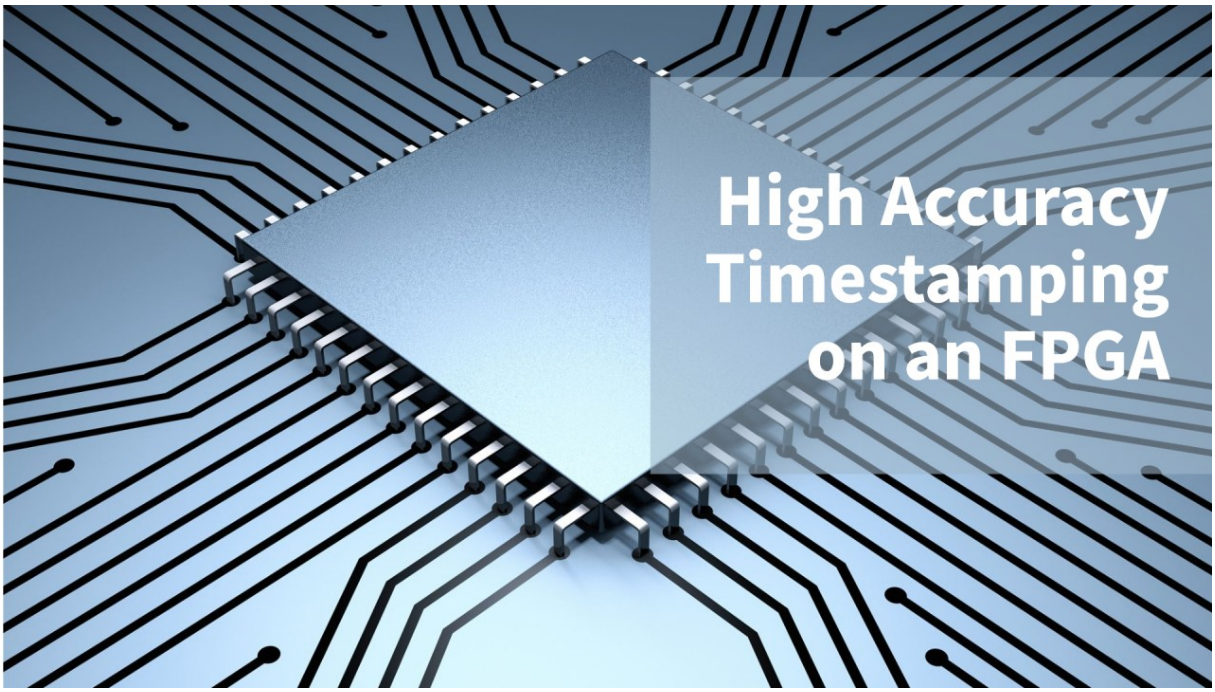


1 Nanosecond Timestamp Accuracy on an FPGA

5. April 2024

During the last weeks we implemented a 1 Nanosecond (ns) Timestamping Resolution for our IP cores.

We consider this a very important step since all our time synchronization solutions are basically based on timestamps. The more accurate the timestamps are, the more accurate is the synchronization.

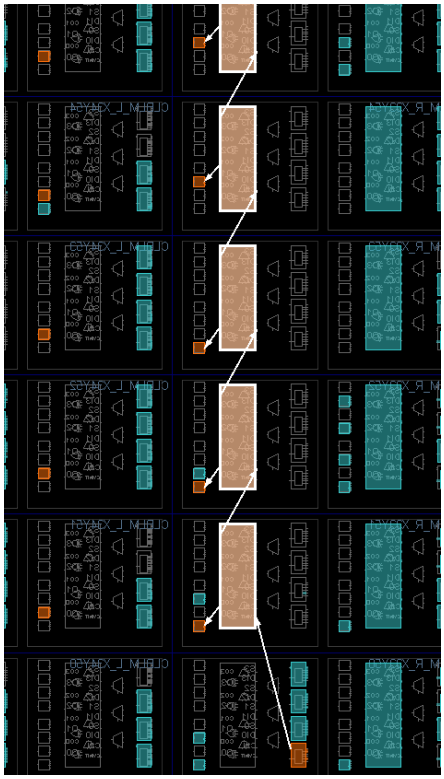
In principle a 1 ns timestamping is not that difficult to achieve in an FPGA, in fact even a resolution < 100ps is achievable the same way. For now, we only concentrate on a resolution of 1 ns since this is so far the minimal step in our time format of seconds and nanoseconds.

But How?

The most obvious approach to achieve a 1 ns resolution would be to run a counter at 1 GHz and take a snapshot of this on a rising edge of an event signal. Running logic at 1 GHz is simply not feasible in an FPGA (especially not with low cost FPGAs

like AMD Artix7 or Altera Cyclone10LP devices). When it comes to a resolution of < 100 ps the required frequency would be 10+ GHz which is obviously not an option even for high end FPGAs.

