Gpsctrl and Board Bring-up Update

Rob Halliday
CWRU

January 21, 2015

1 Board Bring-up

Requirements: We will need a version of Linux to run on the UUB, which will have to be configured specifically for our hardware.

The PetaLinux flavor is advocated by Xilinx and so we plan to use it. We have been able to bring up Ubuntu on the ZedBoard (a Zynq evaulation board) as an exercise and can run the Xilinx supplied version of Petalinux. Once we have a working UUB, we will be able to do the board bring-up and create a distributable installation procedure. In particular we need a device tree for the UUB to procede here.

The process consists of making 4 "main ingredients"

1. Bootstrapper: Initializes the Zynq and is hardware specific but can be made trivially from a board support package.

2. First stage bootloader: Provides resources for second stage bootloader. Is made for Zynq chips in general, not for a specific board.

3. Second stage bootloader: The most commonly used solution here is a program called u-boot, which is available preconfiured for Zynq. Although we would need to create a make option for the UUB once completed, it is fairly generic for Zynq based systems.

4. Operating system: Here we need an operating system, like PetaLinux or Ubuntu. It will be configured using a device tree which is available for the ZedBoard from Xilinx, and can likely be modified to accomodate the hardware of the UUB. It will be cross compiled on an external linux platform using the provided Xilinx toolchain (Code Sourcery).

2 GPS HW and SW

2.1 Serial Communications

Requirements: A UART may be required for serial communications with the GPS.

The NS16550 UART IP provided by Xilinx accomplishes the task of communicating over serial ports on the ZedBoard and so, if needed, it can provide a link betweeen the GPS receiver and the UUB. It is simply implemented in the Vivado software by adding the IP and then it can be set as standard IO for a standalone Zynq configuration, or can be recognized in Linux as a standard serial port, as it would be on the UUB. Once it is recognized in Linux, standard approaches are available to handle interrupts and messaging.

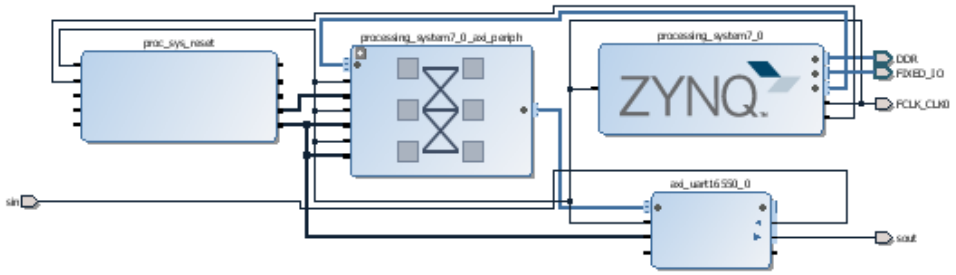

Figure 1: Basic block diagram with the minimum hardware to facilitate UART communications.

2.2 Gpsctrl

Requirements: In switching to the Encore M12M GPS receivers, some of the commands for initializing and communicating with the GPS must be updated, namely in gpscommandes.h and some of the arguments in gpsmain.c and gpsinit.c.

To accomplish this we have added "#ifdef" statements with the modified commands in the appropriate files. This will allow all of the other aspects of the program to remain the same. Rewriting the commands is well underway, and a testing scheme is being devised. The vast majority of the required commands are the same between the two receivers, except for those pertaining to TRAIM and visible satellites, since we now have more available channels.

3 CSAC Tests

Note: this is not directly related to SD upgrade.
Objective: To utilize an atomic clock, ZedBoard and M12M GPS receiver setup to determine the relative jitter between two M12M receivers as a function of distance and possibly of relative weather conditions.

Using the ZedBoard, we can create a testing platform similiar to the previous calibration platform, which is mobile and is trained off of an atomic clock. This will allow us to take the receivers away from each other and watch their relative jitter for as long as the atomic clocks can stay accurate without recalibration. So far we have configured the ZedBoard to run at up to 600Mhz on an internal oscillator, which would allow us to measure down to the stated accuracy of the M12Ms. When we have a fully configured setup we will test the accuracy of the new receivers and once we have two we will be able to begin field testing. Once complete, we should have an accurate idea of exactly how bad the drift between two GPS receivers in the field is.

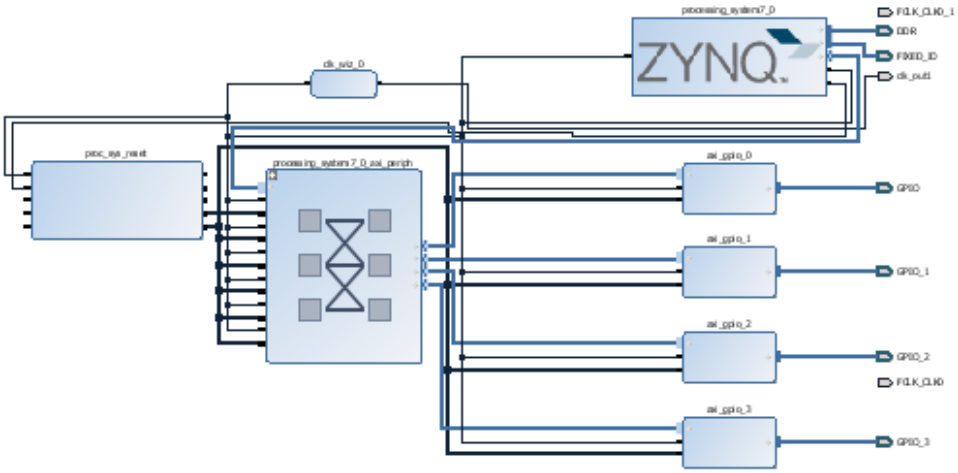

Figure 2: Block diagram based on an early version of Bob Sobin's time tagging system. Utilizes GPIOs to communicate with an unshown second block of verilog code which actually contains the counter, now working up to 600Mhz.