Specification of firmware for Lagrange Heavy Ion Telescope submodule

Revision: 0.3, modified: 13/12/2020

Table of Contents

1. Introduction....4
2. Firmware structure....4
3. Implementation remarks....4
4. Interface....5
5. SPI communication....7
6. Registers....11
7. Triggering scheme....20
8. Readout modes....21
    8.1. Full readout mode....22
    8.2. Region-of-interest (ROI) readout mode....24
    8.3. Smart readout mode....26
9. Trigger counters....28
10. Operation examples....29
11. Example readouts....32
12. Document history....34

List of Tables

Table 1. List of interface signals......6
Table 2. SPI write command......7
Table 3. SPI read command......7
Table 4. SPI write operation diagram......8
Table 5. SPI read operation diagram......8
Table 6. SPI burst read operation diagram......9
Table 7. Structure of data from FIFO......10
Table 8. List of registers......13
Table 9. Bits definition in status register......14
Table 10. Temperature readout examples......15
Table 11. List of changes in this specification document......34

List of Figures

Figure 1. Firmware structure......4
Figure 2. Full readout mode, the whole sequence [Modelsim simulation, 100us/div]......22
Figure 3. Full readout mode, beginning after trigger sequence [Modelsim simulation, 200ns/div]. 22
Figure 4. Full readout mode, beginning of first channel sequence [Modelsim simulation, 200ns/div]......23
Figure 5. Region-of-interest (ROI) readout mode [from DRS4 datasheet]......24
Figure 6. Region-of-interest readout mode, the whole sequence [Modelsim simulation, 10us/div].24
Figure 7. Region-of-interest readout mode, waiting 100 cycles of fast clock with dwrite [Modelsim simulation, 200ns/div]......25
Figure 8. Region-of-interest readout mode, readout of 1st channel [Modelsim simulation, 1us/div]25
Figure 9. Smart readout mode, the whole sequence [Modelsim simulation, 25us/div]......26
Figure 10. Smart readout mode, addressing the starting cell [Modelsim simulation, 10us/div]......27
Figure 11. Smart readout mode, readout of 1st channel [Modelsim simulation, 2us/div]......27
Figure 12. Example of the readout of test data from one channel in full readout mode......32
Figure 13. Example of the readout of noise from one channel in full readout mode......32
Figure 14. Example of the readout of 100MHz sine generator data from one channel in full readout mode......33
Figure 15. Example of the readout of 100MHz sine generator data from one channel in full readout mode, zoomed in......33

1. Introduction

This specification describes the firmware designed for Lagrange Heavy Ion Telescope submodule, revision 1.66.

2. Firmware structure

The firmware structure is presented on Figure 1.

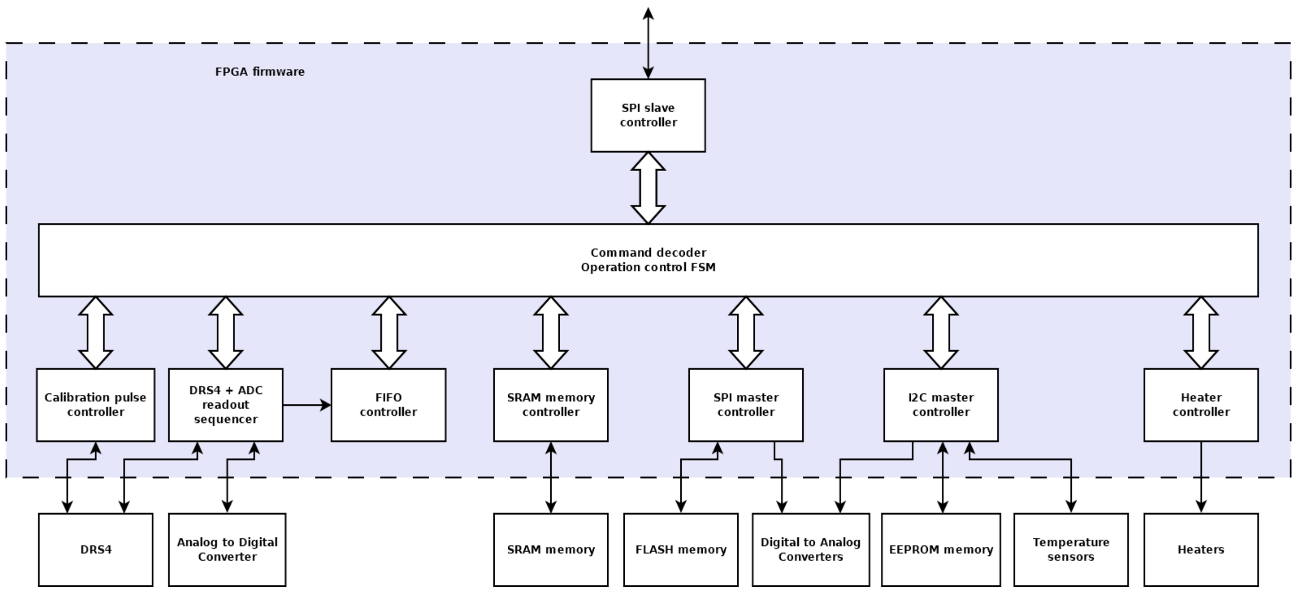

Figure 1. Firmware structure

FLASH memory and EEPROM memory are not implemented.

3. Implementation remarks

Firmware was designed using VHDL language and implemented with usage of following software:

- Microsemi Libero 11.9.5.5
- Synopsys Synplify Pro L-2016.09M-SP1-5
- ModelSim Microsemi 10.5c (for simulations)

Target device for designed firmware is Microsemi ProASIC3 chip, model: A3P1000-FGG256I.

4. Interface

The interface signals are presented in Table 1.

Signal name	Direction	Type	Comment
System signals			
sys_clk	in	std_logic	system clock, 40MHz
sig	inout	std_logic_vector(18 downto 1)	debug connector
ctrl	out	std_logic	led
RM central unit			
cu_i2c_scl	in	std_logic	I2C clock, not used
cu_i2c_sda	inout	std_logic	I2C data, not used
cu_sclk	in	std_logic	SPI clock
cu_smiso	out	std_logic	SPI data input
cu_scs_b	in	std_logic	SPI chip select
cu_smosi	in	std_logic	SPI data output
cu_spi_reset	in	std_logic	SPI reset
SRAM memory			
sram_data	inout	std_logic_vector(15 downto 0)	SRAM data
sram_address	out	std_logic_vector(21 downto 0)	SRAM address
sram_we_n	out	std_logic	SRAM write enable
sram_oe_n	out	std_logic	SRAM output enable
sram_ce1_n	out	std_logic	SRAM chip enable 1
sram_ce2	out	std_logic	SRAM chip enable 2
sram_bhe_n	out	std_logic	SRAM byte high enable
sram_ble_n	out	std_logic	SRAM byte low enable
SPI for Flash memory & DACs			
dac_cs_n	out	std_logic_vector(3 downto 1)	DACs chip select
eecs_n	out	std_logic	EEPROM chip select
sdi	out	std_logic	SPI data output
sclk	out	std_logic	SPI clock
sdo	in	std_logic	SPI data input

