Update of the register size used in the NeXtRad Synchronization Controller

Dominic Manthoko

March 26, 2017

1 Introduction

1.1 Subject and Motivation of Report

The author was requested to make modifications to the NeXtRad Synchronization Controller developed by Johann Burger. In particular, the register size of the Synchronization Controller needed to be increased from 16 bits to 32 bits.

1.2 Background of the NeXtRad Synchronization Controller

1.3 Methodology

2 The Synchronization Controller

2.1 Modifications Made to the Original Synchronization Controller

It is highly recommended that you read the document about the Timing Control Unit(TCU). This document specifies how to set up the synchronization controller, the commands used to operate it and so forth. In this section, only the parts that were modified will be mentioned.

Parameter	Size	Description
MB offset	16 bits	Main Bang offset in 10ns intervals
DIG offset	16 bits	Digitisation offset in 10ns intervals
Next PRI (upper 2 bytes)	16 bits	Next PRI offset in 10ns intervals
Freq	16 bits	Frequency of operation
Pol Mode	3 btis	From table 1.2
Next PRI (lower 2 bytes)	16 bits	Next PRI offset in 10ns intervals

Table 1: Table of all the parameters necessary for each pulse

2.2 Setting up the Synchronization Controller

All the available registers can be accessed in the proc directory:

```
/proc/<PID>/hw/ioreg/
```

where <PID> must be replaced by the process ID of the running firmware. This can be identified by running the 'ps' command in the terminal.

Registers can be written to by using the echo command. The following command writes a 1 to the n register:

```
echo -e -n "\x01\x00" > /proc/<PID>/hw/ioreg/n
```

The registers can then be read back by using the 'od' command:

```
od -x /proc/<PID>/hw/ioreg/n
```

With the modified TCU, reading and writing from the n, m and reg_led registers found in the proc directory was left unchanged. The reg_pulse register was altered such that it would be able to handle 32 bit values for the PRI offset.

The original developer catered for the future improvement of the controller and had 16 bits of unused bits available for use. This free space was made use of in order to increase the register size of the PRI from 16- to 32-bits. In table 1, the various parameters required to set up a pulse are shown. Figure 1 shows how the parameters seen in table 1 would be setup using the echo command.

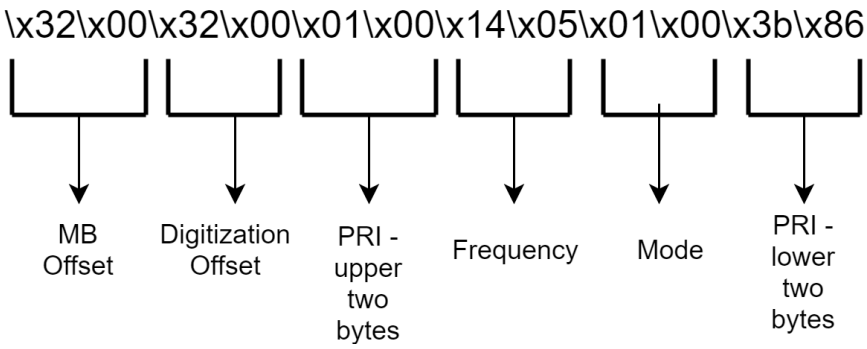

Figure 1: Visual depiction of how the parameters are setup using the echo command

The parameters seen in table 1 are set up by writing to the reg_pulses register. If you wanted to set up a pulse that triggers every 1 ms (1Khz), has a main bang and digitisation offset of 500ns, a band frequency of 1300 MHz and operating at L-band, you would write the following command.

```
echo -e -n "\x32\x00\x32\x00\x01\x00\x14\x05\x01\x00\x3b\x86"
> /proc/<PID>/hw/ioreg/reg_pulses
```

The Rhino operates using a 100MHz clock, which is 10ns cycles. For the main bang and digitization offset, we divide the desired offset by 10ns to get the number of cycles. Since we want an offset of 500ns for both of them, the number of cycles is equal to 50 (x0032 in hexadecimal). The value must be in little endian thus we set these to \x32\x00.

The desired PRI is calculated using the following equation:

$$PRI = \frac{\frac{1}{PRF}}{10ns} - MB - D$$

Where

- PRF is the frequency we want at the output i.e coming out of the main bang pin on the Rhino board
- MB is the main bang offset in cycles
- D is the digitization offset in cycles

Since MB = D = 50 and we want a PRF of 1kHz, we get a value of 99899 for the PRI. Converting the value to hexadecimal we get x0001 863B. Then you would write \x01\x00 to PRI upper two bytes and \x3b\x86 to PRI lower two bytes.

