Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №23

Системен интегриран логически анализатор (System Integrated Logic Analyzer). Проследяване на сигнали между FPGA модули.

==========

1. Превключете джъмпера вдясно на платката на позиция JTAG. Свържете µUSB кабел към PROG/UART USB куплунга. Включете платката от ключа ON/OFF.

2. Копирайте еталонния дизайн от директория 23_2 на настоящото лабораторно упражнение в работната директория на Vivado и разархивирайте проекта:

```
cp -r /home/user/Desktop/PVAES/23_sila /home/user/workspaces/vivado_workspace
```

3. Стартирайте Vivado от главното меню на Линукс. Отворете еталония дизайн, след това Flow Navigator → Open Block Design.

4. Щракнете последователно с десен бутон върху сигналите на блока abs_gradient_out_v1.0 [1] [2] [3]:
*interrupt_0
*interrupt_1
*gpio_out_0_0
*gpio_out_1_0

и върху магистралата:
S00_AXI

и изберете Debug. Върху съответните проводници ще се появи зелена буболечка.

Забележка: премахването на сигнал/магистрала от Debug става чрез десен бутон върху сигнал/магистрала → Clear Debug.

5. За да е възможно наблюдението на регистрите на персонализираният модул abs_gradient_out, трябва да се модифицира кодът му на Verilog. Копирайте и разархивирайте abs_gradient_out_1_0.zip в работната директория на Vivado. Във Vivado → Flow Navigator → IP Catalog → User Repository → десен бутон → Remove from project → десен бутон върху Vivado Repository → Add Repository → укажете пътя до abs_gradient_out → OK → OK.
В новопоявилата се директория AXI Peripheral → abs_gradient_out_v1.0 → десен бутон → Edit in IP Packager → OK → Sources → Design Sources → отворете категорията abs_gradient_out_v1_0.v (това е top-level описанието) и

щракнете два пъти върху abs_gradient_out_v1_0_S00_AXI.v (това е имплементацията на custom IP модула) → потърсете декларацията на регистрите:

```
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg7;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg8;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg9;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg10;
```

→ добавете текстът [4]:

```
(* dont_touch="true", mark_debug="true" *)
```

пред всеки един регистър. Това ще принуди Vivado да НЕ оптимизира изброените регистри и автоматично да добави логически анализатор, както и да направи връзките между анализатора и регистрите.

Натиснете CTRL + s от клавиатурата, за да запазите Verilog файла.

Таб Package IP – abs_gradient_out → File Groups → Merge changes from File Groups Wizard.

Таб Package IP – abs_gradient_out → Review and Package → бутон Re-Package IP → Yes (това ще затвори второто копие на Vivado, в което се извършва редакцията на IP модула).

7. Обратно в първото копие на Vivado → жълта лента: Show IP Status → Upgrade Selected → OK → Generate → OK.

8. Зелена лента в прозореца Diagram → Designer Assistance Available → щракнете върху Run Connection Automation.

9. От новопоявилият се прозорец Run Connection Automation поставете отметки върху всички сигнали и магистрали → OK.

ВНИМАНИЕ: ако сигналите, които ще се дебъгват са от модули, захранени от тактови сигнали с различна честота, то в блоковата схема ще се включат повече от един блок System ILA.

10. На блоковата схема трябва да се появи блок с име System ILA. Щракнете двукратно върху блока → Sample Data Depth: изберете число, което да е съобразено със свободните ресурси на FPGA за конкретния дизайн (вижте графиката вляво – Resource Estimates) → за конкретния пример 1024 е на границата на възможностите на използваната FPGA → отметка на Advanced