Signal name	**Direction**	**Type**	**Comment**
I2C for temperature sensors & HV power control DACs			
i2c_0_sda	inout	std_logic	I2C data
i2c_0_scl	out	std_logic	I2C clock
Heaters			
heat_b	out	std_logic_vector(7 downto 0)	Switches for heaters
Input buffer			
trig	in	std_logic_vector(7 downto 0)	Trigger inputs from discriminators
ADC			
adc_clk	out	std_logic	ADC clock
adc	in	std_logic_vector(13 downto 0)	ADC data
adc_otr	in	std_logic	ADC out of range
Calibration			
cal_inv	out	std_logic	enable 100MHz calibration
tca_ctrl	out	std_logic	enable calibration
DRS4			
dwrite	out	std_logic	DRS4 domino write input
denable	out	std_logic	DRS4 domino enable inpout
a	out	std_logic_vector(3 downto 0)	DRS4 address bit inputs
rsrload	out	std_logic	DRS4 read shift register load input
srin	out	std_logic	DRS4 shared shift register input
srclk	out	std_logic	DRS4 muliplexed shift register clock input
srout	in	std_logic	DRS multiplexed shift register output
wsrin	out	std_logic	DRS4 write shift register input
wsrout	in	std_logic	DRS4 write/read shift register output
plllck	in	std_logic	DRS4 PLL lock indicator output
refclk_p	out	std_logic	DRS4 reference clock input
refclk_m	out	std_logic	DRS4 reference clock input
reset_n	out	std_logic	DRS4 reset input

Table 1. List of interface signals

5. SPI communication

As the communication protocol Serial Peripheral Interface (SPI) is used. The parameters are:

- Speed: 1Mbits
- Data length: 16 bits
- Clock polarity mode: 0
- Clock phase mode: 0

Tables 4, 5 and 6 present diagrams of implemented SPI operations: write, read, read burst. Signals on diagrams:

- CS# – not chip select: cu_scs_b from the interface
- CLK – clock: cu_sclk from the interface
- SDI – serial data input: cu_smosi from the interface
- SDO – serial data output: cu_smiso from the interface

Structures of write and read commands with included register address are presented in Tables 2 and 3.

Structure of data readout from FIFO is presented in Table 7.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF				Register address								0xF			

Table 2. SPI write command

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0				Register address								0x0			

Table 3. SPI read command

CS#	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	X
SDO	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		Command and address to FPGA																**Data to FPGA**																

Table 4. SPI write operation diagram

CS#	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDO	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		Command and address to FPGA																**Empty word**															

CS#	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDO	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	X
	Data from FPGA																

Table 5. SPI read operation diagram

CS#	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDO	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
		Command and address to FPGA																**Empty word**															

CS#	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDO	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Data from FPGA, word: 1																**Data from FPGA, next words**															

CS#	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
CLK	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**	**0→1**
SDI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SDO	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	X
	Data from FPGA, last word																

Table 6. SPI burst read operation diagram

Word	**Bits**															
	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	Timestamp, 16 most significant bits of timestamp															
3	Timestamp, 16 least significant bits															
4	1	1	1	1	1	1	1	1	1	1	1	1	First channel number			
5	0	First channel, first cell														
...	0	...														
...	0	First channel, last cell														
...	1	1	1	1	1	1	1	1	1	1	1	1	Next channel number			
...	0	Next channel, first cell														
...	0	...														
...	0	Next channel, last cell														
...	1	1	1	1	1	1	1	1	1	1	1	1	Last channel number			
...	0	Last channel, first cell														
...	0	...														
...	0	Last channel, last cell														

Table 7. Structure of data from FIFO

6. Registers

The registers implemented in firmware, default values and applicable operations are presented in Table 8, followed by registers descriptions.

Register name	Register address	Operation	Default
constant	0x00	read	0xCAFE
dummy	0x01	read/write	0xC0DE
status	0x02	read	
triggers	0x06	read	external signals
timestamp	0x0E	read	0x0000
operation	0x0F	read/write	0x0004
heaters	0x20	read/write	0x00FF
temp_90	0x21	read	0x0000
temp_92	0x22	read	0x0000
dac_hv1	0x30	read/write	0x0000
dac_hv2	0x31	read/write	0x0000
dac_hv3	0x32	read/write	0x0000
dac_hv4	0x33	read/write	0x0000
dac_hv5	0x34	read/write	0x0000
dac_hv6	0x35	read/write	0x0000
dac_hv7	0x36	read/write	0x0000
dac_hv8	0x37	read/write	0x0000
dac_tlevel1	0x38	read/write	0x0000
dac_tlevel2	0x39	read/write	0x0000
dac_tlevel3	0x3A	read/write	0x0000
dac_tlevel4	0x3B	read/write	0x0000
dac_tlevel5	0x3C	read/write	0x0000
dac_tlevel6	0x3D	read/write	0x0000
dac_tlevel7	0x3E	read/write	0x0000
dac_tlevel8	0x3F	read/write	0x0000
dac_bias	0x40	read/write	0x0000

Register name	**Register address**	**Operation**	**Default**
dac_va	0x41	read/write	0x0000
dac_vb	0x42	read/write	0x0000
dac_rofs	0x4B	read/write	0x0000
dac_calp	0x4C	read/write	0x0000
dac_calm	0x4D	read/write	0x0000
calibration	0x4E	read/write	0x0000
drs_config	0x4F	read/write	0x0000
last_command	0x50	read	0x0000
last_data	0x51	read	0x0000
command_count	0x52	read	0x0000
sram_data	0x60	read/write	0x0000
sram_data_inc	0x61	read/write	0x0000
sram_address_row	0x62	read/write	0x0000
sram_address_col	0x63	read/write	0x0000
sram_demo_data	0x6E	read/write	0x0000
sram_burst_length	0x6E	read/write	0x000A
sram_burst_data	0x6F	write	0x0000
fifo	0x70	read	0x0000
fifo_burst	0x71	read	0x000A
fifo_burst_length	0x72	read/write	0x0000
readout_mode	0x80	read/write	0x0003
readout_channels	0x81	read/write	0x01FF
readout_delay	0x82	read/write	0x0000
readout_length	0x83	read/write	0x0000
readout_start	0x84	read/write	0x0000
refclk_half_period	0x8F	read/write	0x0014
counter_pattern_0	0xD0	read	0x0000
counter_pattern_1	0xD1	read	0x0000
counter_pattern_2	0xD2	read	0x0000
counter_pattern_3	0xD3	read	0x0000
counter_pattern_4	0xD4	read	0x0000
counter_pattern_5	0xD5	read	0x0000

Register name	Register address	Operation	Default
counter_pattern_6	0xD6	read	0x0000
counter_pattern_7	0xD7	read	0x0000
counter_trigger_0	0xD8	read	0x0000
counter_trigger_1	0xD9	read	0x0000
counter_trigger_2	0xDA	read	0x0000
counter_trigger_3	0xDB	read	0x0000
counter_trigger_4	0xDC	read	0x0000
counter_trigger_5	0xDD	read	0x0000
counter_trigger_6	0xDE	read	0x0000
counter_trigger_7	0xDF	read	0x0000
pattern_0	0xE0	read/write	0x00FF
pattern_1	0xE1	read/write	0x00FF
pattern_2	0xE2	read/write	0x00FF
pattern_3	0xE3	read/write	0x00FF
pattern_4	0xE4	read/write	0x00FF
pattern_5	0xE5	read/write	0x00FF
pattern_6	0xE6	read/write	0x00FF
pattern_7	0xE7	read/write	0x00FF
debug_1	0xF1	read	depends on application
debug_2	0xF2	read	depends on application
debug	0xFF	read	depends on application

Table 8. List of registers

Register *constant:*

Contains constant value of 0xCAFE.

Register *dummy*:

Register to test write and read operations.

Register *status*:

Register contains most important information about the operation. Bits definition in status register are presented in Table 9.

