Общество с ограниченной ответственностью «ИМТ»

ТСРП.468157.014

Плата ЦОС-310-4К-АЦП
Руководство пользователя

«Утверждаю»
Генеральный директор
Погребной Д. С.

« » _____ 2018 г.

Ярославль, 2018

Введение

Плата ЦОС-310-4К-АЦП (далее — «плата ЦОС») представляет собой 4-х канальное устройство цифровой обработки на основе ПЛИС. Плата полностью совместима по посадочному месту и габаритам с платами ЦОС-140/25-2К и ЦОС-310-2К. Плата имеет четыре аналоговых входа, и может подключаться к ПК при помощи интерфейса USB 3.0. Внешний вид платы приведён на рис. 1.

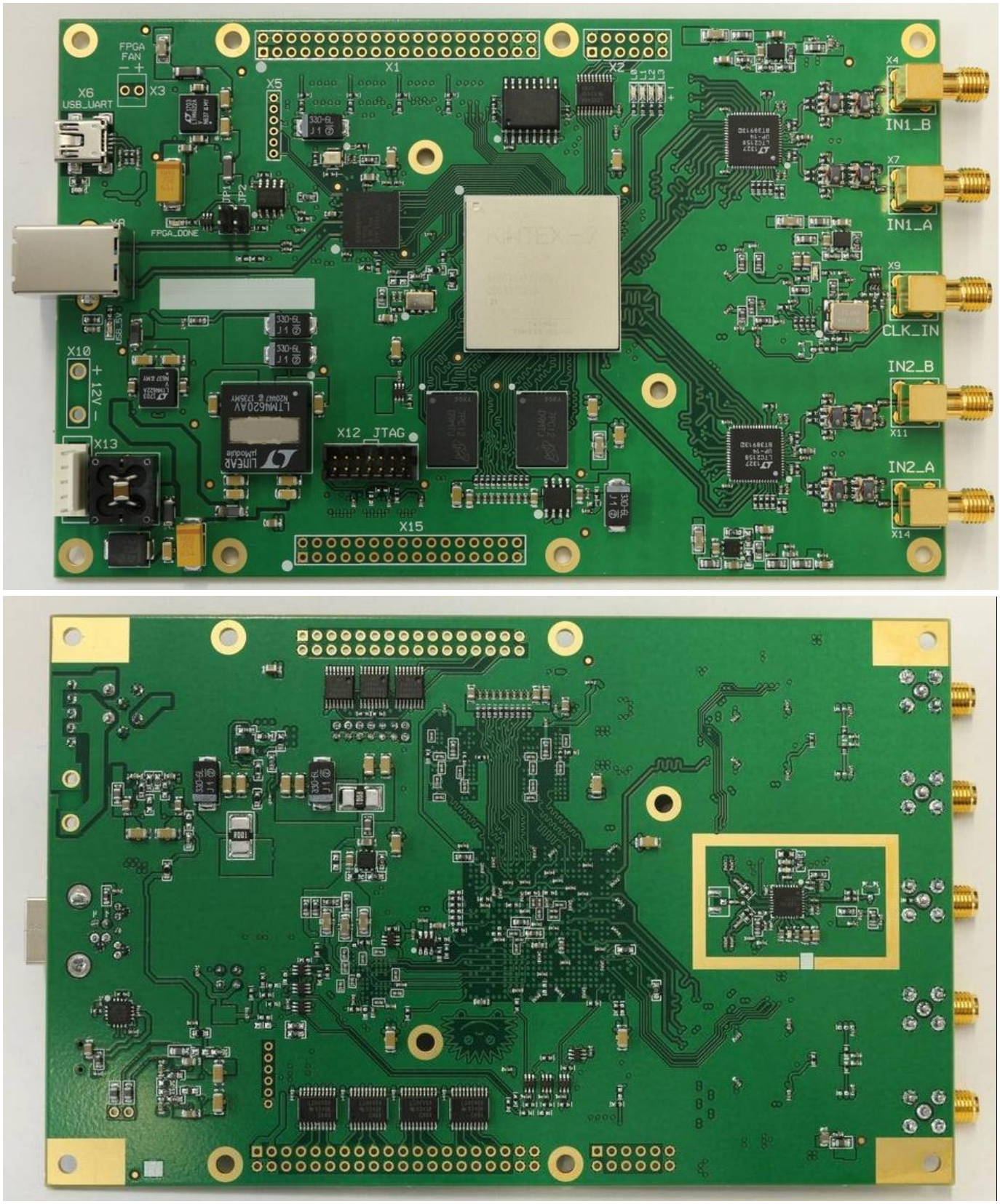


Рис. 1 Внешний вид платы ЦОС-310-4К-АЦП

Технические характеристики

1) **Аналоговые входы**

– Сопротивление – 50 Ω;
– Развязка при помощи широкополосных трансформаторов TC1-1-13M.

2) **АЦП (Linear Technologies, LTC2158-14, www.linear.com)**

– Разрядность: 14 бит;
– Частота дискретизации: до 310 МГц ;
– SFDR: ~88 dB;
– Входной диапазон частот 0…1,25 ГГц.