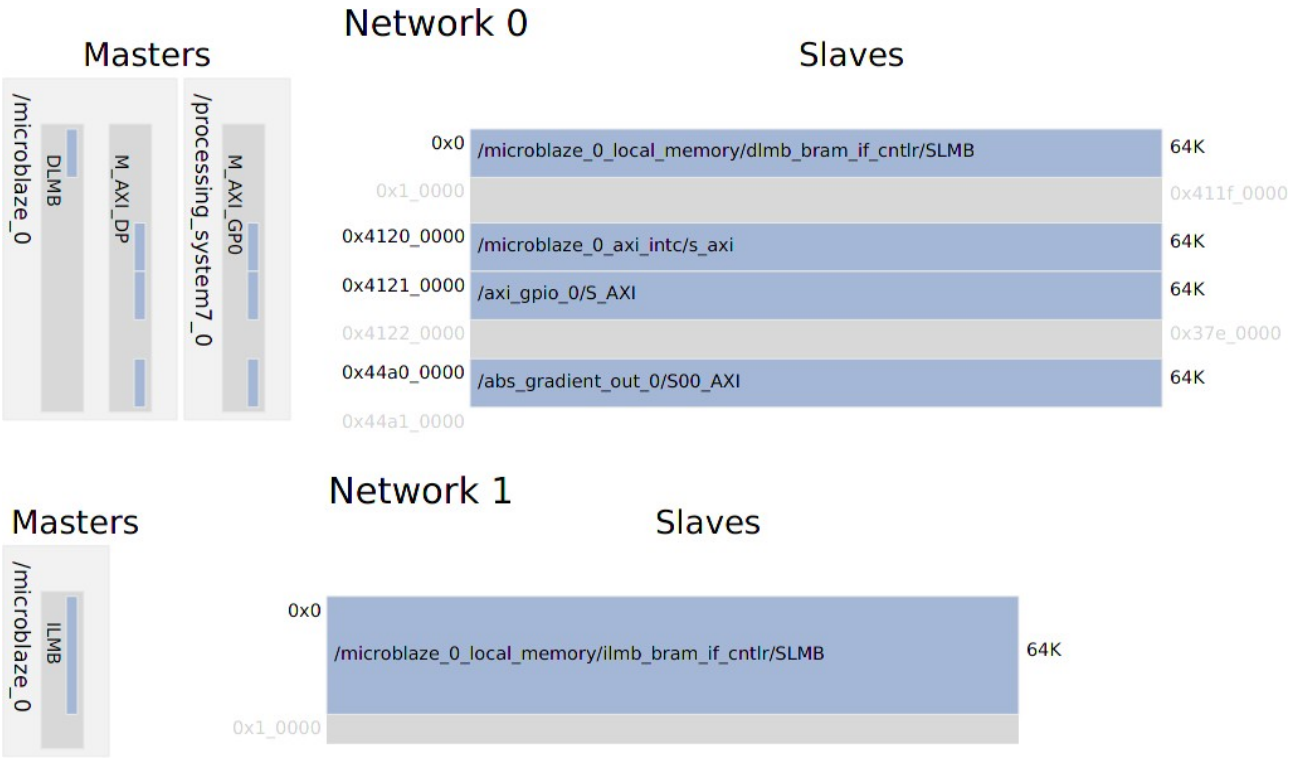
Trigger → OK.

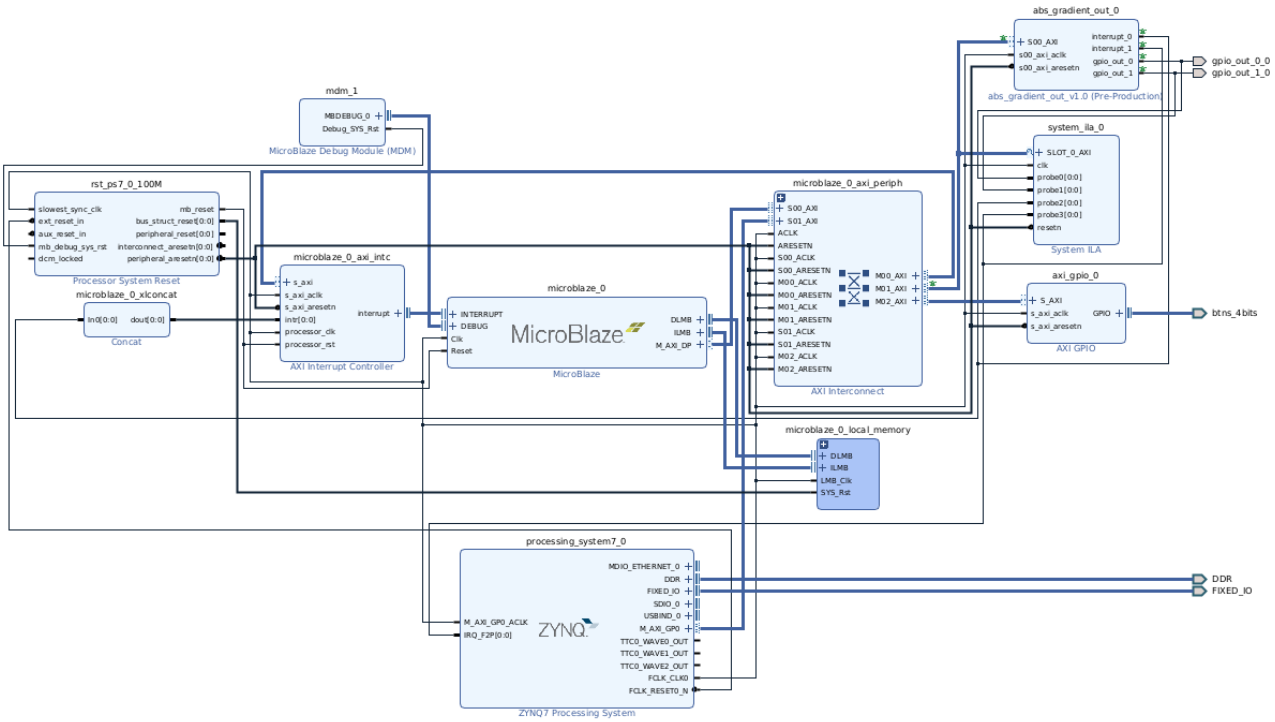
ВНИМАНИЕ: втори System ILA ще бъде добавен в дизайна, който обаче няма да се вижда на блоковата схема. Той ще бъде имплементиран в IP блока abs_gradient_out.