Bit number	**Name**	**Description**
15	plllck	Signal indicating PLL in DRS4 is locked
14	fifo_full	Signal indicating FIFO full
13	fifo_empty	Signal indicating FIFO empty
12	dwrite_int	Signal dwrite from register
11	denable_int	Signal denable from register
10	drs_dwrite	Signal dwrite from drs sequencer
9	drs_denable	Signal denable from drs sequencer
8	drs_enable	Signal enable drs sequencer
7	drs_trigger_command	Signal trigger drs sequencer by command
6	cal_inv	Signal for calibration pulse
5	tca_ctrl	Signal for calibration pulse
4	any_trigger_on	Signal indicating any trigger on
3	any_heater_on	Signal indicating any heater on
2	i2c_master_timeout	Signal indicating I2C master timeout
1	i2c_master_busy	Signal indicating I2C master busy
0	spi_master_busy	Signal indicating SPI master busy

Table 9. Bits definition in status register

Register *triggers*:

Register contains the readout of trigger signals from external discriminators.

Register *timestamp*:

Register contains 16 most significant bits of 32-bits timestamp word. Resolution of the internal timestamp is 25ns. Resolution of timestamp in this register is 1,6384ms. Time to overflow of this register is 107,3741824s.

Register *operation*:

Bit 9 controls the trigger by command. Bit 8 controls starting the DRS4 sequencer and enabling the data acquisition. Bit 0 controls denable signal to DRS4 when the sequencer is not enabled.

Register *heaters*:

Register contains the control bits of switches for heaters. Negative logic.

Registers *temp_90, temp_92*:

Registers contain actual readout of temperature sensors. There are two devices connected via I2C bus to FPGA. Both are read out every 0.5 second. The data in each register is in two's complement format, has a data width of 16 bits and a resolution of 7.8125m°C. Some examples of temperature readout and decoding are presented in Table 10.

Temperature readout		**Temperature (°C)**
binary	**hex**	
1111 0011 1000 0000	0xF380	-25
1111 1111 1111 1111	0xFFFF	–0.0078125
0000 0000 0000 0000	0x0000	0
0000 0000 0000 0001	0x0001	0.0078125
0000 1100 1000 0000	0x0C80	25

Table 10. Temperature readout examples

Registers *dac_hv1, dac_hv2, dac_hv3, dac_hv4, dac_hv5, dac_hv6, dac_hv7, dac_hv8*:

Writing to registers starts the programming of proper channel of DAC setting the values of voltages hv1, hv2, hv3, hv4, hv5, hv6, hv7, hv8. 1 bit equal to 50,35uV, 3.3V over 16 bits.

Registers *dac_tlevel1, dac_tlevel2, dac_tlevel3, dac_tlevel4, dac_tlevel5, dac_tlevel6, dac_tlevel7, dac_tlevel8*:

Writing to registers starts the programming of proper channel of DAC setting the values of voltages tlevel1, tlevel2, tlevel3, tlevel4, tlevel5, tlevel6, tlevel7, tlevel8. 1 bit equal to 50,35uV, 3.3V over 16 bits.

Registers *dac_bias, dac_rofs, dac_calp, dac_calm*:

Writing to registers starts the programming of proper channel of DAC setting the values of voltages bias, rofs, calp, calm. 1 bit equal to 50,35uV, 3.3V over 16 bits.

Registers *dac_va, dac_vb*:

Writing to registers starts the programming of proper channel of DAC setting the values of voltages vp, vp. 1 bit equal to 610uV, 2.5V over 12 bits. Not exceed 2.5V!

Register *calibration*:

Bit 1 controls output signal cal_inv, bit 0 controls output signal tca_ctrl. These signals must be high to enable calibration source.

Register *drs_config*:

Bits 15 downto 12 contains address bits to DRS, bit 7 downto 0 contains the data to be sent to DRS4. Writing to the register starts the operation of sending the data to DRS4 chip.

Register *last_command*:

Register contains the last executed command.

Register *last_data*:

Register contains data of the last executed command.

Register *command_count*:

Register contains the number of executed commands.

Register *sram_data*:

Reading from the register starts data read from memory sequence. 1 word is read out from SRAM memory and sent through SPI interface. Before the readout, the registers: *sram_address_row, sram_address_col* must be written to indicate the memory address.

Writing to the register starts data write to memory sequence. 1 word, sent through SPI interface, is written to SRAM memory. Before the write, the registers: *sram_address_row, sram_address_col* must be written to indicate the memory address.

Register *sram_data_inc*:

Reading from the register starts data read from memory sequence. 1 word is read out from SRAM memory and sent through SPI interface. Before the readout, the registers: *sram_address_row, sram_address_col* must be written to indicate the memory address. After the read operation, the memory address is increased.

Writing to the register starts data write to memory sequence. 1 word, sent through SPI interface, is written to SRAM memory. Before the write, the registers: *sram_address_row, sram_address_col* must be written to indicate the memory address. After the write operation, the memory address is increased.

Registers *sram_address_row, sram_address_col*:

Registers contain the address of the current SRAM memory cell, used in write/read operations invoked by accessing other registers.

Register *sram_demo_data*:

Writing to the register starts filling the memory with demo data. 8 MS bits of each memory cell are written with 8 LS bits of the register, while 8 LS bits of memory cell are written with 8 LS bits of cell address.

Register *sram_burst_length*:

Register contains the amount of data to be read out from SRAM memory using register *sram_burst_data*.

Register *sram_burst_data*:

Reading from the register starts the data burst read from memory sequence. The number of words to be read out is defined in register *sram_burst_length.* Words are read out from FIFO and sent through SPI interface. Before the readout, the registers: *sram_address_row, sram_address_col* must be written to indicate starting address.

Register *fifo*:

Reading from the register starts data read from FIFO sequence. 1 word is read out from FIFO and sent through SPI interface. Writing to the register starts data write to FIFO sequence. 1 word, sent through SPI interface, is written to FIFO.

Register *fifo_burst*:

Reading from the register starts the data burst read from FIFO sequence. The number of words to be read out is defined in register *fifo_burst_length*. Words are read out from FIFO and sent through SPI interface.

Register *fifo_burst_length*:

Register contains the amount of data to be read out from FIFO using register *fifo_burst*.

Register *readout_mode*:

Register contains the selected mode of the data readout from DRS4 chip (see Section 8). The 2 least significant bits defines the mode: "01" for region-of-interest readout mode, "10" for smart readout more, "11" for full readout mode. The 3rd least significant bit turns on/off test mode, in which the data written to FIFO is not read out from DRS4 but generated by the counter.

Register *readout_channels*:

The 9 least significant bits define the enabled channels used during the data acquisition.

Register *readout_delay*:

Register contains the number of cycles of fast clock (100MHz) between the trigger and de-assertion of dwrite signal for DRS4 chip in region-of-interest readout mode (see Section 8.2).

Register *readout_length*:

Register contains the number of cells to read out from DRS4 chip in region-of-interest and smart readout modes (see Sections 8.2 and 8.3).

Register *readout_start*:

Register contains the number of cell to start the read out from DRS4 chip in smart readout mode (see Section 8.3).

Register *refclk_half_period:*

Register contains the value of the counter, running with 40MHz frequency, for half of the period of refclk signal. It defines the sampling frequency of DRS. The default value provides refclk frequency of 1MHz, what corresponds to DRS sampling frequency equal to 2.048GHz.

Registers *counter_pattern_0, counter_pattern_1, counter_pattern_2, counter_pattern_3, counter_pattern_4, counter_pattern_5, counter_pattern_6, counter_pattern_7:*

Register contains the values of counters that counts every occurrence of the accepted trigger pattern. For details see Section 9.

Registers *counter_trigger_0, counter_trigger_1, counter_trigger_2, counter_trigger_3, counter_trigger_4, counter_trigger_5, counter_trigger_6, counter_trigger_7*:

Register contains the values of counters that counts every trigger occurrence on trigger input signals from discriminators. For details see Section 9.

