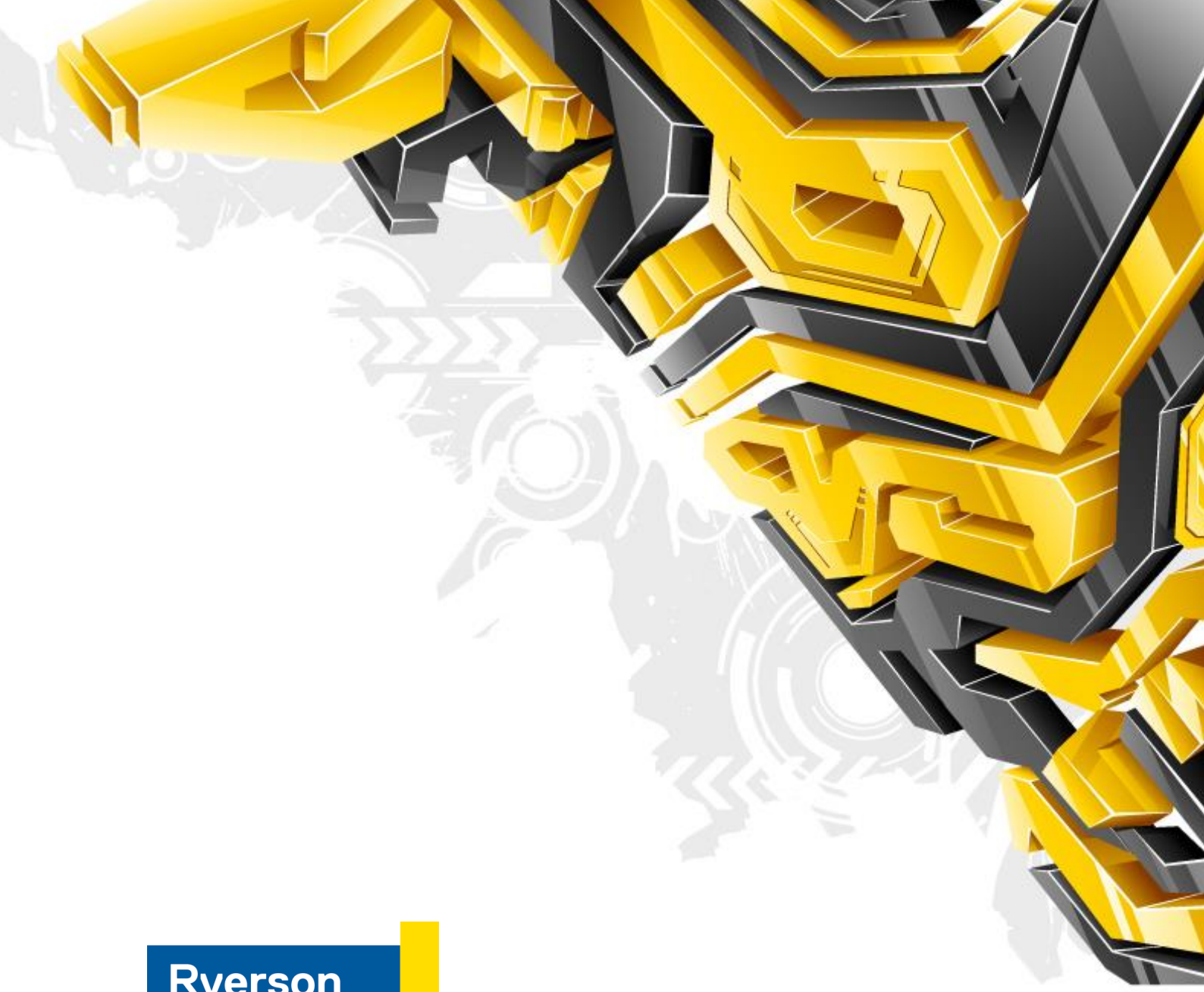


Ryerson University

Interim Report

EE8205 Embedded Computer Systems

Porting Android to DE1 SoC (Altera Cyclone V)

By: *Muhammad Obaidullah*
Supervisor: *Dr. Gul N. Khan*

Introduction

Android has grown to be the number one smartphone operating system. While Android continues to be the most popular Operating System (OS) for smartphones and tablets, it is also being adopted for other embedded devices and industrial equipment. Applications which are soft real-time to hard real-time can take advantage of the GUI rich OS, advanced APIs and Inter-Process Communication (IPC) framework. Since the introduction preemptive kernel mode in Linux version 2.4 and later, Linux OS can now be used for hard real-time applications.