11. От прозорец Diagram → Regenerate Layout.

12. От прозорец Diagram → Validate Design (F6).

Картата на паметта и блоковата схема на системата е показана по-долу.





13. Таб Sources → Design Sources → design_1_wrapper → десен бутон → Refresh Hierarchy.

14. Sources → Constraints → проверете файлът 23_sila_constraints.xdc дали съдържа следните сигнали:

```
#GPIO 0
set_property PACKAGE_PIN V20 [get_ports gpio_out_0_0]
set_property IOSTANDARD LVCMOS33 [get_ports gpio_out_0_0]

#GPIO 1
set_property PACKAGE_PIN W20 [get_ports gpio_out_1_0]
set_property IOSTANDARD LVCMOS33 [get_ports gpio_out_1_0]
```

15. Вляво → Flow Navigator → Generate Bitstream → Yes → OK. Когато синтезът завърши, изберете View Reports.

16. Вляво → Flow Navigator → Synthesis → Open Synthesized Design → Set Up Debug → Next → Continue debugging 52 nets connected to existing debug core → Next → Next → проверете дали са добавени четирите регистъра slv_reg7 ÷ 10 в графата Nets to Debug / Name.

Ако са добавени → задържа се CTRL от клавиатурата и с ляв бутон на мишката се избират и четирите регистъра → Next → отметка на Advanced trigger → Finish.

Ако не са добавени → Find Nets to Add → от прозореца Find nets, в полето Properties → от третото празно поле със звезда “NAME| Constraints| *” изтрийте звездата и напишете “slv_reg7”. В новопоявилият се прозорец Add Nets to Debug → потърсете:

design_1_i/abs_gradient_out_0/abs_gradient_out_v1_0_S00_AXI_inst/slv_reg7

изберете го → OK. Аналогично направете и за другите 3 регистъра. След това задържа се CTRL от клавиатурата и с ляв бутон на мишката се избират и четирите регистъра → Next → отметка на Advanced trigger → Finish.

Бутон Save Constraints → Out of date design: OK → OK.

Вляво → Flow Navigator → **изберете повторно** Generate Bitstream → Yes → OK. Когато синтезът завърши, изберете View Reports.

17. File → Export → Export hardware → Next → Include bitstream → Next → Next → Finish

==========
==========

18. Tools → Launch Vitis IDE

19. Избира се път до workspace за фърмуерния проект→ Launch

ВНИМАНИЕ: възможно е да има останали фърмуерни проекти от минали групи. В таб-а Explorer на средата Vitis със задържане на CTRL от клавиатурата изберете с ляв бутон на мишката всички проекти, след което натиснете десен бутон на мишката и Delete. Ако проектите ще се използват, махнете отметката от “Delete project contents on disk (cannot be undone)” и натиснете OK.

20. File → New → Platform project → Platform project name: 23_sila_cortex_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 23_sila, създаден от Vivado → design_1_wrapper.xsa → Open → избира се Processor: ps7_cortex_a9_0→ Finish.
21. Вляво→Project explorer → избира се 23_sila_pla → right-click → Build Project.
22. File → New → Application project → Next → "Select a platform from repository" → Избира се 23_sila_cortex_pla → Next → Application project name: 23_sila_cortex_app → Next → Next → “Hello World” → Finish.