Registers pattern_*0*, pattern_*1*, pattern_*2*, pattern_*3*, pattern_*4*, pattern_*5*, pattern_*6*, pattern_*7:*

The masks of trigger signals to start the data acquisition. For details see Section 7.

Registers *debug_1, debug_2, debug*:

Registers usage depends on application. Not used for actual operation. To be removed in final firmware.

7. Triggering scheme

There are two triggering methods implemented:

- by command,
- auto.

Trigger by command is invoked by setting corresponding bit in register *operation*. It starts the operation of data acquisition from DRS4 and storing the readout from ADC in FIFO memory.

Second method is based on trigger signals from discriminators and eight registers: *trigger_0, trigger_1, trigger_2, trigger_3, trigger_4, trigger_5, trigger_6, trigger_7*. These registers work as mask. If the pattern of any of them is equal to trigger signals from discriminators the operation of data acquisition starts.

8. Readout modes

There are three readout modes implemented:

- full readout mode (see Section 8.1),
- region-of-interest (ROI) readout mode (see Section 8.2),
- smart readout mode (see Section 8.3).

First two modes: full readout and region-of-interest are direct implementations of readout modes implemented in DRS4 chips. Further details are described in datasheet of the DRS4 chip.

The smart readout mode is modification of full readout mode.

8.1. Full readout mode

In the full readout mode, all 1024 sampling cells are read out consecutively starting from cell 0 with 1024 clock cycles. After initialization, channels can be read out. When the full readout mode is used the trigger immediately stops the acquisition in DRS4 and the readout sequencer starts to read the data from the chip through ADC. The valid channels are defined by register *readout_channels*. Figures 2, 3 and 4 presents crucial sequences of the full readout simulated in Modelsim.

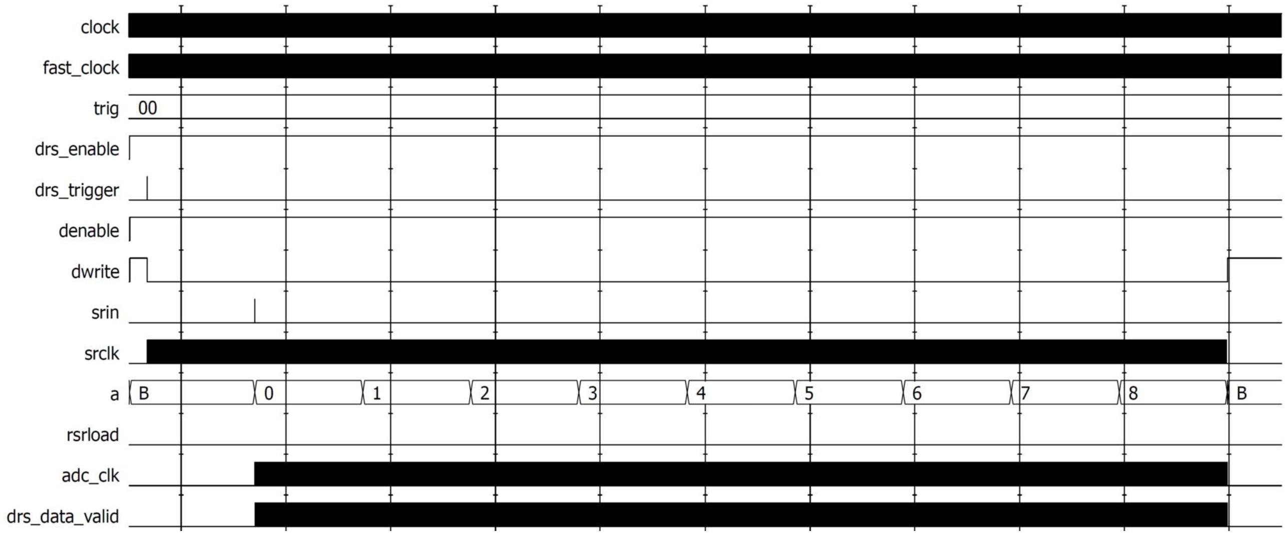

Figure 2. Full readout mode, the whole sequence [Modelsim simulation, 100us/div]

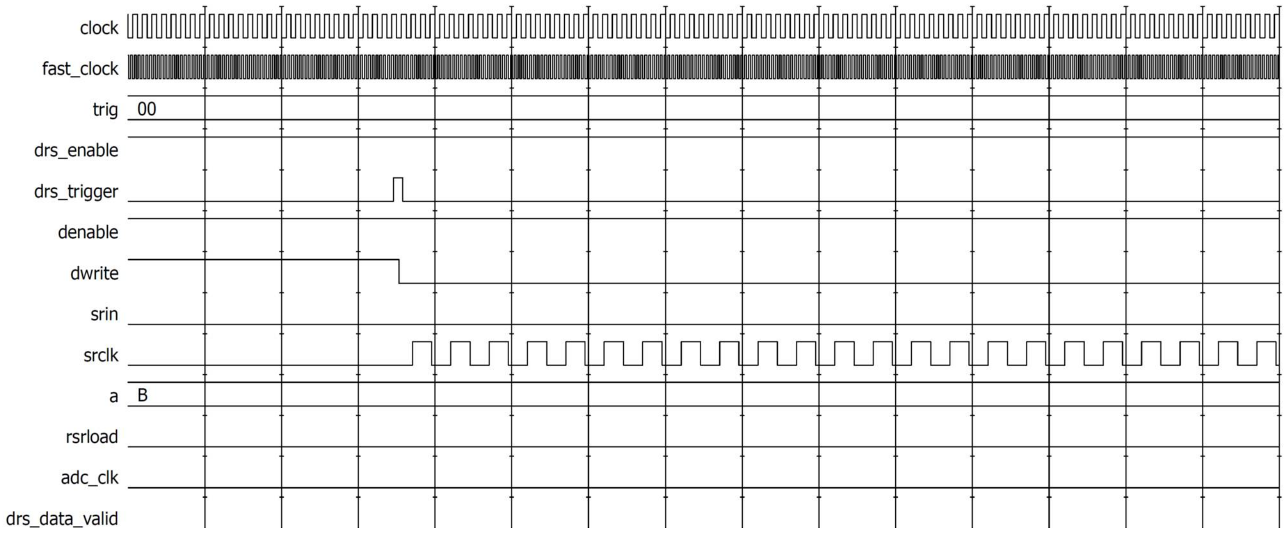

Figure 3. Full readout mode, beginning after trigger sequence [Modelsim simulation, 200ns/div]

