The SOI chip readout firmware

Piotr Kapusta

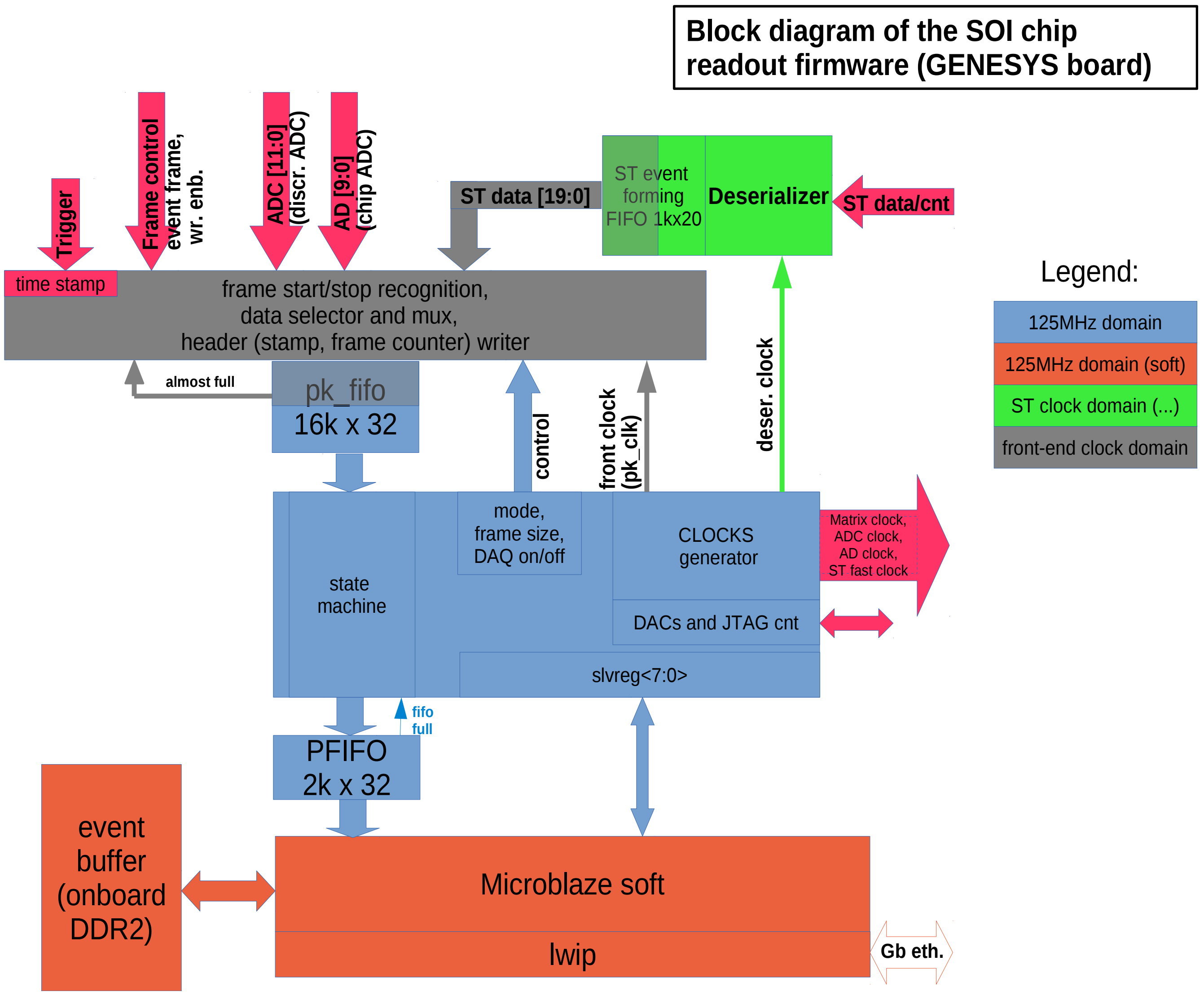
Introduction

The firmware was built basing on the LightWeight IP (lwIP) example application provided by the Digilent company. The project consists of two parts:

- - the Xilinx Virtex V EDK (ver.14.7) design, exploring the *Microblaze* environment and the custom IP core named ***pix_2015_b***, intended to perform all control and readout functions specific to the SOI chip;
- - the SDK design, consisting of C/C++ software for the *Microblaze* processor.