There are a number of free software libraries provided with Android and taking advantage of these will reduce the development cost of the application. Provided libraries include Web browser, File System, WebKit, SQLite, OpenGL/ES, Graph/chart, and Network Stack. Until now, powerful embedded GUIs were rarely available and very expensive. GUI for Android can be designed easily using android XML layout files and tuned according to customer's needs. GUI by default is compatible with multi-touch, giving the familiar easy-to-use feeling of a smartphone operating system.

Why Android on DE1 SoC?

Porting Android to Cyclone V will allow much higher abstraction level than working with plain linux C/C++ native code. Applications within the OS can take advantage of FPGA acceleration and the well-implemented and well-tested Android software stack. Porting android to DE1 SoC allows use of over 2.4 million applications currently available in the Google Play store. It also opens new boundaries for software developers to explore hardware accelerations for compute intensive applications. It has been found that real-time operating systems designed for Symmetric Multi-Processing (SMP) will generally provide similar or better performance and lower latency than bare-metal applications (no operating system). Besides having an FPGA fabric within the Cyclone V, it contains dual ARMv7 cores within the HPS which are well-capable of running Android.

From research point of view, it is beneficial to explore the advantages and drawbacks of an embedded system which makes use of hardware acceleration rather than increased CPU clocks. Use of such hybrid system may reduce overall system power consumptions.

Linux Kernel

Linux is an operating system found in a wide variety of computing devices including personal computers, servers, smartphones and other embedded systems. Among other features, most important feature provided by any operating system is multitasking. This is so that the hardware resources (processing unit, memory, etc.) available on the chip can be shared among requesting tasks and allow the processor to give illusion of performing multiple tasks at once (concurrency).

Although sometimes used interchangeably, there is a difference between a kernel and an operating system. The kernel is part of operating system. It is responsible for memory management, network management, device driver, file management, and process management. An operating system is composed of the kernel and applications (i.e. compiler, text editor, window manager, GUI) which enables users to perform useful tasks. In other words, the kernel is the 'brain' of the operating system.

Any task which is to be performed by the kernel has higher priority than the task performed by the user. Therefore, modern CPUs have two modes in which a task can be run, kernel mode (system mode) or user mode. In kernel mode, the CPU executes instructions which are related to kernel (kernel code) and provides unlimited access to the code. Any instruction can be executed in kernel mode and any memory address can be referenced. All other codes are executed in user mode where some CPU instructions cannot be directly initiated and some

memory locations are off-limits. Hence, if the user code wants to execute some instructions which are not available in user mode, it has to make a '*system call*' in order to perform privileged instructions. Such privileged instructions include process creation and input/output operations.

In non-preemptive type kernels, while a process is in kernel mode, it cannot be arbitrarily suspended and replaced by another process (i.e., preempted) for the duration of its time slice (i.e., allocated interval of time in the CPU), in contrast to user mode, except when it voluntarily relinquishes control of the CPU. Although the Linux kernel was of non-preemptive nature, which means that the kernel mode tasks were not interruptible, the version 2.4 and later are preemptive. This makes Linux kernel strong candidate for embedded devices and control applications where timing and interrupt handling are crucial. Processes in kernel mode can be interrupted by an interrupt or an exception.

Android

Contrary to popular belief, Android is not an operating system. Android is an open source software stack for a wide range of mobile and embedded devices and corresponding open source project led by Google. It is emerging as developer's choice in order to program, develop, and design or IoT, mobile computing, and other embedded devices because of its highly abstracted and modularized components which eases software development. Android provides the freedom to the developer to implement own device specifications and drivers. The hardware Abstraction Layer (HAL) provides a standard method for creating hooks between the Android platform stack and custom hardware.

