Time Difference of Arrival (tdoa)

Timestamp Capture IP module

by Joel

TDOA Overall Scheme

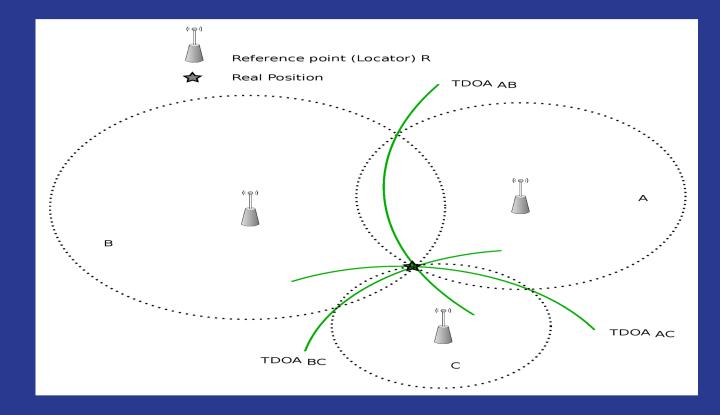
TDOA is based upon a technique called <u>multilateration</u>. Multilateration estimates the location of a wireless emitter, and the target position is determined by the time difference between signals arriving at multiple receivers.

Once these wireless emitted signals are captured and time-stamped via the Pico-Zed receivers they are then placed into a software based TDOA algorithm to determine the location of the emitter target.

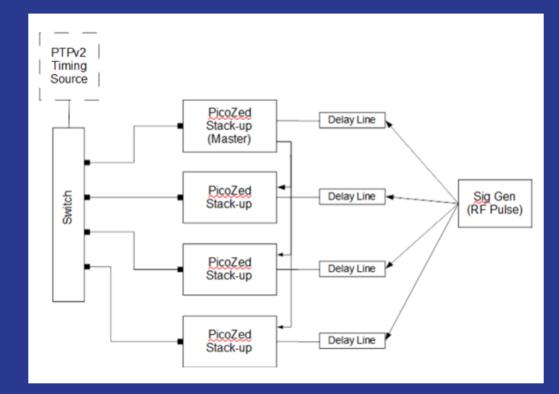
TDOA Illustration

Time Difference of Arrival (TDOA): Timestamp Capture IP Module Time Difference of Arrival (TDOA): Timestamp Capture IP Module