23. Отворете helloworld.c и въведете следната програма:

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "sleep.h"
#include "xparameters.h"

#define ABS_GRADIENT_OUT_BASE XPAR_ABS_GRADIENT_OUT_0_S00_AXI_BASEADDR

typedef struct {
	int32_t grad_0; //0x00
	int32_t grad_1; //0x04
	int32_t grad_2; //0x08
	int32_t grad_3; //0x0c
	int32_t grad_4; //0x10
	int32_t grad_5; //0x14
	int32_t grad_out; //0x18

	int32_t abs_in0; //0x1c
	int32_t abs_in1; //0x20
	int32_t abs_div; //0x24
	uint32_t abs_out; //0x28

	uint32_t abs_interrupt; //0x2c
	uint32_t gpio_out; //0x30
}abs_gradient_out_t;

#define ABSGRADOUT0	((volatile abs_gradient_out_t *) ABS_GRADIENT_OUT_BASE)

int main(){
	int i = 0;
	init_platform();
	print("Starting ...\n\r");

	while(1){
		ABSGRADOUT0->abs_interrupt = 0x01;

		ABSGRADOUT0->gpio_out = 0x1;

		ABSGRADOUT0->grad_0 = i + 2;
		ABSGRADOUT0->grad_1 = i + 5;
		ABSGRADOUT0->grad_2 = i + 2;
		ABSGRADOUT0->grad_3 = i - 5;
		ABSGRADOUT0->grad_4 = i - 5;
		ABSGRADOUT0->grad_5 = i - 5;

		ABSGRADOUT0->abs_div = 4;
		ABSGRADOUT0->abs_in0 = i - 50;
		ABSGRADOUT0->abs_in1 = i – 50;

		ABSGRADOUT0->abs_out;

		i++;

		ABSGRADOUT0->abs_interrupt = 0x00;

		ABSGRADOUT0->gpio_out = 0x2;
	}
	cleanup_platform();
	return 0;
}
```

24. Вляво → Project explorer → избира се 23_sila_cortex_app → right-click → Build Project.

25. File → New → Platform project → Platform project name: 23_sila_mb_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 23_sila, създаден от Vivado → design_1_wrapper.xsa → Open → избира се Processor: microblaze_0 → Finish.

26. Вляво → Project explorer → избира се 23_sila_mb_pla → right-click → Build Project.

27. File → New → Application project → Next → "Select a platform from repository" → Избира се 23_sila_mb_pla → Next → Application project name: 23_sila_mb_app → Next → Next → Empty Application (C) → Finish.

28. Двукратно щракване върху 23_sila_mb_app_system → 23_sila_mb_app → десен бутон върху src → New → Other → C/C++ → Source File → Next → main_mb.c → Finish. Отворете main_mb.c и въведете следната програма:

```c
#include <stdio.h>
#include <xgpio.h>
#include "xparameters.h"
#include "sleep.h"
#include "xil_exception.h"
#include "xintc.h"

typedef struct {
	UINTPTR BaseAddress;
	u32 IsReady;
	int InterruptPresent;
	int IsDual;
} abs_gradient_out_t;

XGpio output;
abs_gradient_out_t abs_module_0;
XIntc intc_0;

void abs_gradient_interrupt_0(void){
	static int flag = 1;
	if(flag){
		flag = 0;
		XGpio_DiscreteWrite(&output, 1, 0x01);
	}
	else{
		flag = 1;
		XGpio_DiscreteWrite(&output, 1, 0x00);
	}
}

int main(void){
	abs_module_0.BaseAddress = (volatile
uint32_t)XPAR_ABS_GRADIENT_OUT_0_S00_AXI_BASEADDR;
	abs_module_0.IsReady = 1;
	abs_module_0.IsDual = 0;
	abs_module_0.InterruptPresent = 1;

	XGpio_Initialize(&output, XPAR_AXI_GPIO_0_DEVICE_ID);
	XGpio_SetDataDirection(&output, 1, 0x0);