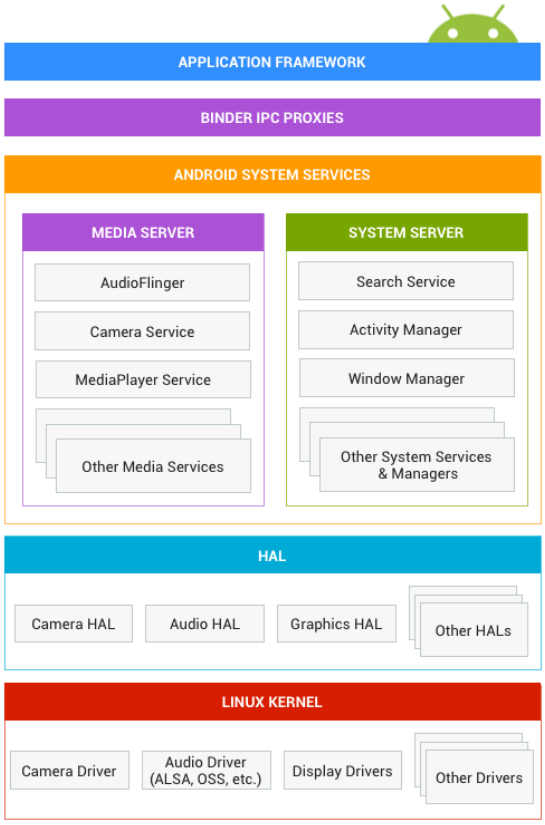


Figure 1: Android System Architecture.

Application Framework

Application framework provides software developers with APIs to use in order to develop android applications. This is the final layer on which the app runs. Some developer APIs map directly to underlying HAL interfaces too.

Binder IPC

Binder Inter-Process Communication (IPC) is a mechanism which facilitates inter-processes communications. It allows application framework to cross boundaries and call into the Android system services code. This allows high-level APIs to interact with Android systems services. Normally, under Linux environment, the processes communicate by using named FIFOs, signals, sockets, semaphores, message queues, or shared memories. A higher abstraction is done of this inter-process communication technique and provided as a feature in Android stack. It is a lightweight RPC (Remote Procedure Communication). One Android process can call a routine in another Android process, using binder to identify the method to invoke and pass the arguments between processes.

System Services

System services lie in the middle of HAL and application framework and allow APIs to communicate with the underlying hardware. They are focused and modular components which provide focused helping hand to the software developer. For example Notification Manager manages all notifications and can be asked by any process to view/edit/delete notifications. Similarly, Camera Service allows processes to view camera stream, trigger image capture etc.

Hardware Abstraction Layer (HAL)

The hardware abstraction layer defines a standard interface for hardware vendors to implement and allows Android to be agnostic about lower-level driver implementations. HAL implementations are packaged into modules (.so file) and loaded by Android system at appropriate time (dynamic linking). Each specific hardware added into the custom system needs to have a corresponding HAL and driver. Android does not specify a standard interaction between HAL implementation and device drivers, so developers are free to do what is best for the situation. However, in order for Android to correctly interact with custom hardware, contract defined in each hardware-specific HAL interface should be followed.

Each hardware-specific HAL interface has properties that are defined in source code provided by Google. This guarantees that HALs have a predictable structure. This interface allows the Android system to load the correct versions of custom HAL modules in a consistent way. There are two general components that a HAL interface consists of: a module and a device.

A module represents a custom packaged HAL implementation, which is stored as a shared library (.so file). It contains metadata such as the version, name, and author of the module, which helps Android find and load it correctly. The hardware.h header file in the source code defines a struct, hw_module_t, that represents a module and contains information such as the module version, author, and name.

In addition, the hw_module_t struct contains a pointer to another struct, hw_module_methods_t, that contains a pointer to an "open" function for the module. This open function is used to initiate communication with the hardware that the HAL is serving as an abstraction for. Each hardware-specific HAL usually extends the generic hw_module_t struct with additional information for that specific piece of hardware.

Linux Kernel

Developing custom device drivers is similar to developing a typical Linux device driver. Android uses a version of the Linux kernel with a few special additions such as wake locks (a memory management system that is more aggressive in preserving memory), the Binder IPC driver, and other features important for a mobile embedded platform. These additions are primarily for system functionality and do not affect driver development.

Developer can use any version of the kernel as long as it supports the required features (such as the binder driver). However, it is recommended to use the latest version of the Android kernel provided by Google.

Literature Review

What has been done?

So far three entities have managed to port Android successfully to Altera Cyclone V SoC.