TDOA Illustration



TDOA Project Overview

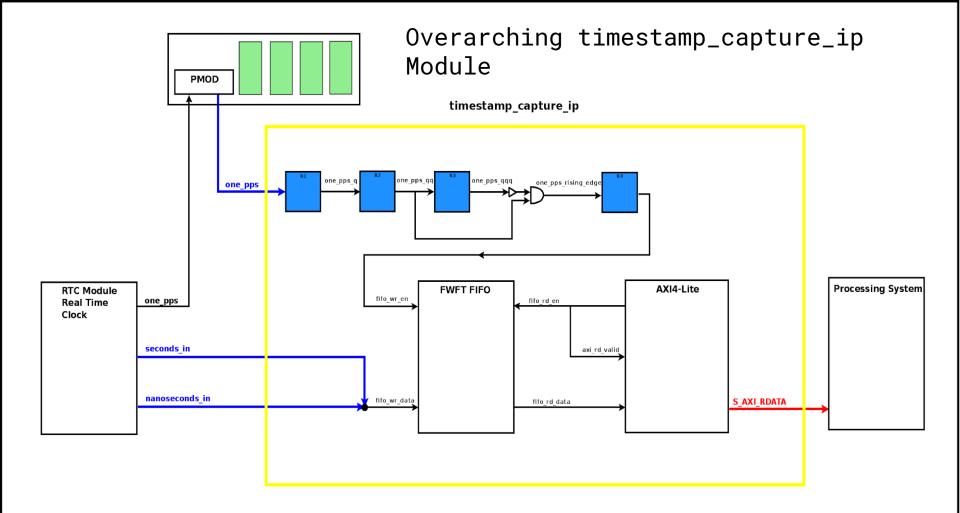


Tasking

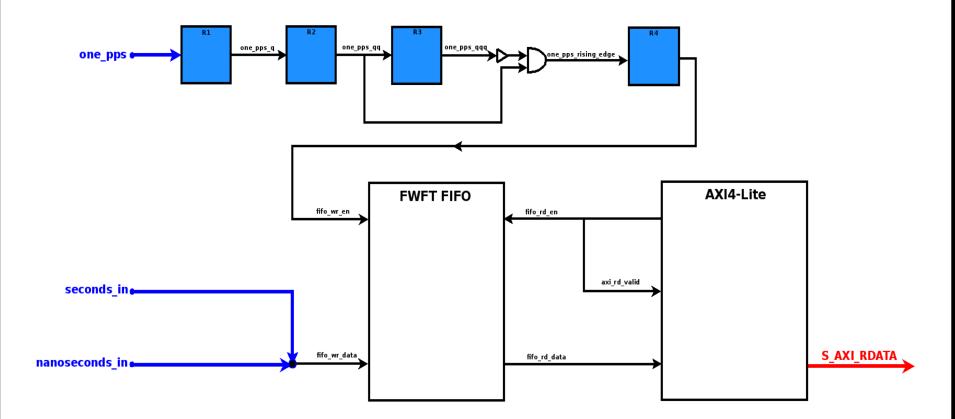
- Develop an AXI-Lite Firmware Intellectual Property (IP) core that can capture PTPv2 timestamps and be configured as a master or a slave.
- When configured as a master, the IP core should generate its own 1 Pulse Per Second (1PPS) digital output with a 1us pulse width.
- When configured as a master or a slave, the IP core should input the 1PPS signal and write the current timestamp (seconds and nanoseconds) to a FIFO. The FIFO should be readable from software.

Hurdles of Design

- Learning VHDL, and the Vivado ecosystem.
- Understanding individual design components such as the FWFT FIFO (First-Word fall through First-In-First-Out) component, and the AXI4-Lite (Advanced eXtensible Interface protocol) component.
- The 4:1 aspect ratio of the FIFO component.
- Time Domain synchronization registers.
- Rising-Edge detect of the 1PPS signal