	XIntc_Initialize(&intc_0, XPAR_INTC_0_DEVICE_ID);
	XIntc_SelfTest(&intc_0);
	XIntc_Connect(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_0_INTR,
(XInterruptHandler)abs_gradient_interrupt_0, &abs_module_0);
	XIntc_Start(&intc_0, XIN_REAL_MODE);
	XIntc_Enable(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_0_INTR);
	Xil_ExceptionInit();
	Xil_ExceptionEnable();
	Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,
(Xil_ExceptionHandler)XIntc_InterruptHandler, &intc_0);
	XIntc_Acknowledge(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_0_INTR);

	while(1){ }

	return 0;
}
```

29. Вляво → Project explorer → избира се 23_sila_mb_app → right-click → Build Project.

30. В основния прозорец на Vitis до бутонът Debug има стрелка надолу → натиска се → Debug configurations … → щраква се двукратно върху Single Application Debug → вдясно ще се появи нова конфигурация на дебъг сесия. Избира се таб Application и се слагат отметки на двата процесора: microblaze_0 и ps7_cortexa9_0. Проверяват се полетата Application, указващи фърмуерния .elf файл за всеки процесор. Полето на MicroBlaze може да бъде празно. Затова с ляв бутон в полето Summary се избира целият ред на microblaze_0 и се натиска бутон Search срещу полето Application. Средата Vitis ще предложи всички .elf файлове, които са достъпни. С ляв бутон се натиска двукратно върху съответния файл на MicroBlaze (23_sila_mb_app.elf). Сега в полетата Application на Summary трябва да се вижда:

microblaze_0 Debug/23_sila_mb_app .elf
ps7_cortexa9_0 Debug/23_sila_cortex_app .elf

Натиска се Apply → Debug

ВНИМАНИЕ: Всяко следващо стартиране на Debug сесия може да стане с бутон надолу до Debug бутона от основния прозорец на Vitis, при условие, че поне веднъж е била стартирана дебъг сесия от Debug configurations… прозореца (в конкретния случай това е станало, когато сме натиснали Apply → Debug).

ВНИМАНИЕ: не трябва да се натиска самият бутон Debug понеже това създава нова дебъг сесия, която по подразбиране зарежда фърмуер само на едно Cortex A9 ядро.

31. Дебъгването на отделните микропроцесори става като се избере с ляв бутон съответния процесор от таб Debug. Дебъг бутоните и всички дебъг табове се присвояват автоматично на избрания процесор, т.е. въпреки че процесорите са два, наборът от дебъг инструменти е един.

32. В изглед Debug на Vitis → таб Debug → избира се MicroBlaze #0 → натиска се бутон Resume (F8) → избира се ARM Cortex-A9 MPCore #0 → натиска се бутон Resume (F8).

Светодиод LD1 трябва да започне да свети постоянно.

33. Върнете се обратно във Vivado → Flow Navigator → щракнете върху текста Open Hardware Manager → зелена лента → Open target → Auto connect → в таб Hardware щракнете върху hw_ila_1 от категорията xc7z010 → в прозореца New

Dashboard → избират се прозорците на логическия анализатор, т.е. слагат се отметки на всички категории → OK.

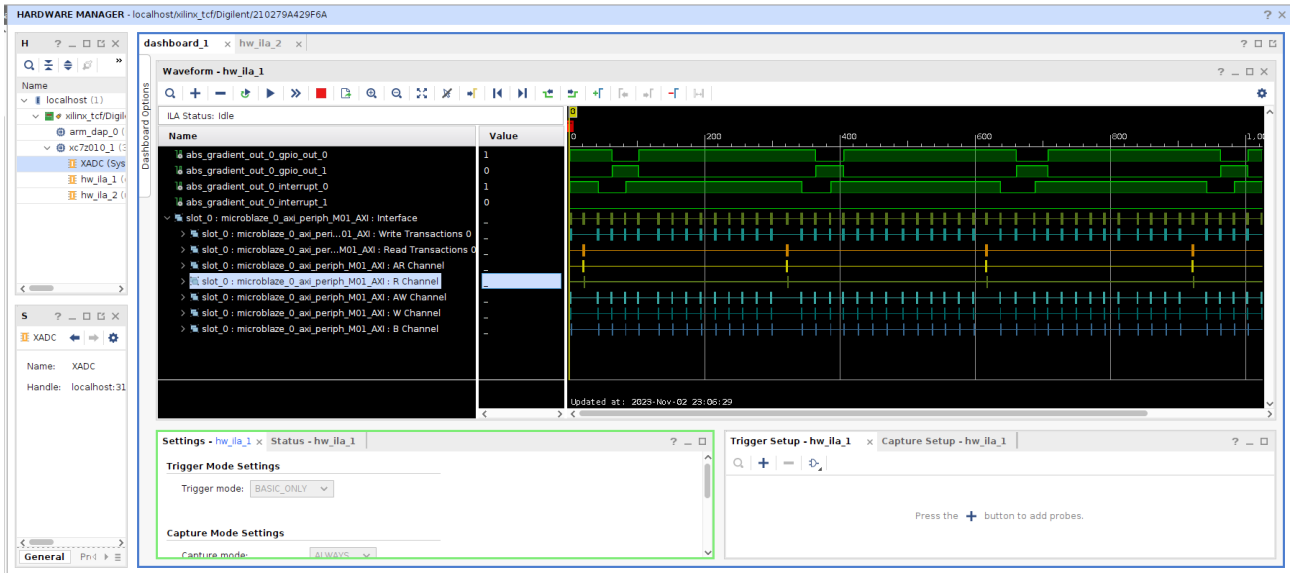
34. Би трябвало да се отвори Waveform прозорец. Под него → таб Settings → на Window data depth → избира се 1024 → бутон Run trigger immediate for this ILA core → Zoom Fit → трябва да се видят сигналите за прекъсване и сигналите на GPIO модула, заедно с данните по AXI магистралата.

Забележка: адресите по AXI магистралата се показват с отместванията от базовия адрес на IP модула, т.е. вместо:

```
0x44a0002c
```

се показва:

```
0x2c
```



slot_0 : microblaze_0_axi_periph_M01_AXI : W Channel
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: WVALID
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: WREADY
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: WDATA ← тук се виждат записите по AXI магистралата в посока към IP модула
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: WSTRB

slot_0 : microblaze_0_axi_periph_M01_AXI : R Channel
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: RVALID
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: RREADY
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: RDATA ← тук се вижда четенето по AXI магистралата в посока от IP модула към главното устройство
|-----+ slot_0 : microblaze_0_axi_periph_M01_AXI: RSTRB

35. Свържете осцилоскоп към изводите V20 (gpio_out_0) и W20 (gpio_out_1), които излизат на JB конектора под имената JB2_P и JB2_N. Проверете дали получената форма на сигналите е аналогична с тази от логическия анализатор.

36. По-долу са дадени отместванията спрямо базовия адрес на персонализираният IP модул:

```
typedef struct {
	int32_t grad_0; //0x00
	int32_t grad_1; //0x04
	int32_t grad_2; //0x08
	int32_t grad_3; //0x0c
	int32_t grad_4; //0x10
	int32_t grad_5; //0x14
	int32_t grad_out; //0x18

	int32_t abs_in0; //0x1c
	int32_t abs_in1; //0x20
	int32_t abs_div; //0x24
	uint32_t abs_out; //0x28

	uint32_t abs_interrupt; //0x2c
	uint32_t gpio_out; //0x30
}abs_gradient_out_t;
```

Спрете изпълнението на програмата във Vitis. Заредете я наново, пуснете (с Resume) само фърмуера на MicroBlaze.
Във Vivado → таб hw_ila_2 (логическият анализатор на abs_gradient) → Trigger Setup → Add probe(s) → избира се slv_reg7.

Чрез останалите полета от таб Trigger Setup поставете тригер с условие, така че логическият анализатор да почне да записва, когато **променливата i стане равна на 100**. Разгледайте графиките и се опитайте да разберете каква стойност е върнал abs_out модула като резултат. Проверете дали този резултат е еднакъв с теоретичния. Натиснете бутонът за еднократен тригер Run trigger for this ILA core и във Vitis натиснете Resume (F8) на ARM Cortex-A9 MPCore #0.

Забележка: с CTRL + scroll на мишката можете да разтеглите графиките по време.

Забележка: формулата, с която работи модула е:

abs_out = (abs_in0 + abs_in1) / abs_div

ВНИМАНИЕ: ако изтриете сигнал от анализатора, за да го възстановите трябва да изберете от Waveform → бутон Add (син знак плюс) → избирате името на сигнала от списъка.

* * *

[1] https://docs.xilinx.com/r/en-US/ug994-vivado-ip-subsystems/Using-the-System-ILA-IP-to-Debug-a-Block-Design

[2] http://www.ue.eti.pg.gda.pl/fpgalab_new/index.php?id=lab-5-hwsw-system-debug&lang=pl

[3] https://xilinx.github.io/xup_embedded_system_design_flow/lab6.html

[4] https://ez.analog.com/cfs-file/__key/telligent-evolution-components-attachments/00-323-00-00-00-03-21-07/DebuggingFPGAimages.pdf

доц. д-р инж. Любомир Богданов, 2023 г.