3 Experimental setup

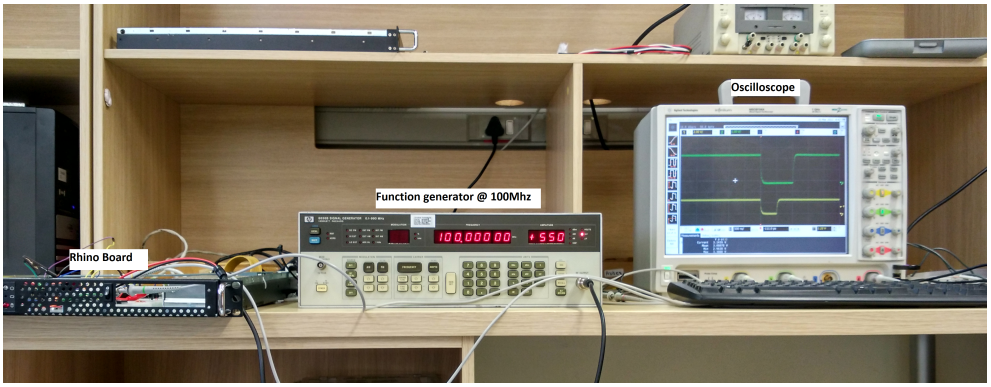

Figure 2: Picture of the equipment used to test the 32 bit version of the synchronization controller

The experimental setup used to test the modified controller can be seen in figure 2. A signal generator operating at 100MHz with an amplitude of +550mV was used as the input clock. This clock signal was passed through a diode then fed into the rhino. An oscilloscope was used to measure the Main bang offset and Digitisation signals coming out of the rhino.

Mention stuff about Xilinx IDe, computer specs, operating system and cabling used

4 Results of the Experiment

4.1 Using the Rhino Board Internal 100MHz clock

Testing of the modified synchronization controller began with using the internal clock of the Rhino board. Using the internal clock, 1kHz, 2kHz, 3kHz and 4kHz signals were produced.

Figure 3 shows the output of the main bang and the digitisation signals measured from the Rhino board. A time division of 1.0 ms was used to display the output.

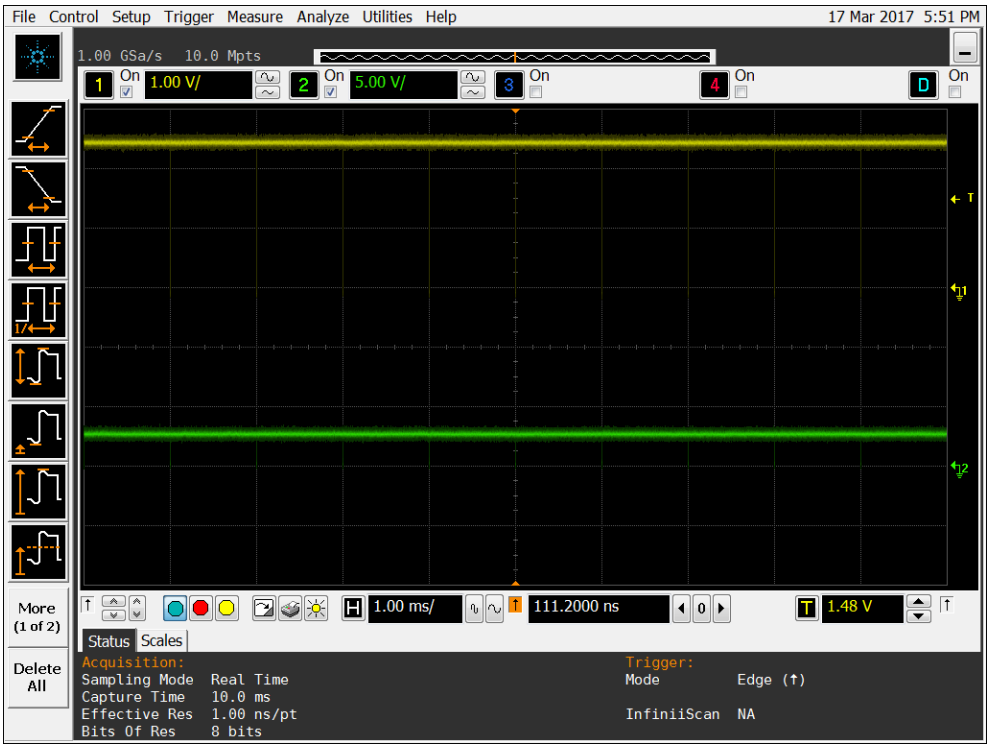

Figure 3: 1khz signal produced using the internal clock of the rhino

4.2 Using an External 100MHz Clock Signal

5 Conclusions

6 Recommendations