Quick start

• For quick-start guide see slide 30
LinoSPAD2 hardware
LinoSPAD2 A5 daughterboard + 2 Spartan 6 FPGAs (#36 and #37)

5 V board power

Sensor powering

USB readout

512 pixels
Sensor powering
In each FPGA

- 64 TDCs (time-to-digital converter), 4 pixels connected to each
- FW 2208: (TDC: pixel) 0: 0, 64, 124, 192
- FW 2212 block: 0: 0, 1, 2, 3
- FW 2212 skip: 0: 0, 64, 124, 192
FPGA LEDs

LED on = board is powered

LED on = FW is installed and working well
Firmware

• FW 2208: at this point, used purely for debugging purposes. Due to TDC-pixel connections, HBT peaks only between pixels in groups of 64 (0-63, 64-127, 128-191, 192-255) are possible.

• FW 2212 (block): main version to work with; HBTs are possible across all 256 pixels

• FW 2212 (skip): same as FW2208, currently not utilized
Software for communication with the FPGA (FW 2212; similar layout in FW 2208)
TDC data acquisition

- Number of cycles
- Length of each cycle: 4 ms max (and recommended)
- Number of timestamps in each cycle (memory buffer-defined)
- Switch for repeating X times (saving X files)
- Auto save, should go hand in hand with “Auto repeat”; suffix for path where to save to

Leave on for binary (.dat) output

5/31/2023
Binary output

• FPGA exchanges 32-bit words with PC
• Lower 28 (0-27) bits are the timestamp (packet=32 bits: packet & 0xFFFFFFFF)
• Bits (28-29) are the address (0, 1, 2, 3) in the given TDC (packet >> 28)
• Address for FW 2212 block: pixel number = 4 * TDC number + pix address. Example: TDC = 17 (TDC ∈ [0, 63]), pix address = 3: 4 * 17 + 3 = 71
• Bit 31 is the validity bit: if 1 => timestamp is valid, otherwise it should not be accounted for in the analysis
• Bit 30 is not used
Reference clock: see next slides for more details
Debug: should be green if connected, “FPGA timestamp” should correspond to the installed version.
FW 2208: clock setup
Clock setup: FW2208, 4 ms acquisition window, 250 Hz clock

Reference output does not correspond to the desired frequency (4 ms – 250 Hz, 500 us – 2 kHz)

Up to 512/pixel/acq. cycle
Clock setup: FW2208, 4 ms acquisition window, 250 Hz clock

Reference output does not correspond to the desired frequency (4 ms – 250 Hz, 500 us – 2 kHz)
Clock setup: FW2208, 500 us acquisition window, 2 kHz clock

Reference output does not correspond to the desired frequency (4 ms – 250 Hz, 500 us – 2 kHz)
FW 2212: clock setup
Clock setup: FW2212, 4 ms acquisition window, 250 Hz

Reference output corresponds to the desired frequency (4 ms – 250 Hz, 500 us – 2 kHz)

Up to 1536/TDC/acq. cycle
Clock setup: FW2212, 500 us acquisition window, 2 kHz.

Reference output corresponds to the desired frequency (4 ms – 250 Hz, 500 us – 2 kHz)
Masking
Masking could be done manually in the “TDC data acquisition” tab of the communication SW. This could be done to cut the noisy pixels from the output. However, this does not cut the pixels from the output completely and it just assigns ‘0’s to this pixels. Extra care should be taken when unpacking binary data. That’s why in the codes provided in these instructions ‘-1’ are assigned to the invalid timestamps (with the most significant bit, 31st, set to 0), otherwise masked and invalid timestamps get mixed up.

In the example on the right, pixels 0, 1, 2, 6, 7, 8, 9, 10 are masked.
Instead of masking all the necessary pixels by hand, it could be done using the SW settings, see the “Debug” tab. In the example on the right, all pixels in regions 2, 3, and 4 are masked (pixels 64-255) and pixels 60-63 (numbering pixels starts from 0 in the SW).

In python, using bin(X), where ‘X’ is the number assigned to the given Mask in the settings, reveals:
1) For Mask0, bin(17293822569102704640) = 0b1111000000000000000000000000000000000000000000000000000000000000, where 4 higher bits correspond to pixels 60-63, and the 60 zeroes to the rest of the pixels (‘1’ equals masking the pixel).
2) For Mask1,2,3,4 , bin(18446744073709551615) = 0b1111111111111111111111111111111111111111111111111111111111111111, or 64 ‘ones’ for masking all pixels. This is an efficient way to mask the required pixels in the SW.
Masking

• For daughterboards NL11 (without microlenses) and A5 (with microlenses), all hot and warm pixels are accounted for in both the application for online plotting and in the offline analysis scripts, so no need to masking anything in the communication SW manually.

• While Mask0, Mask1, Mask2, and Mask3 correspond to each of the 4 regions of 64 pixels (that is 0..63, 64..127, 128..191, 192..255), definition of Mask4 is not clear (no documentation for FW2212 available).

• When uploading settings with a new set of masked pixels, click “Reset” below the masking table in the “TDC data acquisition” tab, otherwise you’ll get a combination of old and new masks.
Analysis software
Real-time plotting

Path to datafiles

LinoSPAD2 daughterboard number

Masking of warm/hot pixels
Should be the same as the number set during collecting

Manual pixel masking

Start plotting

5/31/2023 LinoSPAD2 instructions 26
Data analysis
Data analysis

• Github repo for offline analysis: https://github.com/rngKomorebi/LinoSPAD2/tree/develop

• Github repo for app for real-time plotting: https://github.com/rngKomorebi/LinoSPAD2-app/tree/develop
Quick-start
• Power the board (3.3 V power cable)
• Connect the USB to laptop/PC
• Run the communication software (if the SW window does not open: check FW version of the SW or power cycle the board)
• If SW is running ok and the setup is prepared, power the sensor (voltage generator). !!! NEVER POWER THE SENSOR UNDER DIRECT LIGHT !!! Always have either a sensor cover on or cover the whole board/setup
• Run the real-time plotting software
• Set the parameters in the communication and the real-time plotting SW
• If real-time plotting SW crashed, check path to data files and the number of timestamps