Timestamp_capture_ip



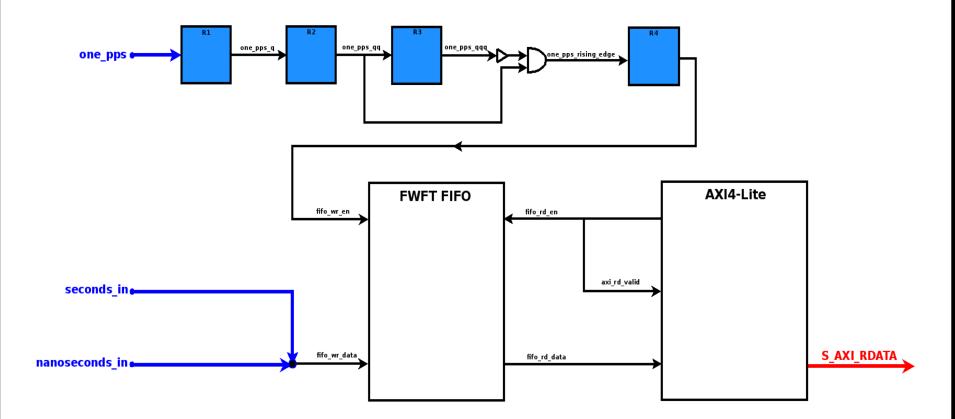
I/O Test-Bench

Name	Value	l0 ns	50 ns	100 ns	150 ns	200 r	217.500 n s		300 ns 3	<mark>357.500 ns</mark> 50 ns	400 ns	450 ns 5	<mark>507.500 ns</mark> 60 ns	550 ns	<mark>615.0</mark> 600 ns	000 ns 650 ns
INFRASTRUCTURE																
🐻 clk	1					INN	որդեր			1,0,0,0,0,0,0,0,0			1.000000000		WUU	
🔓 reset	0															
INPUT																
🔓 one_pps	0															
> 😽 seconds_in[47:0]	105105105105		0000000000	00			1000000	00001		111111	11111			10510510510	15	
> 😻 nanoseconds_in[31:0]	99999999		0000000				00000	002		2222	2222			999999999		
OUTPUT																
> 😻 S_AXI_RDATA[31:0]	99999999	. badadd	0 /	0000000	badadd	00	. 0000	. 0000	00000002	0/1111		22222222	00510	. 🗙 0000 🗙	99	999999
SIMULATION FLAG																
🐻 simulation_done	TRUE	(FALSE							TRUE

Time-stamp samples Test-Bench

Name	Value	 490 ns	500 ns	507.500 ns	520	ns	530 ns	540 ns	550 ns	1560 ns	570 ns	580 ns	590 ns	600 ns	610 ns	615.000 ns
INFRASTRUCTURE																
🐻 clk	1															
🖥 reset	0															
INPUT																
🔓 one_pps	1															
😽 😽 seconds_in[47:0]	105105105105								105105105105							
😽 nanoseconds_in[31:0]	99999999								99999999							
OUTPUT																
S_AXI_RDATA[31:0]	00001051	22222222		00	001051	X		05105105	χ.		00000000	<u>х</u>		99999999		
SIMULATION FLAG																
lasimulation_done	FALSE							F	ALSE						i X	TRUE

Timestamp_capture_ip



Overall Test-Bench

Name	Value	0 ns	100	ns		217.500 ns) ns	300 ns	357.500	<mark>) ns</mark> 400 ns	l ^s	<mark>507.500 ns</mark> 00 ns	<mark>615.000 ns</mark> 50 <mark>0</mark> ns	700 ns
INFRASTRUCTURE													
14 clk	1	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	NO NO NO DO NO NO				ADDODODODOD		0.00.00.00.00.00.00.00.00.00.00.00.00.0		10 00 00 00 00 00 00 00 00 00 00	M .	
Ъ reset	0												
INPUT													
🔓 one_pps	0												
> 😽 seconds_in[47:0]	105105105105	000	00000000		10	0000000001		1111111	11111				1051051
> 😻 nanoseconds_in[31:0]	99999999	0	0000000	X		00000002		22222	222	X			99999
one_pps / Rising-edge													
🖁 one_pps_q	0												
🖁 one_pps_qq	1												
🖁 one_pps_qqq	1			[
	0	L		ſ									
FIFO / AXI					_					_			
🖁 fifo_wr_en	0												
> 😽 fifo_wr_data[127:0]	0000105105105105000000009999999	000000000			000100	000000001000			1111000	<u> </u>		000010510	51051050
、、	0		<u> </u>	C		0 0 0	0		0 0	0		K	
> 😼 axi_rd_valid[0:0]	0		K	C						0		K	
> 😼 fifo_rd_data[31:0]	99999999	K	00000000		X)	(,X,X 00	000002 <u>(</u> .	·X···X) 2222	2222)(.	·X····X····X		
OUTPUT													
> 😽 S_AXI_RDATA[31:0]	99999999	🗙 badadd00	000000	000 Xbad	la)	(XXX.	00000002	<u>Х</u>	X X 22	222222			
SIMULATION FLAG													
🔓 simulation_done	TRUE	Ķ				F	ALSE					_¥	

Notes

- The 4:1 aspect ratio of the FIFO.
- The 'O' padding of the 32 bit data words.

Next Task

- Add IP to repo and pico_zed_fmc2 repo as a submodule.
- Build IP with 'make'.
- Build the picozed_fmc2 with 'make'.
- Open the picozed_fmc2 project in Vivado, open the block design.
- Add a user IP repository in the Vivado IP Catalog that points to the IP.
- Instantiate the IP in the Vivado block design.
- Connect the IP to the Processor System (Zynq) in the block design.
- Connect the IP to the Real-Time Clock in the block design.

End Presentation