OpenCPI
Matchstiq-Z1 BSP Case Study

Version 1.5

Revision History

Revision	Description of Change	Date
v1.1	Initial Release	3/2017
v1.2	Updated for OpenCPI Release 1.2	8/2017
v1.3	Updated for OpenCPI Release 1.3	1/2018
v1.4	Changed formatting, added TOC and References, added missing device workers, added details	9/2018
v1.5	Updated for OpenCPI Release 1.5	4/2019

Table of Contents

1 References

The reference(s) in Table 1 can be used as an overview of OpenCPI and may prove useful.

Title	Published By	Link
Platform Development Guide	OpenCPI	`http://opencpi.github.io/\OpenCPI_Platform_Development.pdf`

Table 1: References

2 Matchstiq-Z1 BSP Case Study

The OpenCPI Platform Development Guide1 provides the high-level steps for developing an OpenCPI board support package or BSP. The first step for developing an OpenCPI BSP was to breakdown the Matchstiq-Z1 platform into it functional components and compare those to what was already supported by the framework, partially supported by the framework or not supported. Fortunately, OpenCPI BSP for the Zedboard platform used the same FPGA, which allowed much of the platform worker to be reused for the Matchstiq-Z1. Additionally, because the Petalinux OS was to be used on the Matchstiq-Z1, the effort for integrating the cross-compile tools was already supported by the framework. The remaining tasks were primarily focused on the device workers and proxies that were either new to the framework, required modifications to support or simply unique to the Matchstiq-Z1.

Figure 1 shows the block diagram for the Matchstiq-Z1's OpenCPI Container of the FPGA. It indicates the organization of the various blocks within the FPGA.

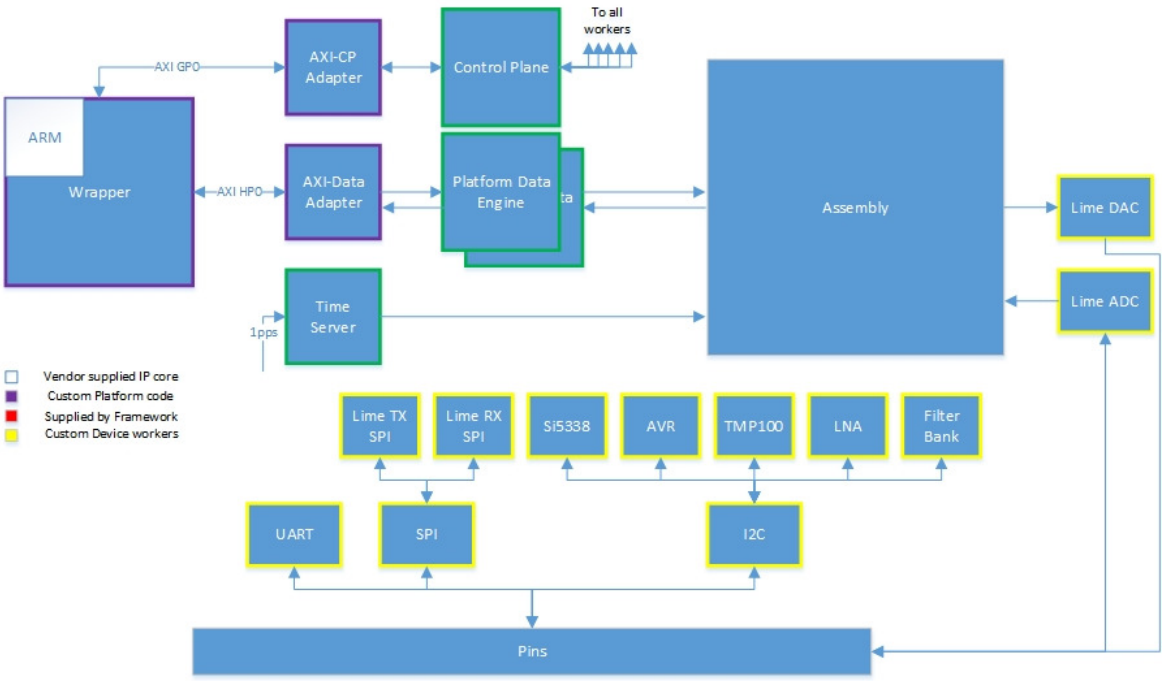

Figure 1: Top Level Block Diagram

The block diagram in Figure 2 shows the connectivity between device and proxy workers in support of the Matchstiq-Z1's OpenCPI BSP.