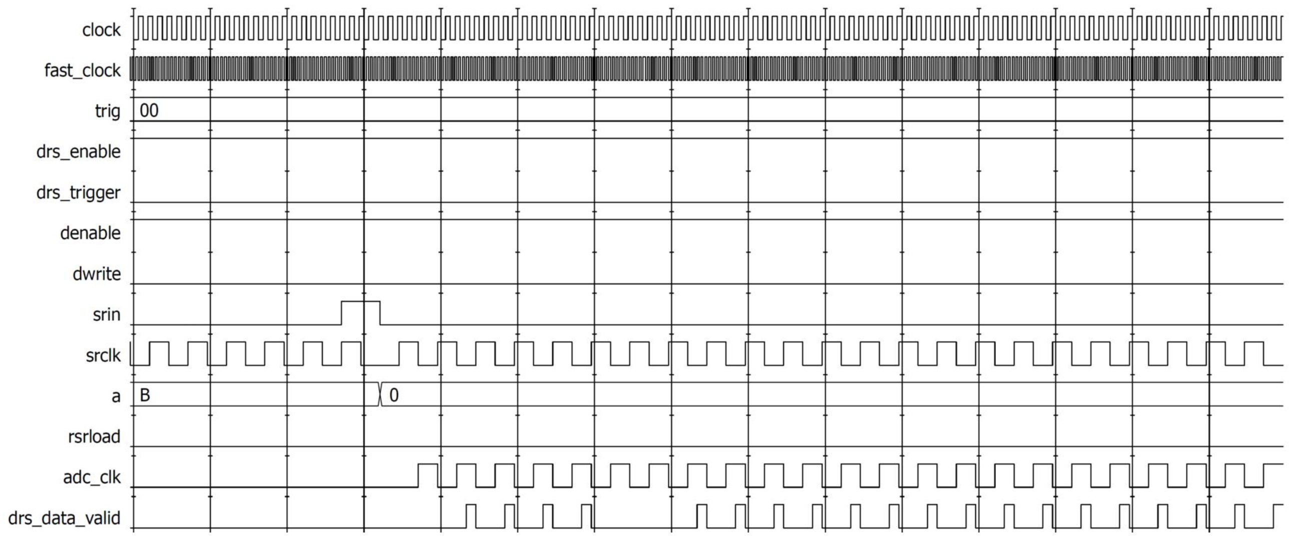

Figure 4. Full readout mode, beginning of first channel sequence [Modelsim simulation, 200ns/div]

8.2. Region-of-interest (ROI) readout mode

For applications where one is interested only in short pulses, to reduce the dead time, it is possible to read only a subset of all sampling cells. The idea of region-of-interest readout is presented on Figure 5. When the region-of-interest readout mode is used the trigger stops the acquisition in DRS4 chip after defined number of fast (100MHz) clock cycles. Then the readout sequencer starts to read defined number of cells from the chip through ADC. The valid channels are defined by register *readout_channels;* the delay between de-asserting of dwrite signal and trigger is defined by register *readout_delay*; the number of cells to be read out is defined by register *readout_length*. Figures 6, 7 and 8 presents crucial sequences of the region-of-interest readout simulated in Modelsim.

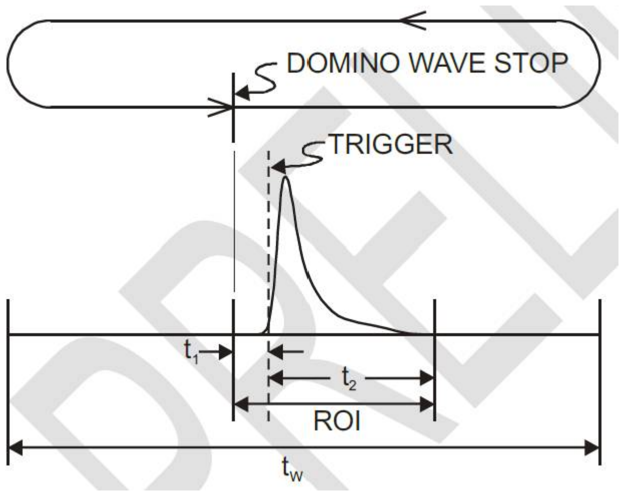

Figure 5. Region-of-interest (ROI) readout mode [from DRS4 datasheet]

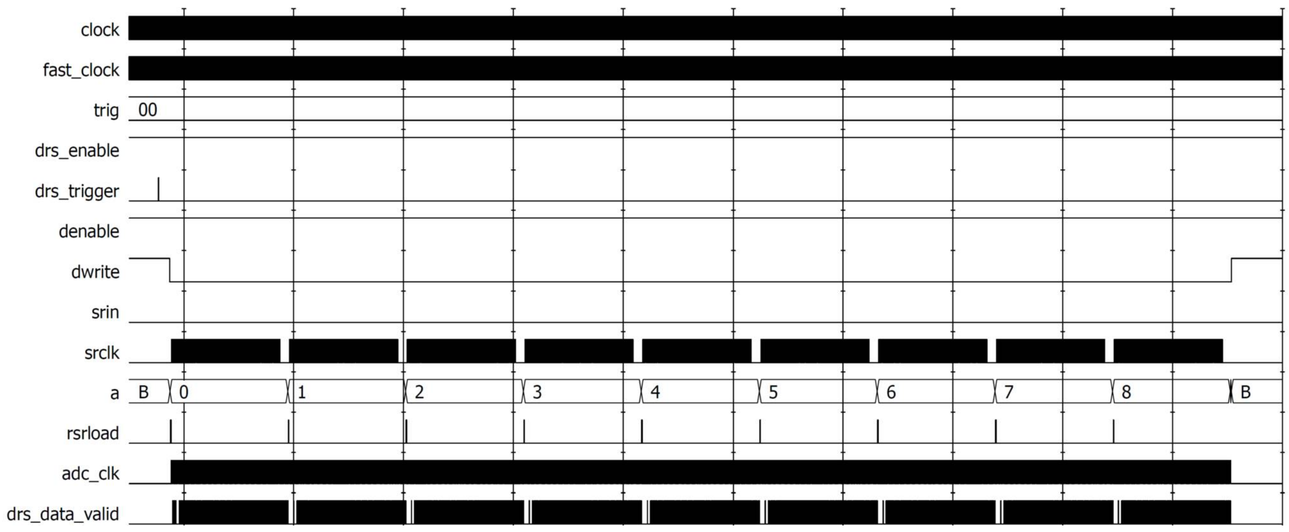

Figure 6. Region-of-interest readout mode, the whole sequence [Modelsim simulation, 10us/div]

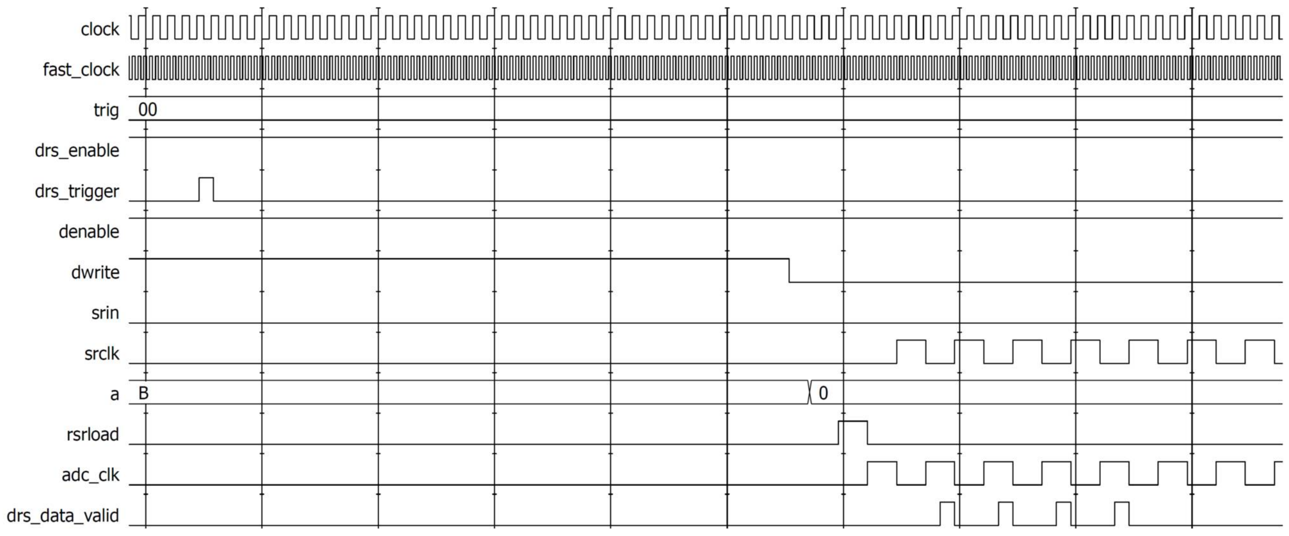

Figure 7. Region-of-interest readout mode, waiting 100 cycles of fast clock with dwrite [Modelsim simulation, 200ns/div]