A block diagram of the system is shown in Drawing 1.

Except the brick-colored ones, all other blocks belong to the ***pix_2015_b core***. The whole Xilinx project consists of several other IPs, mostly devoted to the *Microblaze* processor operation. A full



Drawing 1: A block diagram of the GENESYS board firmware.

diagram of the design can be easily generated withe the EDK software; however, it is of little value concerning the understandability of the subject.

The pix_2015_b IP

Events ("pixel frames") arriving from the SOI chip are stored in the front-end fifo buffer (*pk_fifo*). The simple state machine passes the data from the *pk_fifo* to the packet-fifo PFIFO (2k x 32), connecting the ***pix_2015_b*** IP with the *Microblaze* (the PFIFO output port is mapped into the *Microblaze* address space*)*. The latter one pools on the PFIFO status, then reads and stores the events in a huge buffer implemented in the external DDR2 memory. From there events are transferred outside via the Gbit ethernet link.

There are four data sources in the SOI chip:

- MX matrix data, obtained with the external ADC,with event size of 576 words (36 rows x 16 columns) and words width of 12bits;
- Y matrix data, obtained with the external ADC, with event size of 128 words of 12 bits;
- chip ADC data, with 10-bits wide words, without defined frame structure;
- the Self Triggering (ST) matrix data, without defined frame structure, sent serially out from the chip. Data words are 20-bits wide (10 bits for position, 10 bits for time).

The MX and Y matrices data is enveloped with the event frame signals. In case of the chip ADC, the frame signal is generated using the counter in the ***pix_2015_b*** IP; a predefined event size is taken to be 1024. A situation with the ST matrix is more complicated. Here the randomly arriving serial data needs to be deserialized in the circuitry (light-green part) running with clock different (intended to be much faster ...) then the one used in the rest of the front-end, after which the data must be passed to the front-end clock domain (gray part). This is accomplished with the help of the additional fifo buffer (FIFO 1k x 20). Using the programmable fifo threshold feature the relevant event frame signal is generated, with the assumed event size of 128 words.

Since the front-end fifo is 32-bits wide, the MX, Y and chip ADC data needs to be multiplexed before storing. The ST data passes directly to the pk_fifo; upper 12 bits can be filled with the corresponding actual time stamp values [????].

Following counters are implemented inside the front-end part of the ***pix_2015_b*** (the gray block):

- - the chip ADC data counter for artificial event frame generation;
- - the time stamp counter, wide for 64 bits, driven by the trigger clock and trigger reset signals;
- - the frame counter, counting subsequent frames. The arriving frame may not be written – this happens when the front end fifo (pk-fifo) is almost full (there is no space for the <u>whole</u> frame). Then the whole frame is skipped. The frame counter is <u>always</u> incremented.

