Update of the register size used in the NeXtRad Synchronization Controller

Dominic Manthoko

April 14, 2017

1 Introduction

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.

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.

Listing 1 below shows a snippet of some lines of code from gpmc_test.vhdl. It contains the vhdl description of the TCU and was modified to meet the new specifications.

```
1 -- 16 bit versions  of the P and Pcounter
2 signal P        : integer range 0 to 65535 := 0;
```

```
3  signal Pcounter : integer range 0 to 65535 := 0;
4
5  -- 32 bit versions of P and Pcounter
6  signal P        : std_logic_vector(31 downto 0) := x"00000000";
7  signal Pcounter : std_logic_vector(31 downto 0) := x"00000000";
8
9  -- populate dataout from regbank based on Program Counter (PC)
10 dataout <= reg_bank(PC) & reg_bank(PC+1) & reg_bank(PC+2) &
       reg_bank(PC+3) & reg_bank(PC+4) & reg_bank(PC+5);
11 MB <= conv_integer(reg_bank(PC));
12 D <= conv_integer(reg_bank(PC+1));
13 P <= reg_bank(PC+2) & reg_bank(PC+5);
14 ...
```

Listing 1: snippet of the VHDL code used to specify the TCU controller

Lines 2 and 3 show the previous versions of the P and Pcounter variables. The P variable represents the PRI offset and Pcounter is a counter that counts up to the set value for P. As can be seen, these variables were originally 16 bit integer values. Lines 6 and 8 show the P and Pcounter variables as 32 bit std_logic_vector's.

In order to read in the P value as 32 bits, two 16 bit values needed to be concatenated as down on line 13. The values needed to set up the TCU such as the main bang offset, digitization offset etc., are kept in a register called reg_bank. Register positions 2 and 5 are used for the PRI offset.

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:

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

```
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.

The parameters seen in table 1 are set up by writing to the reg_pulses register. If you wanted to set up a signal 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.

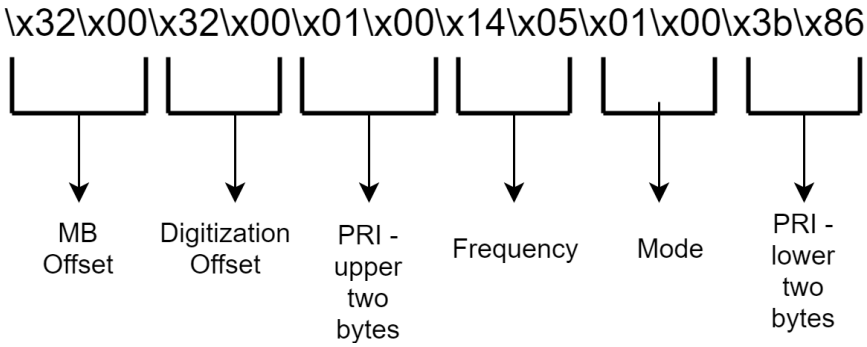


Figure 1: Visual depiction of how to order the parameters seen in Table 1

```
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 equivalent to 10ns cycles. The order of bits are swaped when using the gpmc interface between the ARM chip and FPGA, thus the developer needs to swap the byte order i.e. provid the desired parameters in little endian

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. Remembering that we need to provide parameters in little endian, then you would write \x01\x00 to PRI upper two bytes and \x3b\x86 to PRI lower two bytes.

3 Experimental setup

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

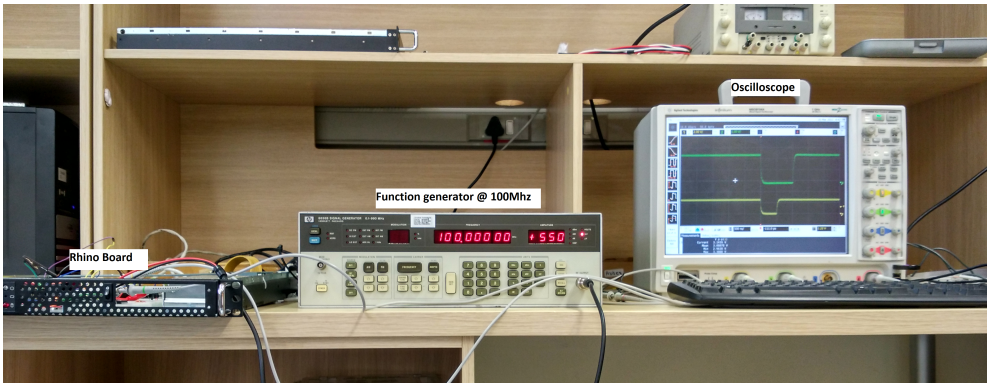


Figure 2: Picture of the equipment used to test the 32 bit version of the synchronization controller

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.

A computer running the 64 bit version of xubuntu 16.04 LTS; which is a linux distribution based on ubuntu; was used to modify the VHDL code. Xilinx IDE verison 14.7 was used to edit the original TCU project which can be found here.