So, how do we achieve a 1 ns resolution in an FPGA then? The concept is called Time-to-Digital Converted (TDC) based on a Tapped Delay Line (TDL). A Tapped Delay Line is as the name says a line of **fixed delay elements** with taps between the delay elements. In an FPGA there are basically two elements which allow to form any functionality desired: Logic Cells (LC) and the Switch Matrix and Interconnect. A Logic Cell in principal contains a Lookup Table (LUT), a Flip Flop (FF), a Carry Logic (CL) and a Multiplexer (MUX). The Switch Matrix and Interconnect are used to connect Logic Cells as desired with each other. Looking at the available Elements in an FPGA, LUTs would be the obvious elements to form a delay line. However, the Switch Matrix and Interconnect has a non-constant delay which makes it impossible to use it for a Tapped Delay Line which requires a fixed delay for each delay element. The only fixed and dedicated paths in an FPGA are the **Carry Elements**. Each Logic Cell has a dedicated path to the next Logic Cell for the Carry signal in the same Column. This allows to build a **Carry Chain** of N Elements which forms the Delay Line, in addition there is also a dedicated path within the Logic Cell from the result of the Carry Element to the Flip Flop which is required for the Taps of the Tapped Delay Line.



Carry Chain with Tap Flip Flops

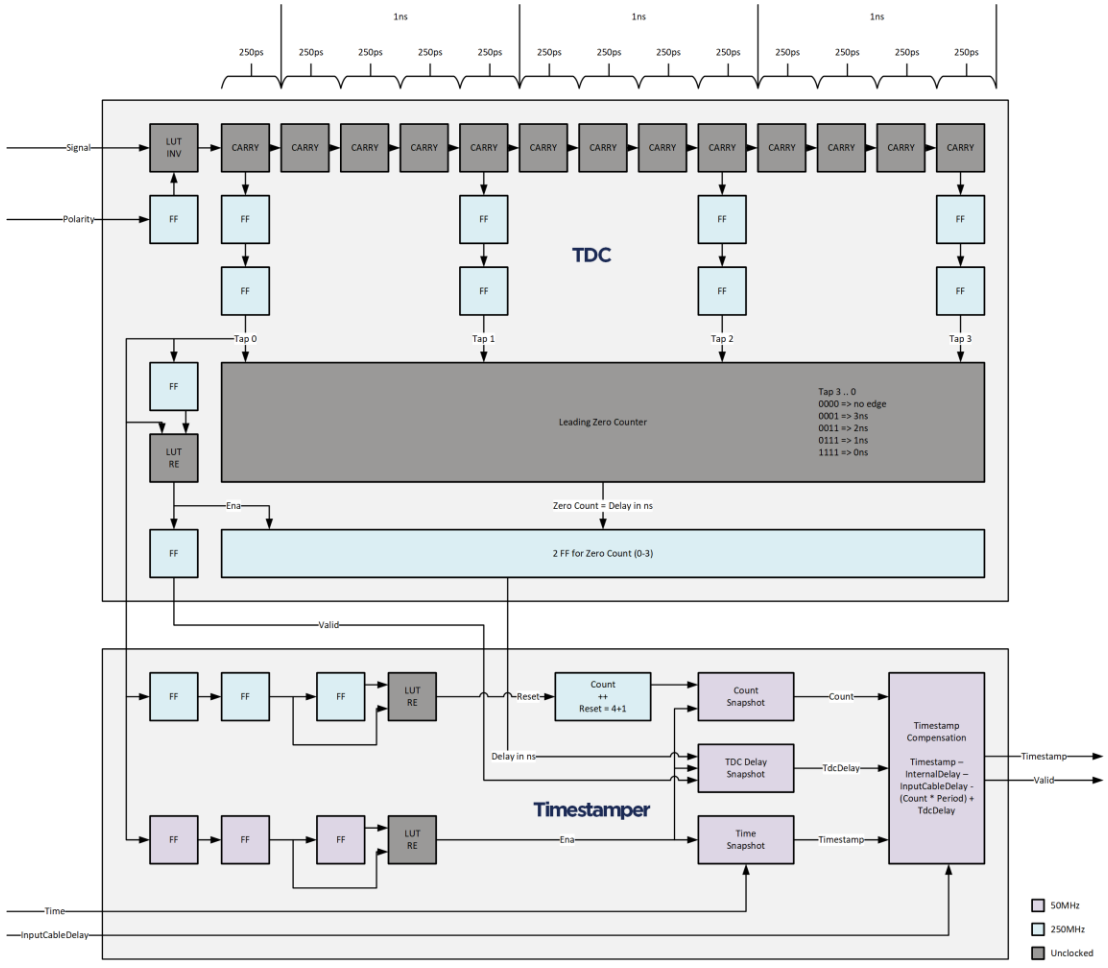
The delay of a Delay Element in the chain basically defines the resolution of the TDC. In an AMD Artix7 FPGA the delay of a Carry Element is ~28.5ps (the Artix7 fabric actually contains a CARRY4 Element which are 4 carry elements within one Logic Cell which has a Delay of ~114ps) which in principle would allow a resolution of ~30ps. The length of the Delay Line is defined by the Clock Frequency of the Flip Flops connected to the Taps of the Delay Line, since the Delay Line shall cover the whole period of the Clock Frequency. E.g. let's assume the sampling clock frequency is 250 MHz (4000 ps period) and the delay of a Carry Element is 250 ps (just for the calculation it is much less) the number of Carry Elements would be 4000 / 250 = 16. However, a bit less are actually required depending on the resolution desired, we see this later. From the example above you see the faster the Clock Frequency the less Carry Elements are required, this must be considered since in an FPGA you only have limited resources available within one Column, generally speaking use an as high as possible frequency for which you can still run the rest of the Converter logic of the TDC without timing violations. The number of Taps required depend on the desired resolution. Based on our example above and a desired resolution of 1 ns, this would require only 4 Taps: at 250 ps per Carry Element this would mean after 4 consecutive Carry Element we need a Tap. Coming back to the number of Carry Elements and the fact that we have the first Tap on the first Carry Element, the number of Carry Elements required is: ((Number of Taps - 1) x Carry Elements between Taps) + 1. For our example this means ((4 -1) x 4) + 1 = 13. At every 250 MHz clock cycle each Tap is sampled into a dedicated Flip Flop within the same Logic Cell and to avoid Metastability the output of the first Flip Flop is sampled into a second Flip Flop. The output of the second Flip Flops is then used to determine the Delay. For the example above the conversion is pretty simple. To calculate how much delay the rising edge has after a clock edge the leading zeros need to be counted. 0001 = 3 ns, 0011 = 2 ns, 0111 = 1 ns, 1111 = 0 ns (in the last). Only the leading zeros are counted so also other patterns are allowed and we are only interested in the first 0 to 1 transition. A pattern of 0000 means no rising edge. The first Tap is also used as a trigger to snapshot the encoded value, which will be done by feeding the output of the second Flip Flop of the first Tap through a third Flip Flop and using a LUT to detect a rising edge (0 to 1 transition) which will be fed to a Flip Flop to signal a Valid and to the enable input of the 2 sample Flip Flops (0-3 can be encoded in two bits) which are fed with the encoded value. And there you have it, a sampling resolution of 1 ns! To come to a timestamp which is accurate to 1 ns, additional steps and compensations are required as we will see in the next chapter.

