cadence

SD Host Firmware Driver Quick Start Guide

Product Version 5.7.1

April 2020

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

1. Overview

This document describes the basics of how to get up and running with the Cadence IP Core Driver.

- Prepare the Cadence Platform Services (CPS)
- Prepare interrupt handling
- Build the core-driver
- Use the core-driver to initialize the Cadence IP

2. Details

Here's how to get started:

1. *Prepare the Cadence Platform Services:*

   This is covered in detail in the Cadence Driver Porting Guide, which is available at doc/porting/porting_guide.pdf. A sample implementation for bare-metal systems is available in RefCode/cps_bm.c. You may be able to use these without modification.

   CPS is a platform-specific code to connect a given Cadence driver to the rest of your system. Not all of the features of CPS are required for this driver. You only need to implement the functions for uncached access to memory:

   ```
   CPS_UncachedRead8, CPS_UncachedRead16, CPS_UncachedRead32,
   CPS_UncachedWrite8, CPS_UncachedWrite16 and CPS_UncachedWrite32
   ```

   These functions will be used by the core driver to access the memory in a portable manner. CPS_CacheFlush() / CPS_CacheInvalidate() are not used by this driver. However, this means that you must ensure cache consistency of any areas of memory passed to the driver for use in DMA transfers.

   After this step you should have an object file implementing the CPS for your target system which you can link with later.

2. *Prepare interrupt handling*
   Driver can work in two modes interrupt mode and polling mode.

   To work in interrupt mode user needs to call *start* function. In this mode interrupts are used to handle controller requests. If function is not called then driver works in polling mode.

3. *Build the core-driver*

   All driver sources are provided in the CoreDriver directory. You may need to modify environemnt_config.h file to ajust driver to your platform.

4. *Use the core-driver to initialize the Cadence IP*

   There are two samples available in RefCode folder. Each of them executes the same initiaization steps.

   The sample code is designed to require only minor changes before it can be used. You will however need to change some default values to match your hardware. Change these

```
/* This should be the base address of
   the Cadence IP core on your system */
#define SD_REG_BASE            0x50000000
```

   The sample code then calls *probe* to confirm that the hardware is available at the specified address, and to determine the size of memory which is required for private data and dma descriptors. This memory must be provided by the code calling the driver, and may be allocated at compile time (e.g. by an array allocation as shown in the sample code) or dynamically. The private data and addresses(virtual and logical) will be passed to the driver as a parameter for each call to the API.

   The sample code then calls *init* to initialize the driver.

   As well as initializing the driver, the sample code also displays some diagnostic information using stdio. If stdio is not available on your system, these calls should be stubbed out or replaced with an alternative (e.g. write to memory or local storage.)

   After modifying the sample code to suit your environment, you should compile for your target system and link with the driver and CPS objects prepared previously, then download and run the code on your target system. This will be hardware and operating system specific, and is not covered in this documentation.