Synchronizing the LimeSDR transmitter and receiver #269

Closed **kohlsne** opened this issue on Jul 21, 2020 · 3 comments

kohlsne commented on Jul 21, 2020 • edited

I am trying to make a FMCW radar with the LimeSDR paired with a Raspberry Pi 4. I am currently using python. For testing I use a 1/4 meter coaxial cable to short the transmitter and receiver together. The issue is that even when using the timestamps the receiver seemed to be delayed when compared to the transmitter. I print out the timestamps and it seems like it would work, but when I mix the transmit and receiver frequency to get a difference of a chirp I find that it is not near zero. Timing is crucial in a radar system and I need control of when the LimeSDR is transmitting and receiving.

Everything looks fine when I plot the receive signal. It looks like I received the chirp.

I've looked at issue #138

I can not pinpoint the issue. I'm guessing it might be an issue with the MTU? or with Timeout?

```
Fs = int(4.096e6)
FreqCarrier = 37e6
sdr.setMasterClockRate(30720000)

sdr.setSampleRate(SOAPY_SDR_RX, 0, Fs)
sdr.setFrequency(SOAPY_SDR_RX, 0, FreqCarrier)
sdr.setGain(SOAPY_SDR_RX, 0, 32)  # power amp gain 32
sdr.setSampleRate(SOAPY_SDR_TX, 0, Fs)
sdr.setFrequency(SOAPY_SDR_TX, 0, FreqCarrier)
sdr.setGain(SOAPY_SDR_TX, 0, 64)  # power amp gain

txStream = sdr.setupStream(SOAPY_SDR_TX, SOAPY_SDR_CF32,[0])
rxStream = sdr.setupStream(SOAPY_SDR_RX, SOAPY_SDR_CF32,[0])
time.sleep(2) # let things settle

Tc = 0.025 #Create Chirp to transmit
sizeOfChirp = int(Fs * Tc)
Fc_start = 500e3
B = 1.0e6
Fc_end = Fc_start + B
t = np.arange(0,Tc,1/Fs)
beta = B/Tc
```

```
TXsignal = np.exp(2*np.pi*1j*(beta/2*(t**2) + Fc_start * t)).astype(np.complex64)
del t

flags = SOAPY_SDR_HAS_TIME | SOAPY_SDR_END_BURST

timeStamp = int(0.2e9)
timeoutUs = int(Tc * 1e6 * 1.1)

RXsignalBuff = np.array([0] * sizeOfChirp, np.complex64)
sdr.activateStream(rxStream, flags, timeStamp, len(RXsignalBuff))
sdr.activateStream(txStream)

status_tx = sdr.writeStream(txStream, [TXsignal], len(TXsignal), flags, timeStamp, timeoutUs=timeoutU
if status_tx.ret != len(TXsignal):
    raise Exception('transmit failed %s' % str(status_tx))

status_rx = sdr.readStream(rxStream, [RXsignalBuff], len(RXsignalBuff), timeoutUs=timeoutUs)
if status_rx.ret  != len(RXsignalBuff):
    raise Exception('receive fail %s' % (str(status_rx)))

sdr.deactivateStream(txStream)  # stop streaming
sdr.closeStream(rxStream)
sdr.closeStream(txStream)

print(timeStamp)
print(status_rx.timeNs)
print(sizeOfChirp)
print(status_tx.ret)
print(status_rx.ret)

signal = np.multiply(TXsignal, np.conj(RXsignalBuff))
signal_fft = np.log10(np.fft.fft(signal))
numOfSamples = int(np.squeeze(signal.shape, axis=0))
print(numOfSamples)
fft_fre = np.fft.fftfreq(n=numOfSamples, d=1 / Fs)
print(fft_fre[(np.where(signal_fft == signal_fft.max()))])
```

👍 1

guruofquality commented on Jul 21, 2020 Contributor

BTW only lime is really going to be able to address hardware specific issues:
https://github.com/myriadrf/LimeSuite

But speaking generally, usually the FPGA handles the timestamps, so anything that happens between the RF and the FPGA potentially adds group delay, such as digital up and down conversion chains, and buffering.

So I would expect to see a timestamp delay for a loopback waveform. That said it should be consistent from run to run for the same sample rate and filter configuration.

FWIW, there is a python example for doing just this: https://github.com/pothosware/SoapySDR/blob/master/python/apps/MeasureDelay.py You may have to play with gains a little bit, but I recall using it recently.

kohlsne commented on Jul 22, 2020 Author

That makes a lot of sense thank you!

guruofquality closed this as completed on Aug 22, 2020

MountainLogic commented on Aug 23, 2020

FWIW, while this can't be addressed in FW, this is a critical feature that is needed in the lime gateware and SDRs in general to be fully useful in radar

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants