHINE_ARCH}"
PV = "1.0\${PR}"

FILESEXTRAPATHS_prepend := "\${THISDIR}/files:"
SRC_URI = " file://soc_system.rbf"

do_install () {
        install -d \${D}/images/\${MACHINE}
        install -m 0644 \${WORKDIR}/soc_system.rbf \${D}/images
/\${MACHINE}
        install -d \${DEPLOY_DIR}/images/\${MACHINE}
        cp \${WORKDIR}/soc_system.rbf \${DEPLOY_DIR}/images
/\${MACHINE}
}

FILES_\${PN} += "/images/\${MACHINE}"

EOF
```

3. Create the files folder in meta-custom/recipes-bsp/rbf and copy the rbf file there:

```
mkdir -p path/to/yocto/poky/meta-custom/recipes-bsp/rbf/files
cp path/to/soc_system.rbf path/to/yocto/poky/meta-custom/recipes-bsp
/rbf/files/
```

4. Add with the variable IMAGE_BOOT_FILES the rbf file to the FAT partition of the image in the machine configuration file of meta-custom layer:

```
echo -e '\nIMAGE_BOOT_FILES += "soc_system.rbf"' >> path/to/yocto
/poky/meta-custom/conf/machine/custom.conf
```

5. Modify the image recipe to add the createed recipe to your image. This command refers to the provided image core-image-minimal.bb, you can also create your own image recipe and add your created recipes.

```
echo -e '\nIMAGE_INSTALL_append = " custom-rbf"' >> path/to/yocto
/poky/meta/recipes-core/images/core-image-minimal.bb
```

6. Add the following commands to CONFIG_BOOTCOMMAND via u-boot menuconfig, so that u-boot will configure the fpga with the rbf file:

    a. Run menuconfig as described here: Configure u-boot (Note: If a devtool-fragment.cfg file for u-boot already exists in the meta-custom layer, you should update it manually, otherwise the existing changes there will be deleted)
    b. Go to *Boot options* and enable '*Enable a default value for bootcmd*'
    c. Change the value of '*bootcmd value*' to 'load mmc 0:1 $loadaddr soc_system.rbf; fpga load 0 $loadaddr $filesize; bridge enable; run distro_bootcmd'

References

1. Linux Kernel Development Manual
2. Reference Manual#devtool Quick Reference
3. Board Support Package (BSP) Developers's Guide
4. u-boot-socfpga/doc/README.socfpga
5. Patching the source for a recipe
6. Intel SoC EDS User Guide