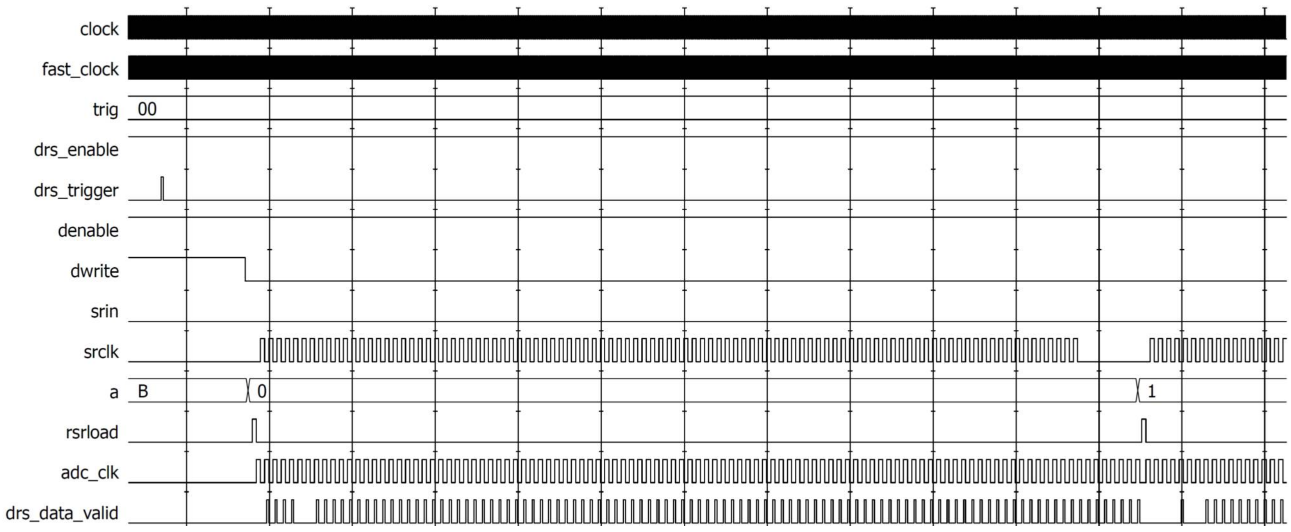

Figure 8. Region-of-interest readout mode, readout of 1st channel [Modelsim simulation, 1us/div]

8.3. Smart readout mode

The smart readout mode is modification of full readout mode in which the readout starts at defined cell of DRS4 chip and ends after defined number of cells. When the smart readout mode is used, after the trigger occurrence, the initialization of DRS4 readout addresses the starting cell. Then the readout sequencer starts to read defined number of cells from the chip through ADC. The valid channels are defined by register *readout_channels;* the starting cell is defined by register *readout_start*; the number of cells to be read out is defined by register *readout_length*. Figures 9, 10 and 11 presents crucial sequences of the region-of-interest readout simulated in Modelsim.

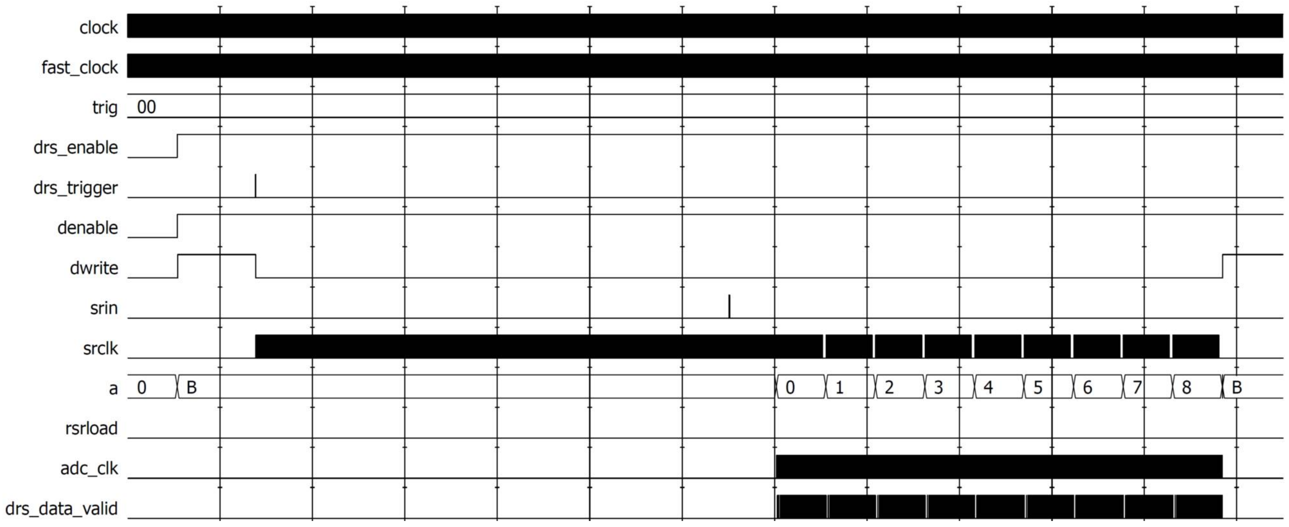

Figure 9. Smart readout mode, the whole sequence [Modelsim simulation, 25us/div]

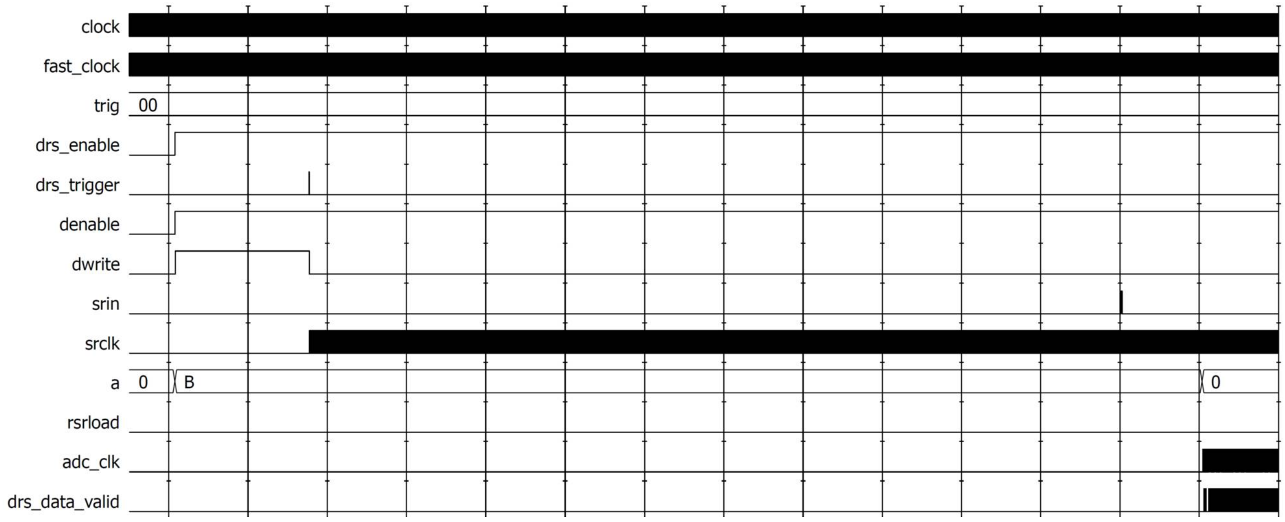

Figure 10. Smart readout mode, addressing the starting cell [Modelsim simulation, 10us/div]