4 Results of the Experiment

4.1 Using the Rhino Board Internal 100MHz clock

Testing of the modified synchronization controller began by making use of 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. Observing the yellow signal, it can seen that a 1kHz signal was indeed being produced by the Rhino board.

Figure 4 shows a zoomed in image of Figure 3. The yellow signal depicts the main bang signal and the green signal depicts the digitization signal. The yellow signal goes high after 500ns and the green signal triggers 500ns after

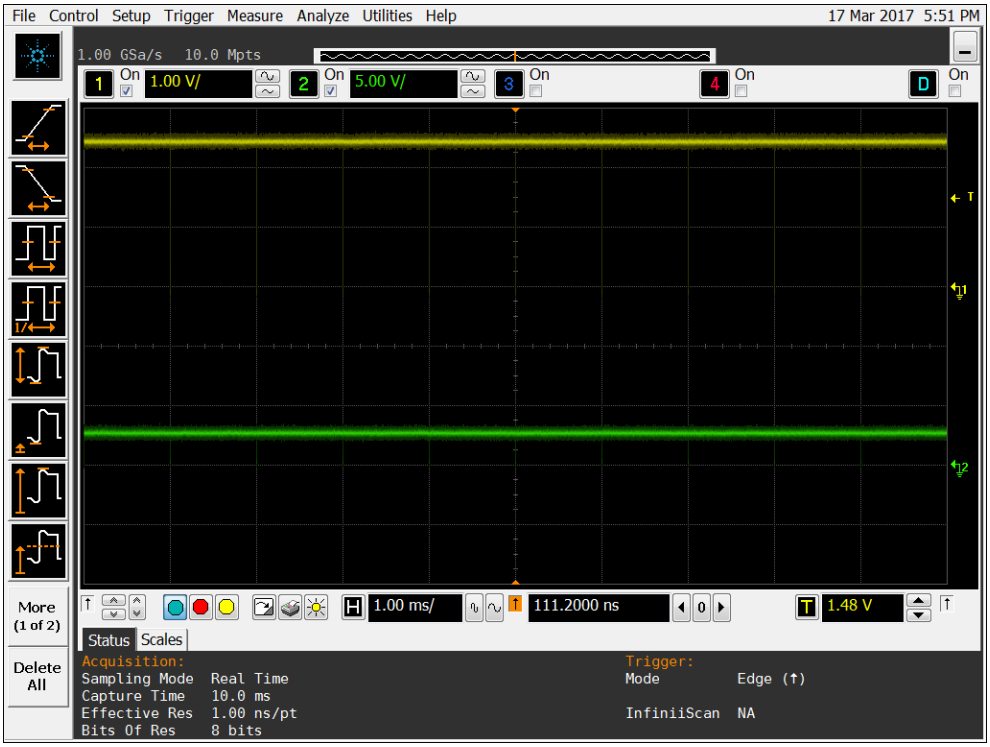


Figure 3: Image showing a 1kHz signal produced using the internal clock of the rhino

the yellow signal. These results are as desired.

To change the main bang offset and the digitization offset to 1000ns without changing the other parameters, the following command was used:

```
echo -e -n "\x64\x00\x64\x00\x01\x00\x14\x05\x01\x00\x3b\x86"
> /proc/<PID>/hw/ioreg/reg_pulses
```

Figure 5 shows am image of the main bang and digitization signals with 1000ns offsets. With a 500ns time division on the horizontal axis, it can be seen that it took two time divisions for the main bang signal(yellow signal seen at the top of figure 5) to go high. This equates to 1000ns, which is the desired result. Similarly for the digitization signal, we see that it also took 2 time divisions to trigger, thus meeting our desired result.

To show that the main bang offset and digitization offset can be set to different values, the main bang offset was set to 1650 ns and the digitization offset was set to 2950ns. The following command was used to set these values:

