Ryerson University

Project Proposal

EE8205 Embedded Computer Systems

Porting Android to DE1 SoC (Altera Cyclone V)

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

Android

A Linux-based Software Stack

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.

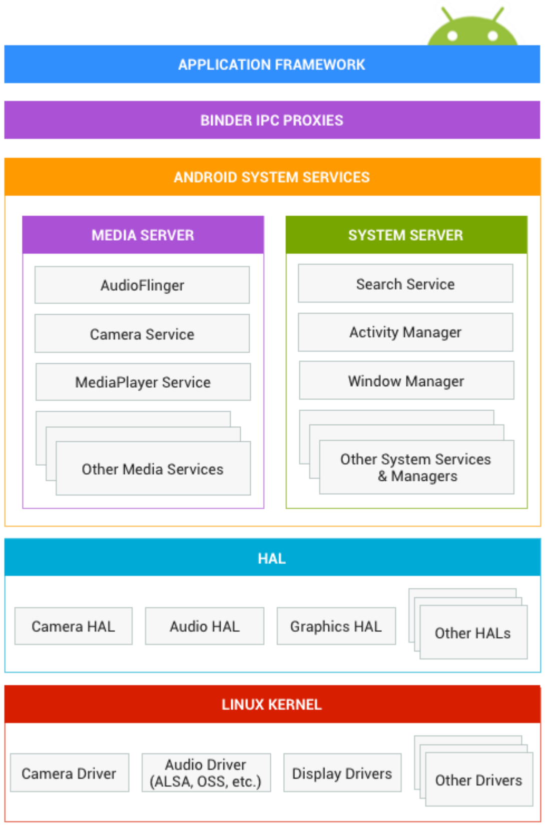

Figure 1: Android System Architecture.

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.

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.

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	-	No	No	https://youtu.be/zHqS_yWiMNI
Fujisoft	-	No	No	http://www.fsi.co.jp/solution/android/e/solution/
University of Toronto	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.

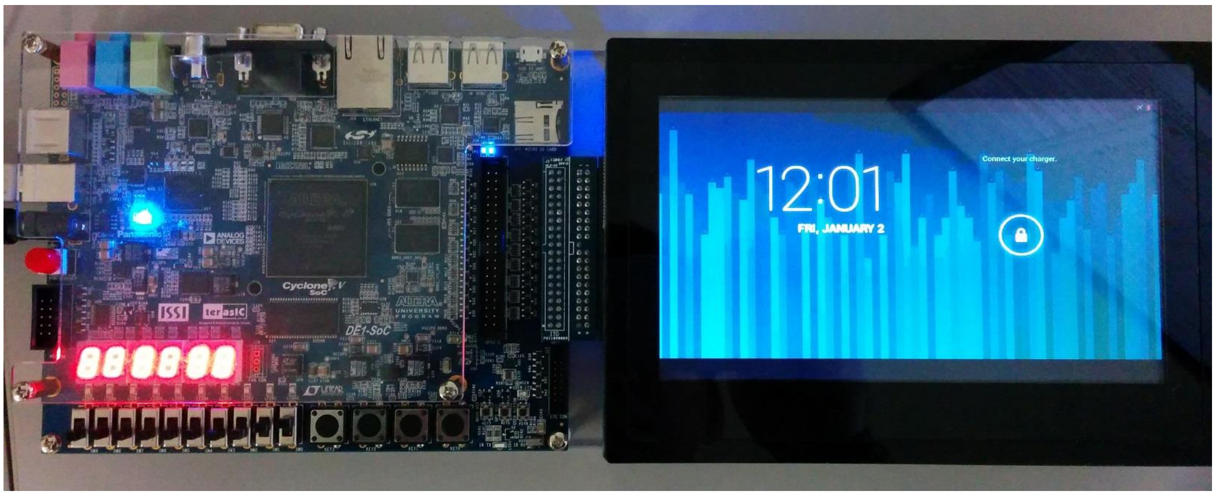

Figure 2: Android running on DE1 SoC Altera Cyclone V by University of Toronto.

Example Application

Dynamic App-based Partial Reconfiguration

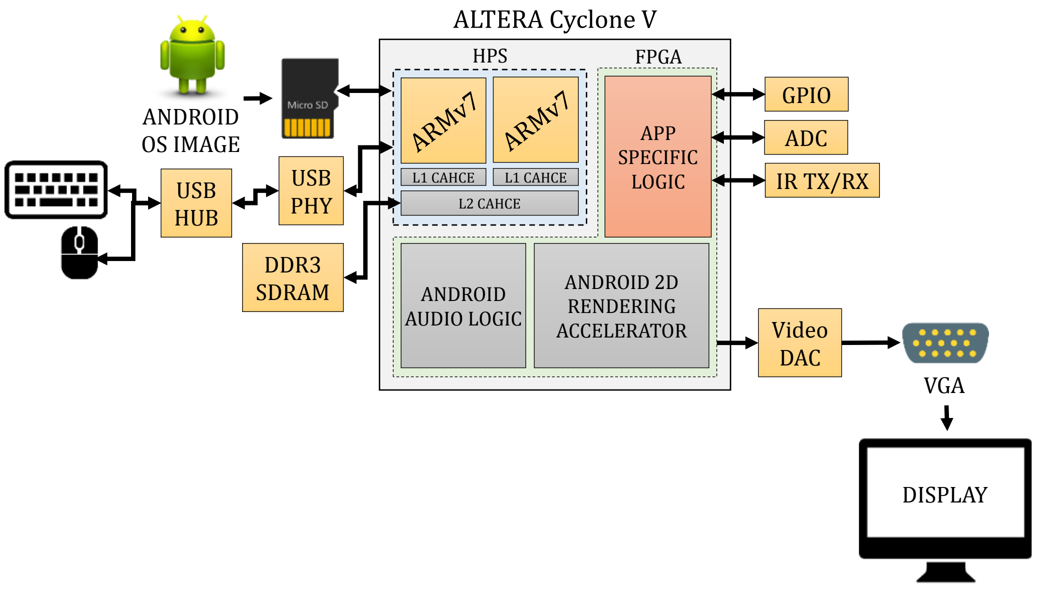

Figure 3: Hardware System for Android OS on Altera Cyclone V.

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.