Entity Name	Missing Features	OS Image Available	Source Code Available	Link
MRA Digital [1]	-	No	No	https://youtu.be/zHqS_yWiMNI
Fujisoft [2]	-	No	No	https://www.fsi-embedded.jp/e/_emb/gaforandroid_e/
University of Toronto [3]	Audio CODEC	Yes	No	https://rocketboards.org/foswiki/view/Projects/AndroidForDE1SoCBoard

One implementation is from University of Toronto's 4th year capstone project at Department of Electrical & Computer Engineering, with group members Kevin Nam, David Xie, and Steven Nesmith, under supervision of Prof. Stephen Brown.

An Android app called "DE1SOC" comes preinstalled in the image. This Android app was created to demonstrate Android app to FPGA communication. Shown in Figure 2, the app has an interface that allows users to toggle the LEDs on the board, as well as read the value of the switches. These LEDs and Switches are connected to the FPGA's I/O pins, which are in turn connected to PIO IP cores that have been instantiated in the FPGA. These cores provide memory-mapped register interfaces which are made available to the ARM processor through the lightweight HPS-to-FPGA bridge. The Linux kernel's GPIO drivers are used to expose the pins as GPIO devices (in /dev/), providing a file-based I/O mechanism. The Android app is then able to read and write these files to read and write the values of the LEDs and switches. The FPGA is programmed automatically as part of the boot sequence.

Users can interact with the GUI by using the touch screen of the Terasic MTL. If an MTL is not available, users can use a USB mouse, connected to one of the two USB ports. When a mouse is connected, a mouse cursor will appear on the screen. Users can provide internet connectivity by plugging in an ethernet cable. Support for the board's audio CODEC has not been implemented at this time. They used Chris Rauer's modified Linux kernel with added support for the Altera frame buffer.

Figure 2: Android running on DE1 SoC Altera Cyclone V by University of Toronto.

Porting

Building Altera Linux OS

Altera provides Linux BSP support for the Cyclone V SoC FPGA Development Kit, and provides the following:

- Linux kernel 3.7
- Preloader
- Uboot version 2012.10
- Yocto version 'Danny'
- The packages for the root file system
- The tool chain (Linaro-GCC, v4.7)

Yocto is used to build the sources of the kernel, the u-boot and the root file system. There are many source code packages available under the Yocto project. The Linux kernel source and binaries can be downloaded from Altera(http://software.altera.com/linux_socfpga). In addition to this, Git also has source code for modified linux for Altera SoCs at: https://github.com/altera-opensource/linux-socfpga

Host Setup

I am using 64 bit Ubuntu MATE (16.04.1 LTS) as a host machine to compile the Altera Linux OS. There are few packages required to be installed before building the Linux OS image for Altera Cyclone V FPGA.

1. First we need to update the package manager using the following command:

```
$ sudo apt-get update
```

2. Update the package manager using the following command:

```
$ sudo apt-get uprade
```

3. Install the required packages using the following command:

```
$ sudo apt-get install sed wget cvs subversion git-core coreutils unzip texi2html texinfo libsdl1.2-dev docbook-utils gawk python-pysqlite2 diffstat help2man make gcc build-essential g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff libtool xterm
```

4. Install software to make bootable image using the command given below:

```
$ sudo apt-get install uboot-mkimage
```

5. Also, since the host machine runs 64 bit version of OS, we also need to install:

```
$ sudo apt-get install ia32-libs
```

Yocto Setup

The source code package (linux-socfpga-13.02-src.bsx) was downloaded from Altera's website and the following steps were taken.

1. The default install location is /opt/altera-linux. To install the package, enter the following command:

```
$ sudo linux-socfpga-13.02-src.bsx /opt/altera-linux
```

2. Update the package manager using the following command:

```
$ sudo /opt/altera-linux/bin/install_altera_socfpga_src.sh ~/yocto- 13.02
```

3. Run the script provided by Altera to set up the build variables needed to compile the Linux kernel:

```
$ sudo source altera-init ~/yocto-13.02/build
```

4. Build u-boot:

```
$ sudo bitbake u-boot
```

5. Build Linux kernel:

```
$ sudo bitbake linux-altera
```

6. Build root filesystem:

```
$ sudo bitbake altera-image
```

7. The images are generated in ~/build/tmp/deploy/images

Testing the Linux on board

To boot the linux images on SoC FPGA development kit, we can write the images we just built with Yocto into one of the three Flash devices: SDMMC, NAND and QSPI. We will use SDMMC due to its easy detachability. For SDMMC boot, all boot images will be located inside SDMMC card. A script is provided with the release that will create an SD card image, ready to be deployed.