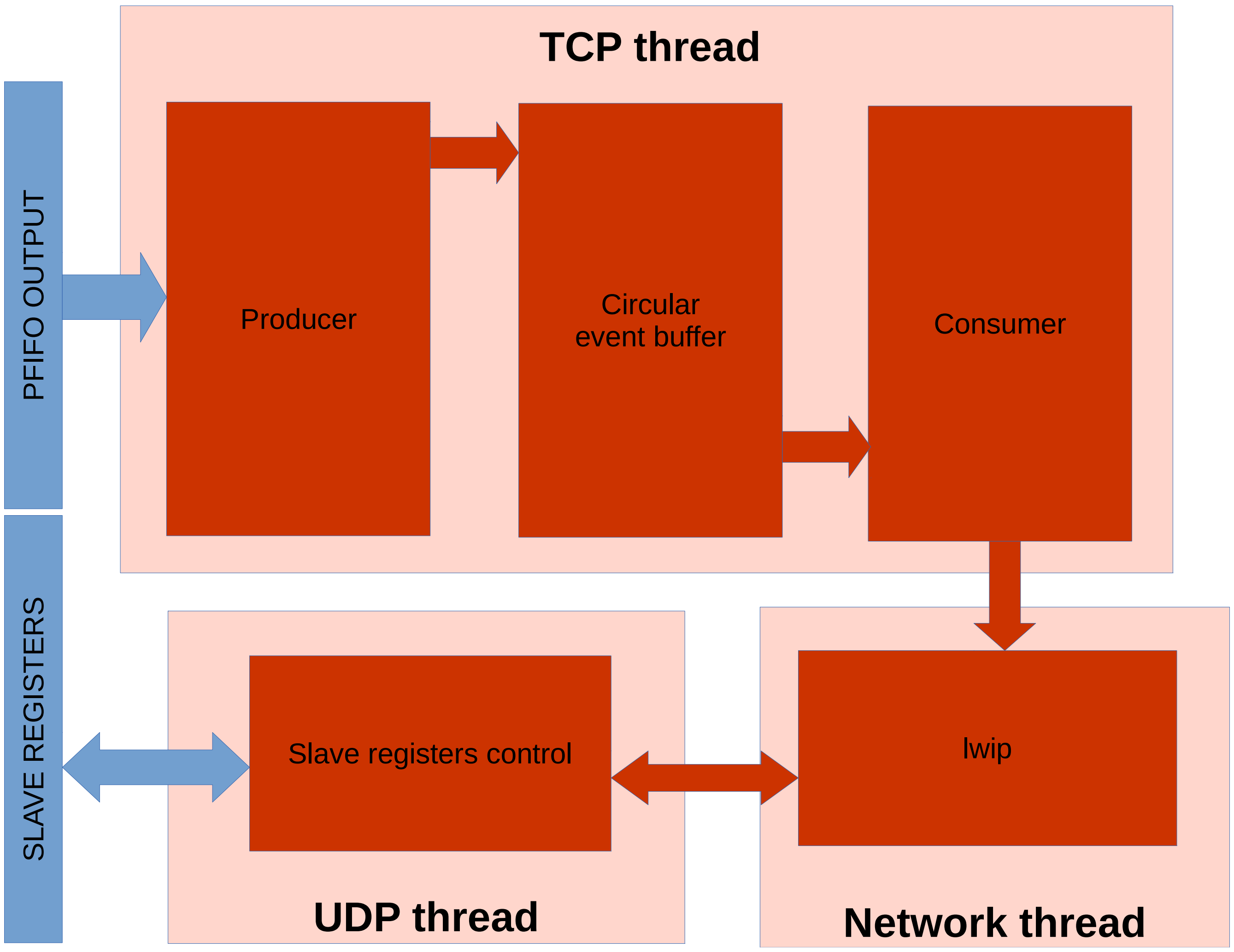
The front-end part of the ***pix_2015_b*** IP runs with the clock (the front-end clock, pk_clk) of a frequency obtained from division of the main system clock, defined as 125MHz. The matrices readout is performed with the same clock; however, [perhaps], [some kind of] a phase shift is necessary to guarantee an error-free operation when the [long] cables between the GENESYS and chip boards are used. This issue is still under investigation.

All ***pix_2015_b*** functions are controlled by means of 8 32-bit wide registers (slvreg[7:0]), mapped into the *Microblaze* space. The Microblaze can both write and read the registers; during the read operation a meaning of the particular bits can be different then in the write one.

The external serial DACs and chip JTAG signals are mapped directly into the slave registers signals (see the table). The *Microblaze* controls both serial interfaces by writing/reading relevant bits in the registers; the protocols rules are not programmed into the ***pix_2015_b***.