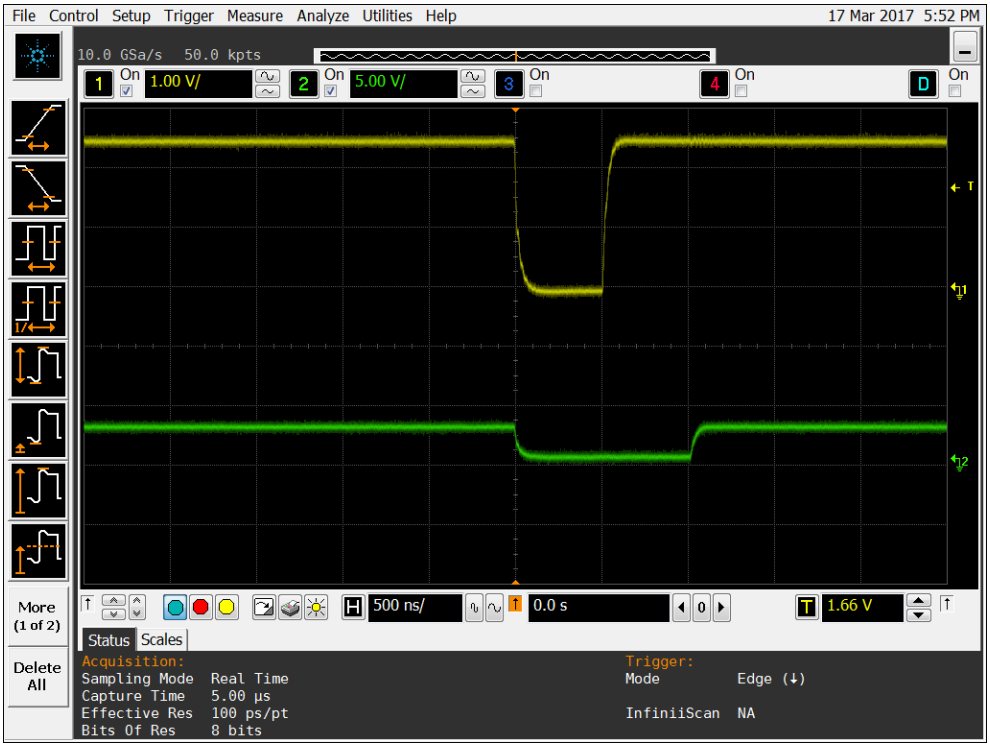


Figure 4: Zoomed in image of Figure 3

```
echo -e -n "\xa5\x00\x27\x01\x01\x00\x14\x05\x01\x00\x3b\x86"
> /proc/<PID>/hw/ioreg/reg_pulses
```

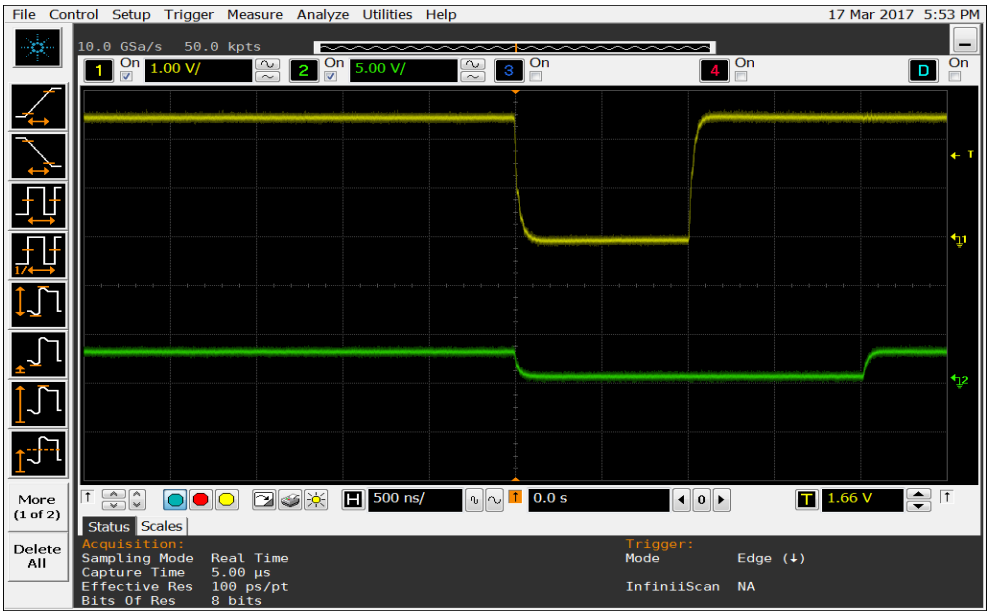
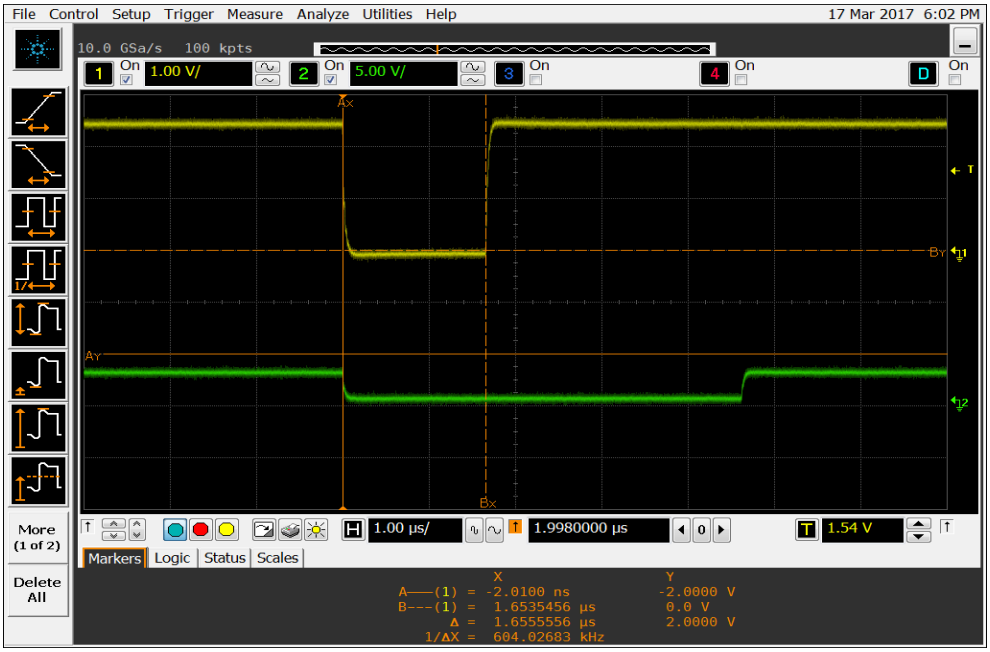
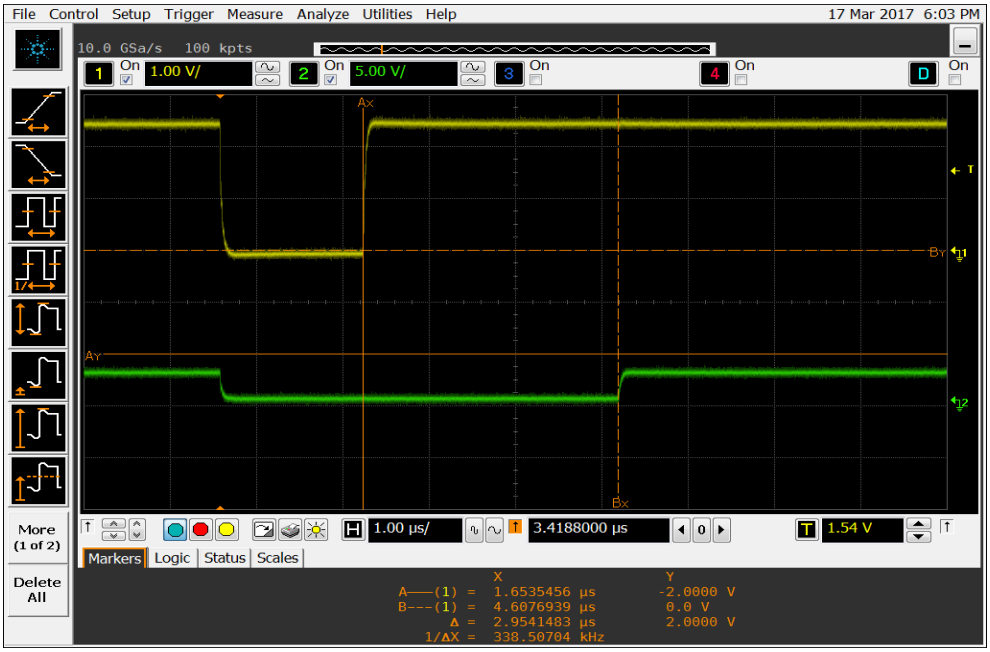