The provided tool, named make_sdimage.sh, will create a 2GB SD card image, with three partitions:

Partition 1 (FAT)	**Partition 2 (ext3)**	**Partition 3 (NONAME)**
Linux kernel and device tree	The Linux root file system	• Partition used by the SoCFPGA to load the preloader. • Also contains u-boot image

Following command is entered to make the image:

```
$ sudo   make_sdimage.sh \
         -k uImage,socfpga.dtb \
         -p u-boot-spl-socfpga_cyclone5.bin \
         -b u-boot-socfpga_cyclone5.img \
         -r fs \
         -o sd_image.bin
```

Where:

- -k accepts a comma separated list of files. Here, we show the kernel and the device tree blob.
- -p the preloader raw binary, as generated by Yocto or the U-Boot Makefile
- -b the bootloader image, as generated by Yocto or the U-Boot Makefile
- -r the directory where the file system is located
- -o the image name

Building Android from Source

Android Open Source Project (AOSP) has several branches, repositories (repos) and contributors. This is the reason they use a script called Repo to manage all git repositories in context of Android. [4]

Installing Repo

1. Create a directory called bin in home folder by entering the following command:

```
$ sudo mkdir ~/bin
```

2. Add the newly created directory to PATH using the following command:

```
$ sudo PATH=~/bin:$PATH
```

3. Download curl. Curl downloads files from servers using the http link provided. This is done by using the following command:

```
$ sudo apt-get install libcurl3 php5-curl
```

4. Download the Repo tool:

```
$ sudo curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

5. Change the downloaded script to executable:

```
$ sudo chmod a+x ~/bin/repo
```

Downloading Source

1. Create an empty directory called WORKING_DIRECTORY in home folder by entering the following command:

```
$ sudo mkdir WORKING_DIRECTORY
```

2. Install git (version control tool) by using the following command:

```
$ sudo apt-get install git-all
```

3. Configure git by putting user name and email:

```
$ git config --global user.name "Muhammad Obaidullah"
$ git config --global user.email "mobaidullah@ryerson.ca"
```

4. Initialize repo in the WORKING_DIRECTORY by:

```
$ sudo repo init -u https://android.googlesource.com/platform/manifest
```

5. Checkout, sync, and download the source from Google's repository by using the following command:

```
$ sudo repo sync
```

Building Android for Altera Cyclone V SoCFPGA

This is the next step which involves placing the already build kernel into the working directory of the android source code and then building the Android OS image.

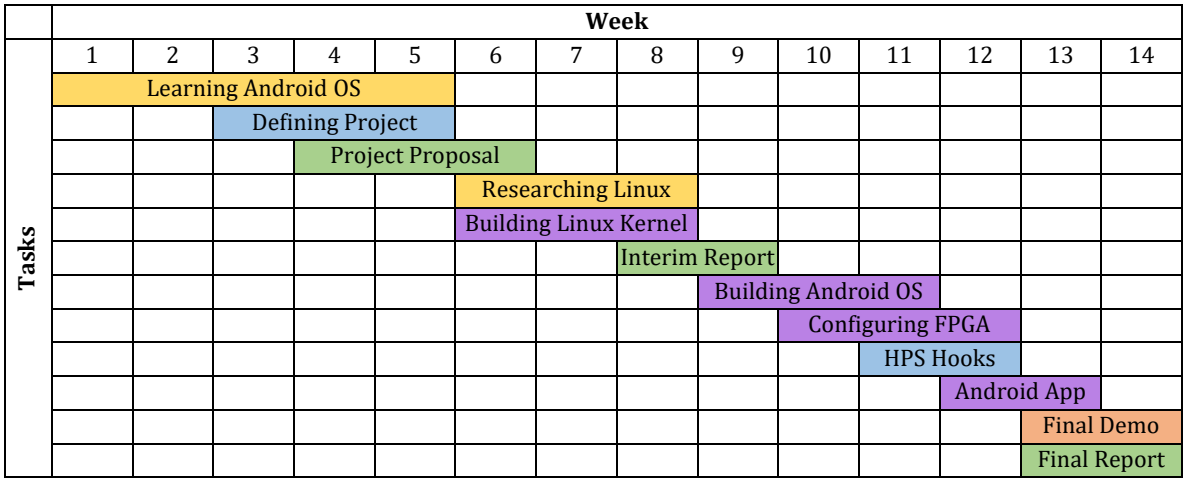
Expected Final SD Card Image Structure

Partition 1 (IMAGE)	**Partition 2 (ROOTFS)**	**Partition 3 (NONAME)**	**Partition 4 (DATA)**
Linux kernel	Android OS APKs eg. apps from Google Play	Uboot	Flexible Space Pictures (.jpg, .png)

Project Progress

Timeline

Gantt Chart



Legend:

- Study or Research
- Submission
- Writing or Design
- Coding
- Demo

Conclusion

Dynamic App-based Partial Reconfiguration

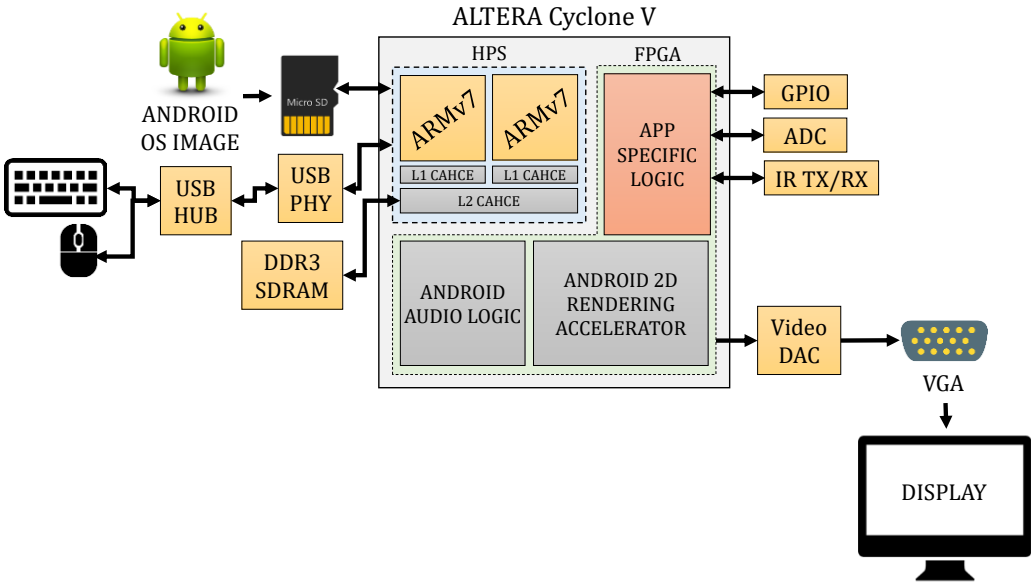


Figure 3: Hardware System for Android OS on Altera Cyclone V.

Porting android stack to a HPS-FPGA system allows apps to take advantage of FPGA fabric and perform partial reconfiguration to build and tear-down application specific accelerators on the remaining FPGA fabric. Since Cyclone V is not designed to run Android by default, graphic rendering engine and audio logic needs to be implemented in the FPGA. The left-over Les can be used to implement any app-specific logic accelerator.

Since, audio and video logic should not be touched while the Android OS is running, partial reconfiguration of the FPGA fabric needs to be done. Cyclone V can be dynamically partially reconfigured using Partial Masked SRAM Object File (.pmsf) and Raw Binary File for Partial Reconfiguration (.rbf). Each app can load its own .rbf file from SD card onto FPGA. However, there should be certain checks in place in order to catch and avoid short-circuits and other common I/O errors used by loaded logic and logic to be loaded.

References

[1] Intel, "Implementation of an Android™ Operating System on an Altera SoC," Intel, 8 January 2014. [Online]. Available: https://youtu.be/zHqS_yWiMNI. [Accessed 9 November 2016].

[2] "Graphics Accelerator for Android," FUJISOFT INCORPORATED, 15 November 2013. [Online]. Available: https://www.fsi-embedded.jp/e/_emb/gaforandroid_e/. [Accessed 9 November 2016].

[3] M. Daum, "Android for DE1-SoC Board," RocketBoards.org, 6 October 2016. [Online]. Available: https://rocketboards.org/foswiki/view/Projects/AndroidForDE1SoCBoard. [Accessed 9 November 2016].

[4] Google Inc., "Android Open Source Project," Google, 23 August 2016. [Online]. Available: https://source.android.com. [Accessed 9 November 2016].