The Microblaze software

The *Microblaze* software operates (a development version) as a *xilkernel* application. This allows to implement a multi-threading structure with a large circular event buffer (in the external 256 MB of DDR2 memory). The buffer is controlled by the write and read pointers together with the slot flags register, containing an information about slots occupancy. A structure of the software is shown in Drawing 2. The Producer checks continuously the state of the PFIFO. When a whole frame presence is detected, the frame data is transferred to the event buffer, but only if the relevant buffer slot, pointed to by the write pointer, is free. The write pointer is incremented. On the other side , the Consumer pools on the flag pointed by the read pointer, and if the slot is full, the data is sent oout, the slot is marked as free and the readout pointer is incremented.



Drawing 2: A structure of the Microblaze software.

Slave registers of the pix_2015 core

Bits in the *pix_2015_b* core are inverted with respect to the *Microblaze* core – bit 0 in the pix_2015 core corresponds to the bit 31 in the *Microblaze*.

slv_reg0

Bits pix	**Bits MB**	function
28:31	**3:0**	R/W: reserved
24:27	**7:4**	R/W: Mode of operation
11:23	**20:8**	R/W: reserved
0:10	**31:21**	R/W: PIX_FRAME_SIZE

Modes of operation (bits 27:24 of the pix core):

1000	BIG MATRIX DAQ
0100	Y MATRIX DAQ
0010	ST MATRIX DAQ
0001	ADC DAQ
1111	Test mode – data from internal pattern generator
1110	Test mode – data from external ADC

slv_reg1

Bits pix	**Bits MB**	function
31	**0**	R/W: JTAG TDI, output data to the chip =>
30	**1**	R/W: JTAG TCK =>
29	**2**	R/W: JTAG TMS =>
28	**3**	R/W: Serial data to the ST Matrix control, output to the chip =>
27	**4**	R/W: Serial clock to the ST Matrix control =>
26	**5**	R: JTAG TDO, input data from the chip <=
25	**6**	R: Serial data from the ST Matrix control <=
24	**7**	R: Serial data from the biasing DACs; input from the chips (not usable) <=
23	**8**	R/W: Serial data to the biasing DACs, output data to the chips =>
22	**9**	R/W: Serial clock to the biasing DACs =>
21	**10**	R/W: Synchronization frame to the biasing DACs =>
0:20	**31:11**	R/W: reserved

slv_reg2

Bits pix	**Bits MB**	function
0	**31**	R/W: DAQ ON (results in removing relevant matrix reset)
1	**30**	R/W: Use Trigger
2:31	**29:0**	R/W: reserved

slv_reg3

Bits pix	**Bits MB**	function
0	**31**	R/W: Reset of the front-end fifo (PIX_FIFO)
1	**30**	R/W: Fast reset – force high

2	**29**	R/W: Fast reset – force low
3	**28**	R/W: Front-end general reset
4:28	**27:3**	R/W: reserved
29	**2**	R: PIX_FIFO_EMPTY
30	**1**	R: PIX_FIFO ALMOST FULL
31	**0**	R: PIX_FIFO FULL

slv_reg4

Bits pix	**Bits MB**	function
24:31	**7:0**	R/W: front-end clock (pk_clk) counter limit

Bits pix	**Bits MB**	function
16:23	**15:8**	R/W: Integration Clock (for Y Matrix) counter limit, reset value: 0x0F
13:15	**18:16**	R/W: reserved
12	**19**	R/W: Invert external ADC clock
8:11	**23:20**	R/W: ADC pipeline delay, reset value: “0110” (6 stages)

slv_reg5

Bits	Bits	function
31:0	0:31	R/W: reserved

slv_reg6

Bits pix	**Bits MB**	function
31	**0**	R: state of the “fastblock” signal
1:30	30:1	reserved

slv_reg7

Bits pix	**Bits MB**	function
31	**0**	R/W: DCM PSEN
30	**1**	R/W: DCM PSINCDEC
29	**2**	R/W: DCM_1 PSEN