Figure 5: Image of the main bang signal and digitization with 1000ns offsets



(a)



(b)

Figure 6: (a) Main bang and digitization signal with the measurement of the main offset shown in the markers box toward the bottom of the image. (b) Same as figure 6a but with the measurement of the digitization offset shown in the markers box.

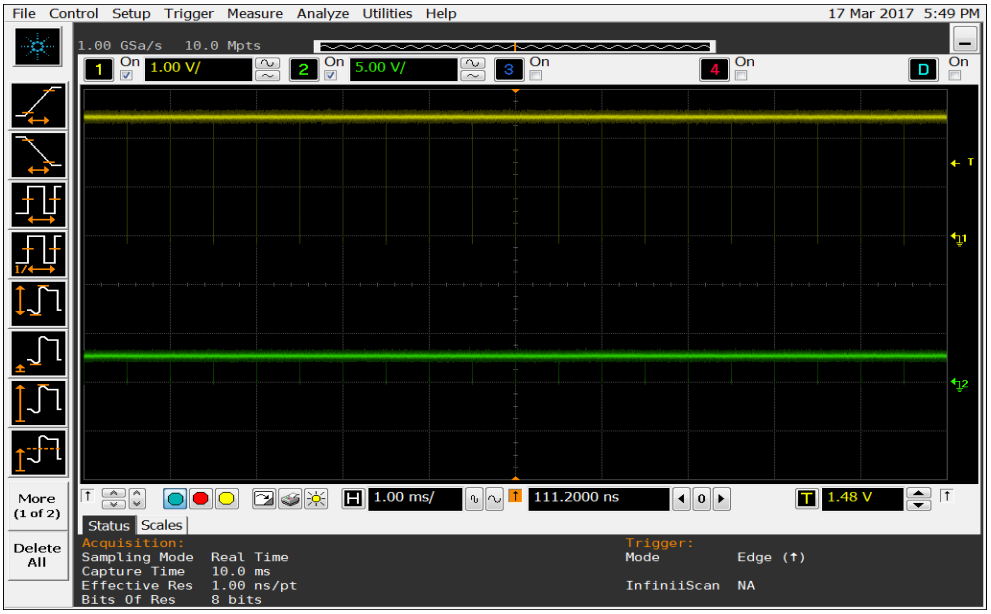


Figure 7: Image showing a 2kHz signal produced using the internal clock of the Rhino board

Figure 6 shows the plots of the main bang signal and the digitization signal together with the measurements. Making use of cursors on the oscillscope, which allow you to measure a value on the horizontal and vertical axis, the offsets were measured. In the markers tab seen in figure 6a, there is a delta value. From this plot, we see that $\Delta = 1.6555556\mu$. Observing the delta seen in figure 6b, we see that $\Delta = 2.9541483\mu$.