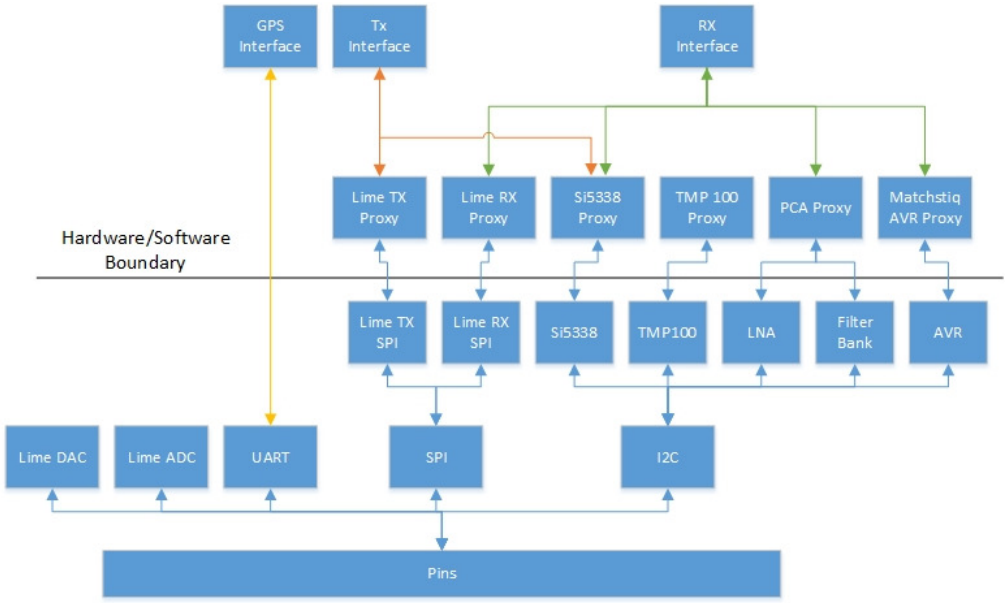

Figure 2: Workers Block Diagram

3 Petalinux

NOTE: The method of forcing an OpenCPI supported Linux OS and compiler onto a platform is no longer the recommended method for enabling a new GPP platform. The new recommended method is to enable OpenCPI for the existing platform's OS by integrating the cross compiler tools supplied by the vendor, when applicable.

The first task that needs to be completed is to install the OpenCPI version of Petalinux onto the platform. In order to boot the platform on the Matchstiq-Z1, there are three things that are needed on the SD card: Device tree, Linux kernel, and the root filesystem. The First Stage Boot Loader for the Matchstiq is located in flash and was left alone for our implementation.

There are environment variables that U-Boot (second stage boot loader) use to load files from the SD card into memory to boot the processor. The original Matchstiq-Z1 SD card image had created a kernel with a small root file system embedded within the kernel. This used different U-boot settings and we had to edit these settings to boot our version. On the Matchstiq-Z1, these U-Boot variables are located on the second partition of the SD card. This is important to keep in mind when trying to create new SD cards for the Matchstiq-Z1.

To enable the MatchStiq-Z1 to support the OpenCPI *Network Mode* development environoment, an Ethernet connection was required. While the Matchstiq-Z1 does not have an Ethernet port, a USB to Ethernet adapter and a modification to the Petalinux kernel were all that was needed to enable Ethernet for this platform. It is worth noting that the Zedboard device tree/build was also modified to support USB on-the-go.

4 Cross Compiler

When the Matchstiq-Z1 was being brought into the framework, the Zedboard platform work had already integrated the cross compiler for Zynq-7000 into OpenCPI. Had this not been the case there would need to be work done to integrate a cross compiler into the framework.

5 Device Workers

The platform has a set of devices that are connected to the FPGA. The devices were analyzed for functionality and each function was implemented as a device worker. As is the case with component reuse, several of the device workers were already available (lime_spi, lime_tx, lime_rx, lime_dac, lime_adc), since they had been previously developed in support of the ZedBoard/Zipper SDR. However, some of these device workers did require expanding their support for a different clock distribution circuit compared to that of the Zedboard/Zipper SDR.

5.1 I2C Bus

There are several devices connected to the I2C bus, the peripherals and their slave addresses were known from the Epiq documentation. The order and the values of the register settings were determined by monitoring the I2C bus with a bus analyzer while using Epiq's provided API.

The I2C connected devices are: SI5338 clock generator, TMP100 temperature sensor, AVR microcontroller, a low noise amplifier, and a RF filter bank. A device worker and a proxy worker was developed for each of these peripherals.

5.2 SPI Bus

There is a single slave SPI connection on the Matchstiq-Z1 platform connecting to the LMS6002D transceiver for control to the configuration registers of the transceiver. A device worker and a device proxy needed to be implemented for this device, this was reused from the Zedboard platform.

5.3 DAC/ADC

There are 12 bit D/A and A/D data lines to move the I/Q samples between the LMS6002D and FPGA. Device workers that were developed for the Zedboard/Zipper SDR were reused, with modifications to the clock distribution.

5.4 TX/RX

There are TX and RX enable pins and registers to enable/disable the RF portions of the transceiver. Device workers and proxies that were developed for the Zedboard/Zipper SDR were reused but only supported access to the tx/rx enable/disable pins of the transceiver, where as the Matchstiq-Z1 does not provide direct access to those pins. Therefore, these device workers and proxies were modified to support enable/disable of the transceiver's RF transmitter via a register.

5.5 GPS UART

There is a GPS receiver on the Matchstiq-Z1 that communicates with the FPGA over UART. A device worker and a device proxy (TBD) needed to be implemented for this device.

6 HDL Platform

The device workers that are described above are instantiated in the container. The ARM processing system needed to be instantiated into the platform and configured in the correct way. This was a non trivial task to learn how to do but was reused from the Zedboard platform.

6.1 Control Plane Integration

The control plane modules are connected to General Purpose AXI port 0 on the ARM processor. An adapter needed to be written to take data being streamed to the FPGA via AXI to be passed to the HDL framework. Driver software also needed to be written for the processor to move data from the ARM processor down to the FPGA. This driver was reused from the Zedboard platform.

6.2 Data Plane Integration

The data plane modules are connected to a High performance AXI port on the ARM processor. An adapter needed to be written to take data being streamed to the FPGA via AXI to be passed to the HDL framework. Driver software also needed to be written for the processor to move data from the ARM processor down to the FPGA. This driver was reused from the Zedboard platform.

7 Common interfaces

Each radio platform will have a common generic software interface for receive and transmit. These interfaces are documented in the workers data sheets. The workers for each of these interfaces needed to be implemented on the Matchstiq-Z1 platform.