28	**3**	R/W: DCM_1 PSINCDEC
27	**4**	R/W: DCM RESET
26	**5**	R/W: DCM_1 RESET
4:25	**27:6**	R/W: reserved
3	**28**	R/W: DCM_1 phase done (READ); DCM_1 phase done reset (WRITE)
2	**29**	R/W: DCM phase done (READ) ; DCM phase done reset (WRITE)
1	**30**	R: DCM_1 locked (READ)
0	**31**	R: DCM locked (READ)

Registers of the GENESYS SOI firmware, accessible via UDP

(under construction)

0	BASE+0	RUN CONTROL	Direct access to the **slv_reg0**
1	BASE+0x04	reserved	
2	BASE+0x08	DAQ CONTROL	Direct access to the **slv_reg2** ; DAQ_ON
3	BASE+0x0C	CLOCK CONTROL	Direct access to the **slv_reg4**
4	BASE+0x10	reserved	Serial control of the ST Matrix, to be defined
5	BASE+0x14	DACS CONTROL	See below
6	BASE+0x18	JTAG CONTROL	See below
7	BASE+0x1C	CLOCK PHASEs	See below

REGISTER 0: RUN CONTROL

PIX_FRAME_SIZE	reserved	MODE	reserved
31:21	20:8	7:4	3:0

REGISTER 2: DAQ CONTROL

DAQ ON	reserved
31	30:0

Writing to this register with DAQ ON = 0 (stopping a run) results in F & P fifos reset.

REGISTER 3: CLOCKS CONTROL

Format of the word used to set the clocks :

pk_clk	reserved	int_clk (Y)	reserved	pipe	reserved
31:28	27:24	23:16	15:12	11:8	7:0

Writing to this register results in F & P fifos reset.

REGISTER 5: DACS CONTROL

Format of the word used to set DACs pair:

UPDATE	CHANNEL	reserved	DAC A	DAC B
31	30:28	27:24	23:12	11:0

Where assignmnent of the DACS channels is listed in the following table (to be checked !!!):

CH	DAC A	bias		DAC B	bias	
0	vdac0	I_BUG_PIX_FT	100u	vdac8	vdiscr_n	1.1V (?)
1	vdac1	I_BUG_COL_FT	10u	vdac9	vdiscr_p	1.3V (?)
2	vdac2	I_YBUFF_FT	200u	vdac10	I_DISCRIM	20u (?)
3	vdac3	I_PIX_FT		vdac11	vrst_pix	0.9V

4	vdac4	0-> MX, 2V->Y	0V	vdac12	adc_ref	1.6V
5	vdac5	spare		vdac13	adc_vcm	1.1V
6	vdac6	spare		vdac14	vcmh_ref	0.5V
7	vdac7	ADC mode	1.1V	vdac15	vcm_ref	1.1V

REGISTER 6: JTAG CONTROL

Format of the word used to access JTAG interface:

R/W	reserved	register	register data
31	30:16	15:8	7:0

Register	access	
0x00	RW	UID register
0x01	RW	CONTROL (Y2014, dummy in the 2015 version)
0x02	W	DAC0; preamplifier bias; def. 10u
0x03	W	DAC1; pdbias; def. 5u
0x04	W	DAC2; shabias; def 1u
0x05	W	DAC3; vhbias; def 35u ; to jest chyba AB
0x06	W	DAC4; folded cascode bias; def 30u;
0x07	W	GLB0, dummy in the 2015 version
0x08	W	GLB1, dummy in the 2015 version

On read the stored serial data is produced.

REGISTER 7: CLOCKS PHASES

Format of the word used to shift phases (write format):

DCM phase shift	DCM_1 phase shift	MSHIFT	reserved	DCM_1 U/D	DCM U/D
31:24	23:16	15:6	5:2	1	0

Value of the phase shift corresponds to a number of pulses sent to the DCM.