A main bang offset of 1650ns is equivalent to 1.65μ. The value of delta seen in figure 6a is thus approximately equal to the value that was set using the echo command. For the digitization offset, we set a value of 2950ns (2.95μ). The delta value seen in figure 6b is also approximately equal to the value that was set. This proves that the main bang and the digitization offsets can be set to two different values.

Figures 7, 8 and 9 show images of the output produced when a signal with either a 2kHz, 3kHz or 4kHz frequency is set using the echo command. Each of the plots was set using a 500ns offset for the main bang and digitization offsets, using a band frequency of 1300 Mhz, operating at L-band. The only that changes when switching between frequency is the PRI that is set.

To produce a 2kHz signal, we would need a PRI of 49899 (calculated using

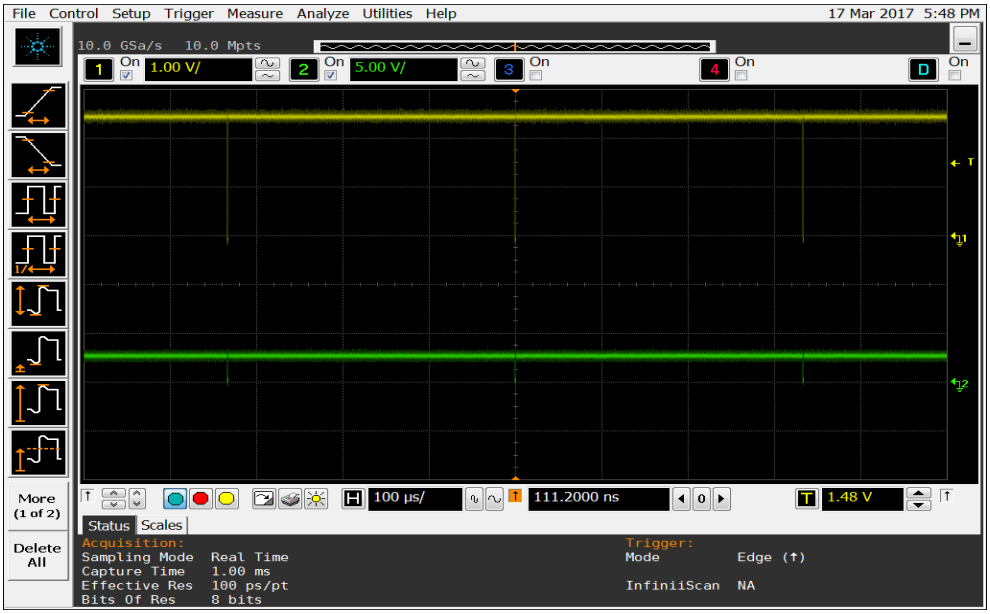


Figure 8: Image showing a 3kHz signal produced using the internal clock of the Rhino board

the equation for PRI found in section 2.2). The following command was used to set the parameters for the 2kHz signal:

```
echo -e -n "\x32\x00\x32\x00\x00\x00\x14\x05\x01\x00\xeb\xc2"
> /proc/<PID>/hw/ioreg/reg_pulses
```

To produce the 3kHz and 4kHz signals, the same command as above was used, the only difference being that a new PRI value was calculated and this new value plugged into the appropriate position in the command.

Observing figure 7, we see that a 2kHz signal was being produced. A 1.00ms time division was being used and in 1 time division, the signal goes high twice. This equates to the signal going high every 2ms which is equivalent to 2kHz.

Observing figure 8, which has a time division of $100\mu s$, we see that a 3kHz is being produced. The main bang signal (yellow signal) goes high at about 3.3 time divisions. When you multiply the number of time divisions with the value of 1 time division, we get $330\mu s$. This is equivalent to 3kHz, which is exactly what we wanted. From figure 9 we can see that a 4kHz signal is being produced. A time division of $500\mu s$ is being used. Thus 1 time division is