NetTimeLogic's TDC enhanced Timestamper

Let's see how we have implemented and integrated the TDC in our Timestamping logic (which can also run without TDC with 4ns resolution). We intentionally leave some details out, but it gives you the basic concept.

We have an Adjustable Counter Clock which provides a Time in Seconds and Nanoseconds format, which runs normally at 50 MHz. 50 MHz is used to allow to run things also on very low-cost FPGAs. The 50 MHz for the Counter Clock are generated from an FPGA PLL which also creates a 250 MHz clock which is phase and frequency aligned to the 50 MHz clock. The alignment is required and can be guaranteed due to the fact that the two frequencies are generated from the same clock and the same PLL. Running only a very small part with 250 MHz relaxes the timing closure and is still possible with low-cost FPGAs and requires a decent length of the Delay Line (~140 cascaded Carry Elements on an Artix 7).

Below you see the simplified architecture of the TDC assisted Timestamper.



Timestamper architecture

As in the last chapter, for simplification of the diagram a Carry Element has 250 ps of delay (actually ~28.5 ps but this would require a huge chain in the diagram).

In the last chapter we have seen how the TDC part in general works and it was almost exactly implemented this way. Additionally, a LUT is instantiated before the first Carry Element to change the polarity of the of the input signal (also to clean the signal from the IO) and the output of the second Flip Flop of the first Tap is exported from the TDC part. This output is used as the Signal to Timestamp as it would be if no TDC would be used. This Signal is then fed through a chain of 3 Flip Flops for both the 50 MHz and 250 MHz clock domains. The first two Flip Flops are again used to overcome the Metastability issue (which doesn't exist when the TDC is used) and the third Flip Flop together with the output of the second Flip Flop is used to detect a rising edge. The rising edge detection on the 250 MHz clock domain will reset a Counter to 5 (2.5 if no TDC) to accommodate for the Flip Flop chain, it will then increment the counter by 1 for every 250 MHz clock cycle. This Counter will then represent the number of 250 MHz clock cycles since the rising edge of the signal. The rising edge detection on the 50 MHz clock domain will cause a snapshot of the 250 MHz Counter, the Time and the TDC Delay in Nanoseconds. Then the Timestamp is compensated by taking the raw Timestamp subtracting the Counter value multiplied by 4 ns (250 MHz), adding the TDC Delay and subtracting FPGA internal Delay (Routing and Logic Delays of the TDC) and a user provided Input Delay (e.g. external buffers on the hardware, cabling etc.) if any.

There you have your Timestamp with 1 ns accuracy!

Where do we use it?

The IP cores from NetTimeLogic which currently support already TDC are:

- Signal Timestamper
- PPS Slave

These two cores are also integrated into our PPS Analyzer Solution which therfore now also supports 1 ns of accuracy