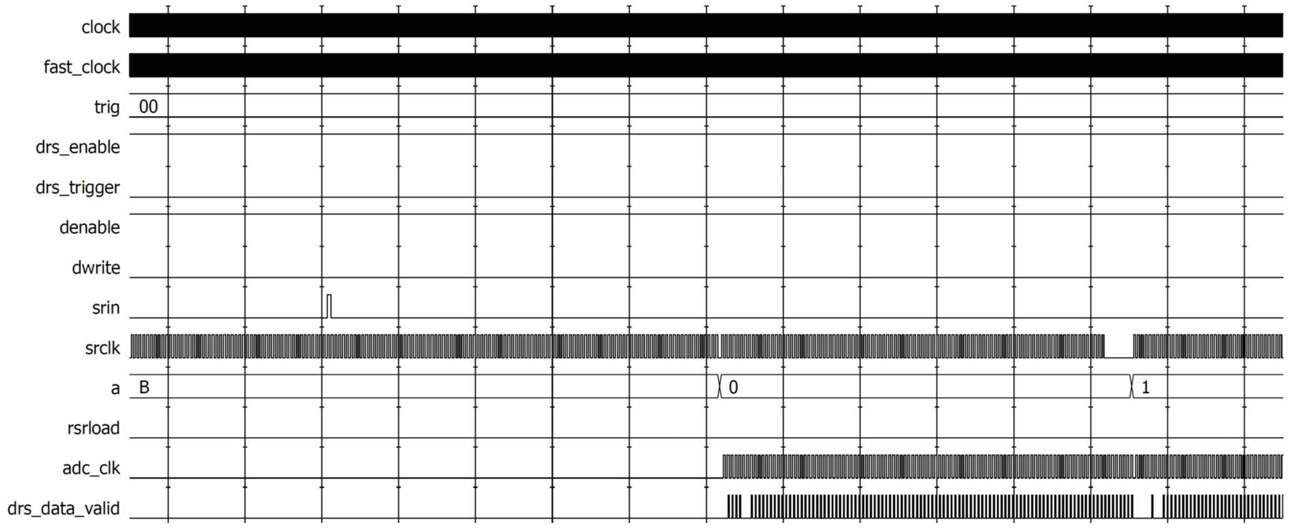

Figure 11. Smart readout mode, readout of 1st channel [Modelsim simulation, 2us/div]

9. Trigger counters

There are two type of counters implemented:

- trigger counters,
- pattern counters.

The trigger counters count every trigger occurrence on trigger input signals from discriminators.

The pattern counters counts every occurrence of the accepted trigger pattern.

10. Operation examples

This Section presents a few examples of basic device operation.

Basic initialization:

```
set register: 0x20 to: 0x000F      // Heaters off
set register: 0x40 to: 0x4E10      // Setting DAC: BIAS to 0.8V
set register: 0x4B to: 0x9FFF      // Setting DAC: ROFS to 1.595V
set register: 0x4F to: 0x0C0F      // Setting DRS4 config register to 0xFF
set register: 0x4F to: 0x0D0F      // Setting DRS4 write shift register to 0xFF
```

Starting calibration source:

```
set register: 0x4D to 0x51EB       // Setting DAC: CAL- to 0.8V
set register: 0x4C to: 0x51EB      // Setting DAC: CAL+ to 0.8V
set register: 0x4E to: 0x0003      // Starting calibration source
```

Stopping calibration source:

```
set register: 0x4E to: 0x0000      // Stopping calibration source
```

Reading the temperatures:

```
get register: 0x21                 // Reading the temperature from I2C device 0x90
get register: 0x22                 // Reading the temperature from I2C device 0x92
```

Reading the status:

```
get register: 0x02                 // Reading the status
```

Reading the info about last command:

```
get register 0x50                  // Reading the last command
get register 0x51                  // Reading the data from last command
```

```
    get register 0x52                    // Reading the commands counter
```

Configuring the full readout mode:

```
    set register: 0x81 to: 0x01FF        // Enabling all channels
    set register: 0x80 to: 0x0003        // Setting the full readout mode
```

Configuring the region-ot-interest readout mode:

```
    set register: 0x80 to: 0x0001        // Setting the full readout mode
    set register: 0x81 to: 0x01FF        // Enabling all channels
    set register: 0x82 to: 0x00C8        // Setting the delay to 200
    set register: 0x83 to: 0x0064        // Setting the readout length to 100
```

Configuring the smart readout mode:

```
    set register: 0x80 to: 0x0002        // Setting the full readout mode
    set register: 0x81 to: 0x01FF        // Enabling all channels
    set register: 0x83 to: 0x0064        // Setting the readout length to 100
    set register: 0x84 to: 0x012C        // Setting the start cell to 300
```

Enable the readout sequencer:

```
    set register: 0x0F to: 0x01A0        // Enable DRS4 sequencer
```

Trigger DRS by command:

```
    set register: 0x0F to 0x03A0         // Enable DRS4 sequencer trigger
    set register: 0x0F to 0x01A0         // Disable DRS4 sequencer trigger
```

Disable the readout sequencer:

```
    set register: 0x0F to: 0x00A0        // Disable DRS4 sequencer
```

Burst read from FIFO:

```
set register: 0x72 to: 0x0064          // Setting the burst readout length to 100
get burst from register: 0x71          // Burst read of100 words from register
```

11. Example readouts

Figures 12, 13, 14 and 15 present example readouts from the device.

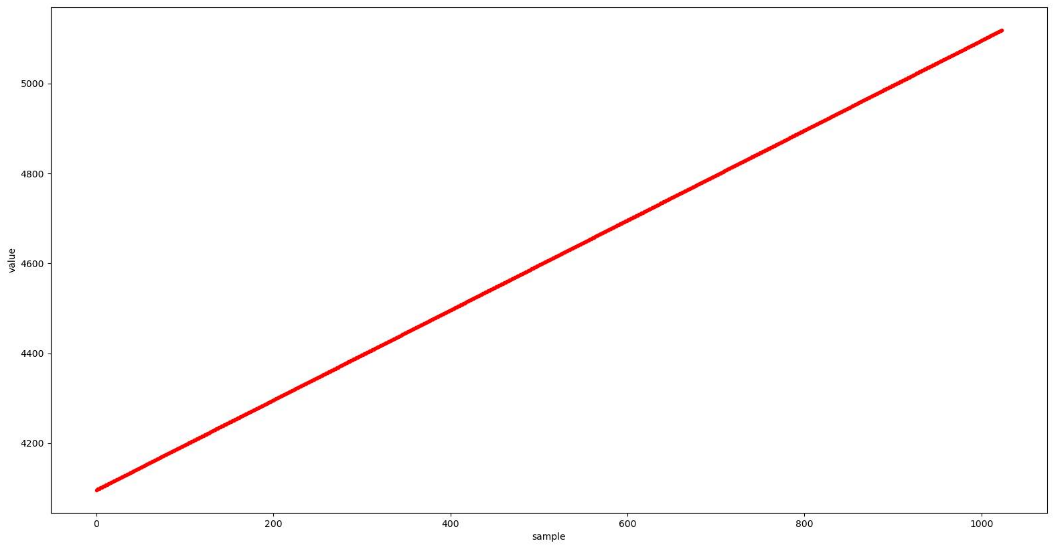

Figure 12. Example of the readout of test data from one channel in full readout mode