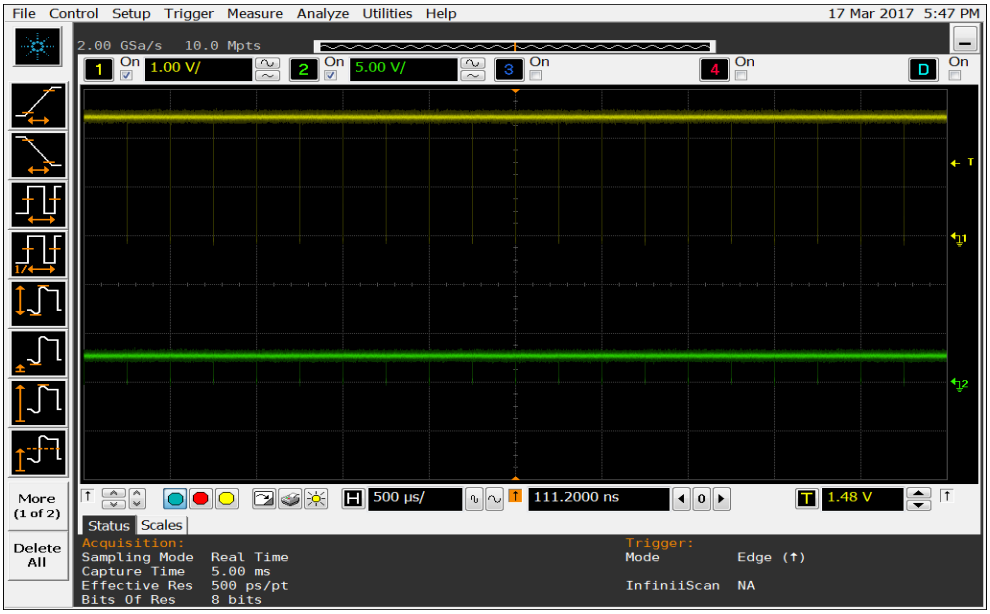


Figure 9: Image showing a 4kHz signal produced using the internal clock of the Rhino board

equal to 2kHz. In this picture, we see that the yellow signal went high twice for every time division. Thus, we have twice the frequency observed in a single time division. i.e. a 4khz signal is being produced.

4.2 Using an External 100MHz Clock Signal

Initial testing was done by making use of the internal 100MHz clock of the rhino board as mentioned in the previous section. Since the final set-up would be run using an external 100MHz clock, tests were conducted to ensure the updated synchronization controller would be able to operate correctly using an external clock.

The same commands used during the internal clock tests were used for the internal clock tests. The command used to setup up a signal 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 is shown below.

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

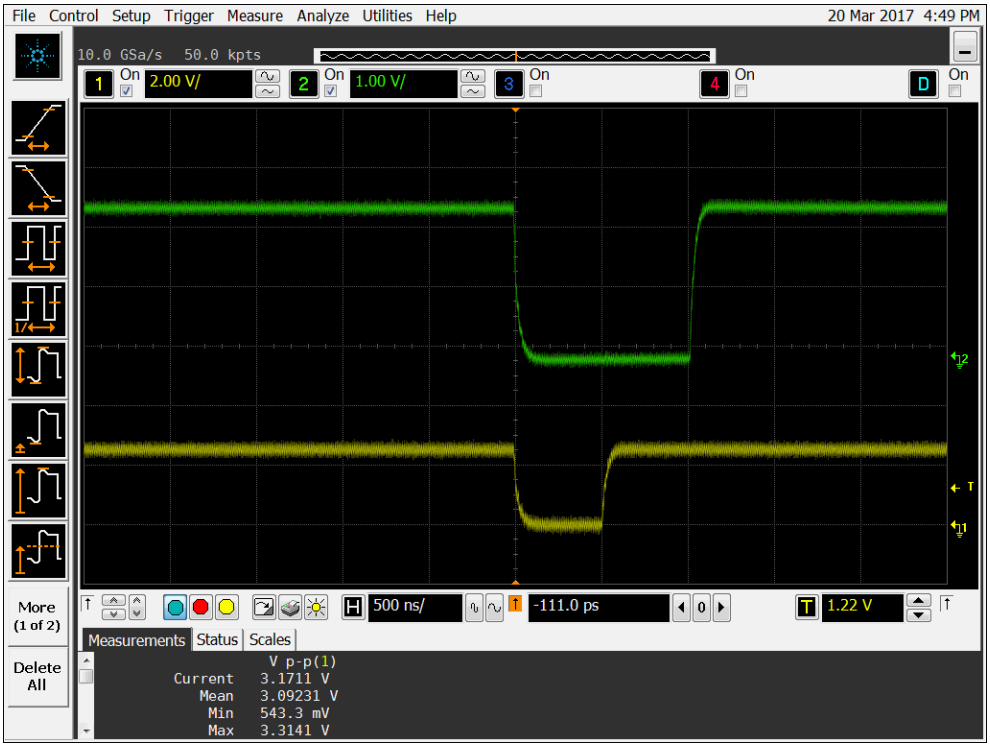


Figure 10: Image showing a 1khz main bang signal(yellow) and the associated digitization signal, both with an offset of 500ns

Figure 10 shows the output produced by using the above command. A 500ns time division was used to display the output. The yellow signal (bottom signal) represents the main bang signal and the green signal represents the digitization signal. As can be seen in the image, it took a single time division for the main bang signal to trigger high. The digitization signal took a further 500ns to do the same. We set the main bang and digitization offsets to 500ns and figure 10 proves that these values are correctly set up. The output seen in figure 10 is also in agreement with the output seen in figure 4.