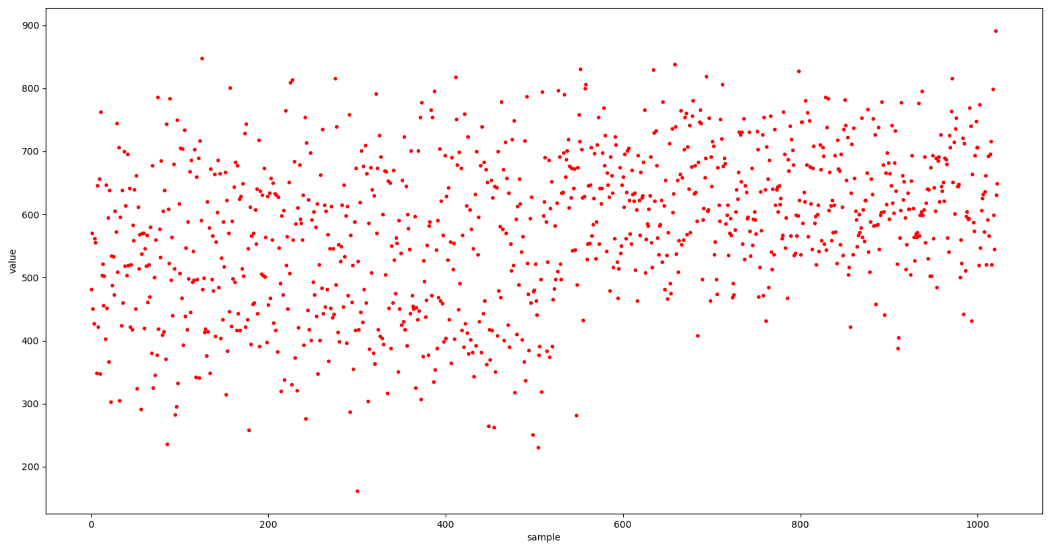

Figure 13. Example of the readout of noise from one channel in full readout mode

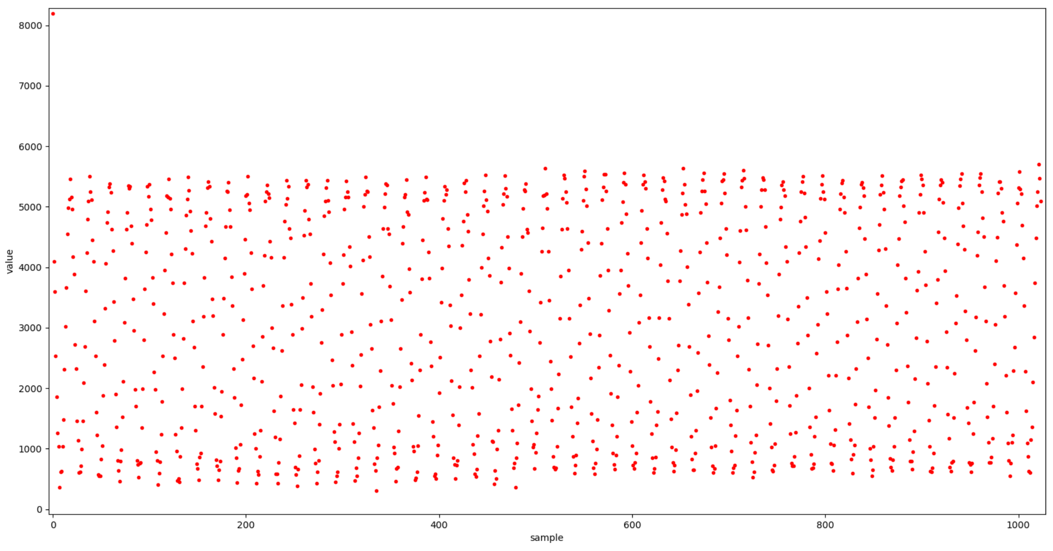

Figure 14. Example of the readout of 100MHz sine generator data from one channel in full readout mode

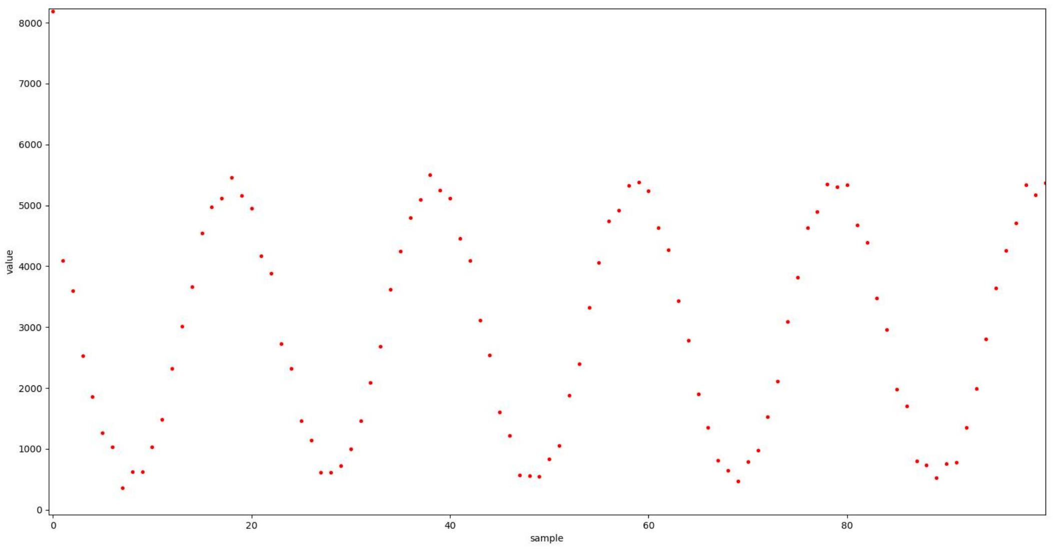

Figure 15. Example of the readout of 100MHz sine generator data from one channel in full readout mode, zoomed in

12. Document history

Revision number	Date	Author	Changes:
0.3	2020.12.06	Dominik Rybka	• Table 7 updated. • Figures 2 - 11 updated. • Description of register *sram_burst_length* added. • Register name changed from *sram_data_burst* to *sram_burst_data*. • Address of register *sram_demo_data* changed from 0x6E to 0x6D. • Description of registers *sram_pointer_read* and *sram_pointer_write* removed. • Section 11 added.
0.2	2020.11.01	Dominik Rybka	• FIFO memory added to Figure 1. • Registers' names *trigger_0, trigger_1, trigger_2, trigger_3, trigger_4, trigger_5, trigger_6, trigger_7* changed to *pattern_0, trigger_1, trigger_2, trigger_3, trigger_4, trigger_5, trigger_6, trigger_7.* • Descriptions of registers: *timestamp, fifo, fifo_burst, fifo_burst_length, readout_mode, readout_channels, readout_delay, readout_length, readout_start, trigger_4, trigger_5, trigger_6, trigger_7, counter_trigger_0, counter_trigger_1, counter_trigger_2, counter_trigger_3, counter_trigger_4, counter_trigger_5, counter_trigger_6, counter_trigger_7, counter_pattern_0, counter_pattern_1, counter_pattern_2, counter_pattern_3, counter_pattern_4, counter_pattern_5, counter_pattern_6, counter_pattern_7* added. Table 8 updated. • Changes in *status* register description in Table 9. • *Sections 8, 8.1, 8.2 and 8.3* added. • Section 9 added. • Section "Events memory mapping" removed. • Section "Measurement results" removed. • Section 10 changed.
0.1	2020.07.23	Dominik Rybka	• Initial version.

Table 11. List of changes in this specification document