To change the main bang offset and the digitization offset, the same command used in previous section was used. The command can be found here. The output produced after making the change can be seen in figure 11. A time division of 500ns was used to display the image. Observing the yellow (main bang) signal, we see that it took 2 time divisions for the signal to trigger high. This means that it took 1000ns in total. A similar observation was made about the green (Digitization) signal.

Tests were then done to see if the 2kHz, 3kHz and 4kHz signals could be

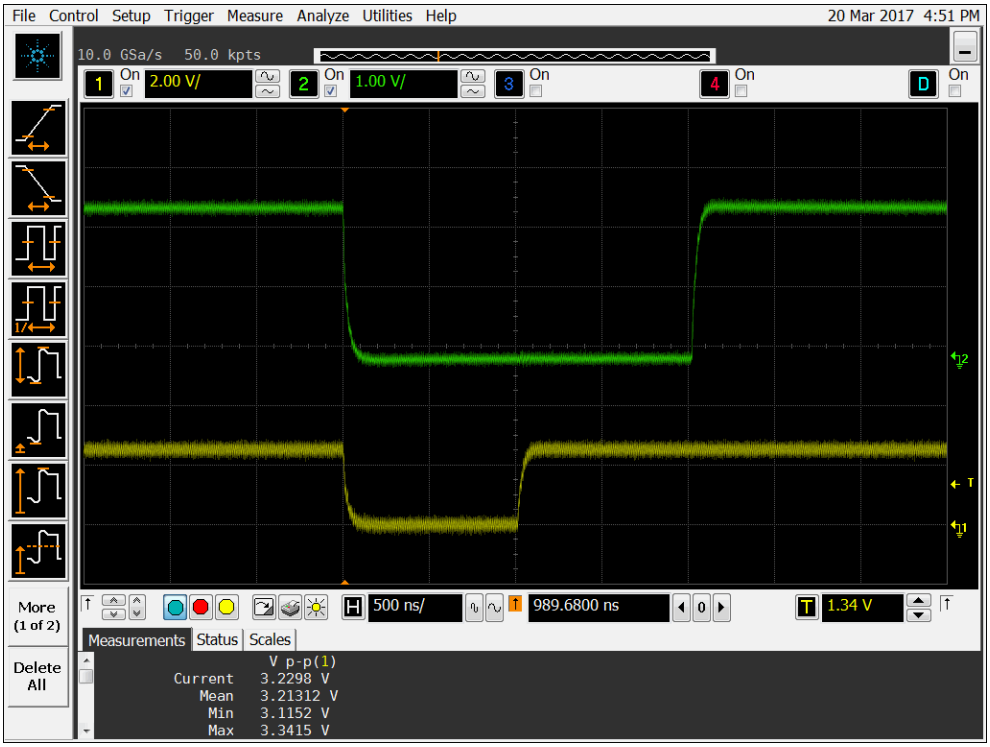


Figure 11: Image showing a 1khz main bang signal(yellow) and the associated digitization signal(green), both with an offset of 1000ns

produced using the external clock. The commands used to set the rhino to produce to appropriate commands are shown below.

The following command was used to set the rhino up to produce a 2kHz main bang signal:

```
echo -e -n "\x32\x00\x32\x00\x00\x00\x14\x05\x01\x00\xeb\xc2"
> /proc/<PID>/hw/ioreg/reg_pulses
```

The produce the 3kHz signal, this command was used:

```
echo -e -n "\x32\x00\x32\x00\x00\x00\x14\x05\x01\x00\xd0\x81"
> /proc/<PID>/hw/ioreg/reg_pulses
```

And to produce the 4kHz signal, this command was used:

```
echo -e -n "\x32\x00\x32\x00\x00\x00\x14\x05\x01\x00\x43\x61"
> /proc/<PID>/hw/ioreg/reg_pulses
```

Figure 12: Image showing a 2kHz main bang signal (green) and the digitization signal (yellow)

Similar to what was done for the internal clock tests, each of the signals was set using a 500ns offset for the main bang and digitization offsets, using a band frequency of 1300 Mhz, operating at L-band. The only that changes when switching between frequency is the PRI that is set. The only difference being that the value that was specified for the signal.

Figure 12 shows an image of the output produced when a 2kHz signal was set. A 1ms time division was used to display the output. Observing the green signal, which represents the main bang signal, we see that the signal goes high twice in a single time division. This means that we have twice the frequency represented by a single time division. In this, it means that the a 2kHz signal is being produced by the main bang signal output of the rhino board. This is what was expected.

Figure 13 shows images of the output produced when a 3kHz and 4kHz signals were set up using the aforementioned commands. A 1ms time division was used to display the output. From figure 13a, we see that the main bang signal goes high about 3 times in a single time division. The time interval between

each trigger high level is evenly spaced. This means that the frequency of the main bang signal is indeed 3kHz. Similarly, in figure 13b, we see that the main bang signal goes high about 4 times in a single time division. This means that the frequency of the main bang signal in the second figure is 4kHz.

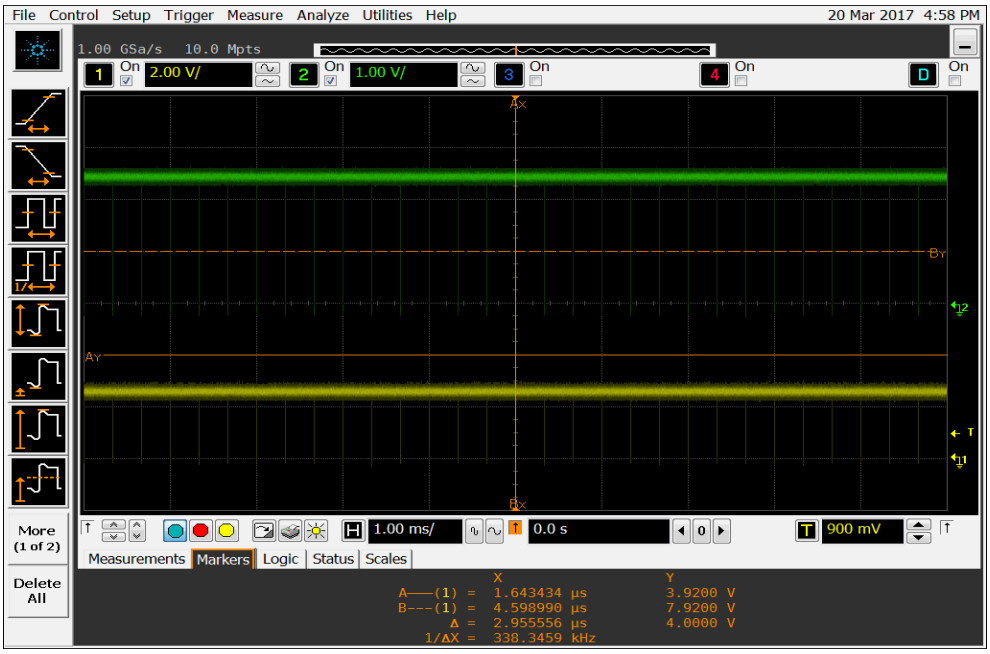
5 Conclusions

Modifying the PRI register size from 16 bits to 32 bits has not affected the rest of the functionality of the TCU controller. As observed in the results section, a user would be to set the main bang offset, digitization offset and other parameters to their desired values and the rhino would produce behaviour which closely approximates the set values for these parameters.

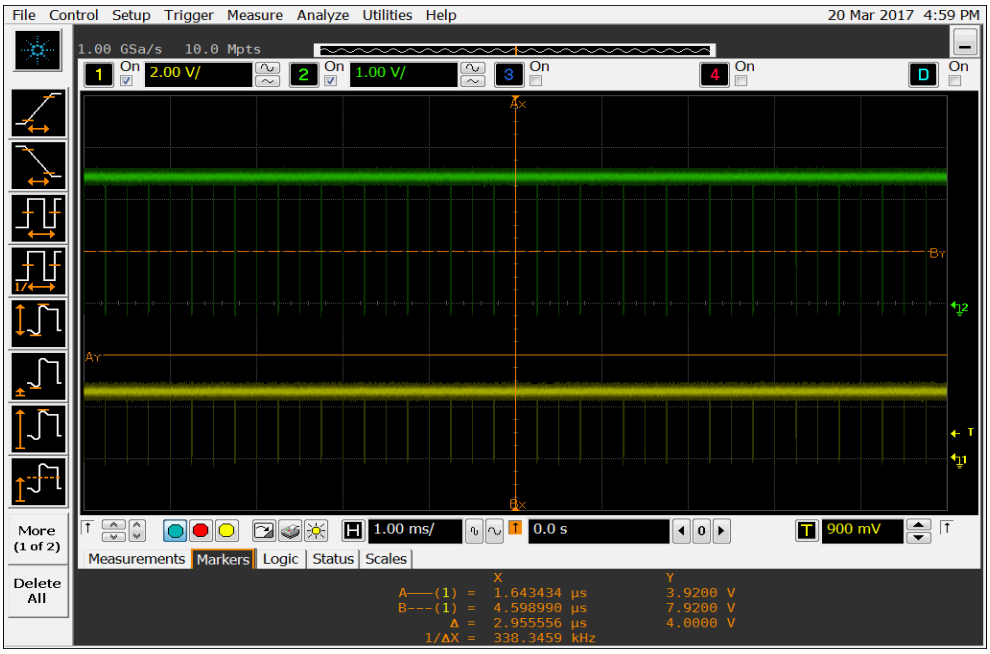
Additionally, it was proven that the rhino was capable of accepting a 32 bit value for the PRI offset and able to produce a main bang signal of some specified frequency.

6 Recommendations

The CLI(Command line Interface) program and the gui for the TCU need to be updated to so that they can handle 32 bit values for the PRI offset.



(a)



(b)

Figure 13: (a) Main bang and digitization signal produced when the rhino was set up to produce a main bang signal of 3kHz. (b) Same as (a) but for the case when a 4kHz signal was set up