3) **Синтезатор тактовой частоты (Texas Instruments, LMX2581, www.ti.com)**

– Интегральный джиттер ~300 фс@310МГц;
– Фазовый шум -123 дБн/Гц при отстройке 10 кГц на частоте 310 МГц (при использовании установленного на плату опорного генератора 10 МГц Geyer KXO-V97T);
– Возможность подачи опорной частоты или тактовой частоты с внешнего разъема;

4) **ПЛИС (Xilinx Kintex-7, Part Number: XC7K160T-2FFG676I, www.xilinx.com)**

– ~ 160000 LE;
– 600 аппаратных умножителей DSP48E1 25*18;
– 325 блоков двухпортовой памяти по 36 Кбит;

5) **Память DDR2-800**

– Объем 512 Мбайт, организация 128M x 32 (две микросхемы MT47H128M16RT-25E-IT)

6) **Интерфейсы**

– USB 3.0 (CYUSB3014**),** www.cypress.com);
– Логические входы-выходы TTL 3.3 В, всего 68 линий;

7) **Питание**

– Напряжение DC: 9...14 Вольт;
– Макс. потребляемая мощность – 15 Ватт.

Устройство и основные компоненты

Структура платы ЦОС приведена на рисунке 2.

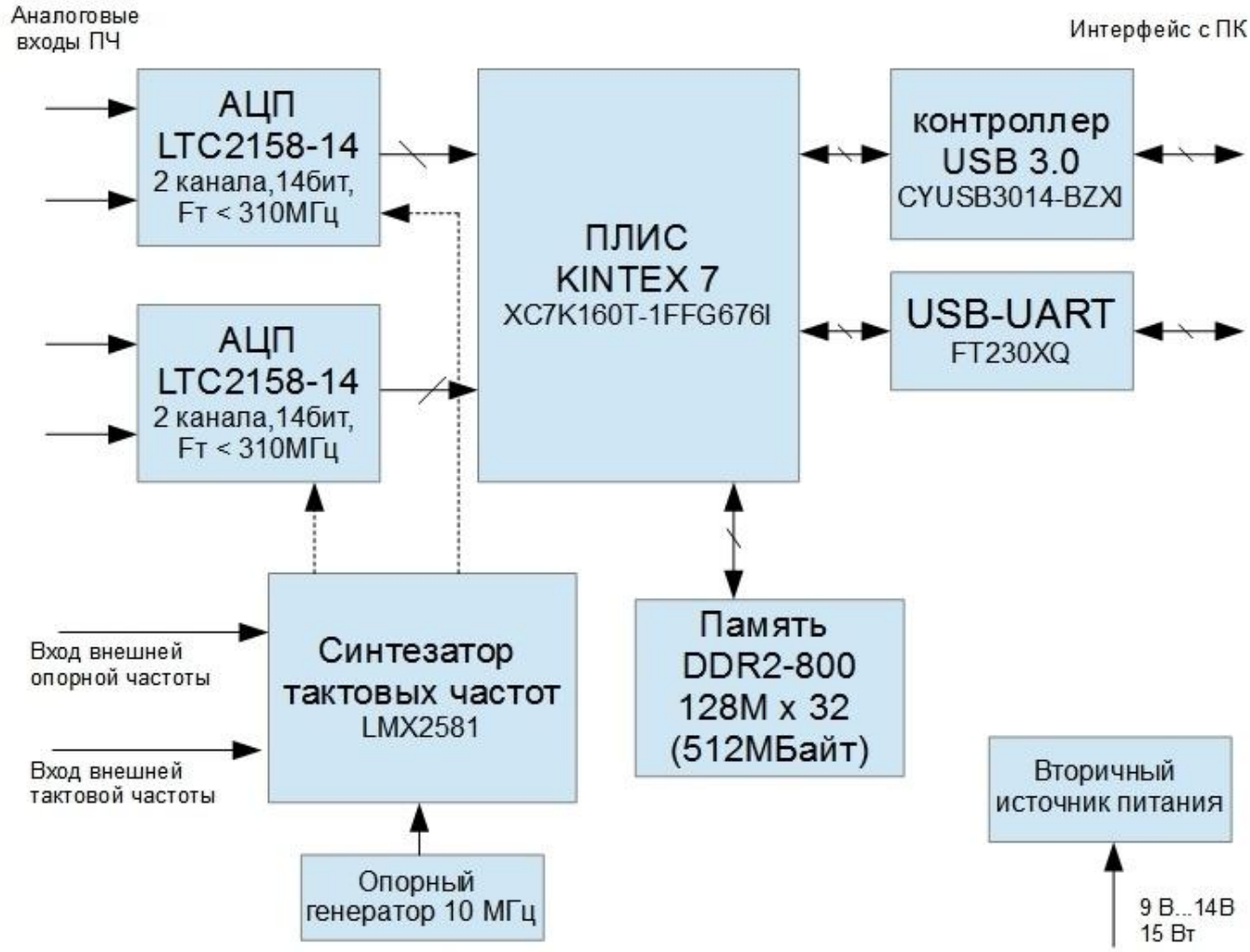


Рис.2. Структурная схема платы ЦОС.

Внешние интерфейсы

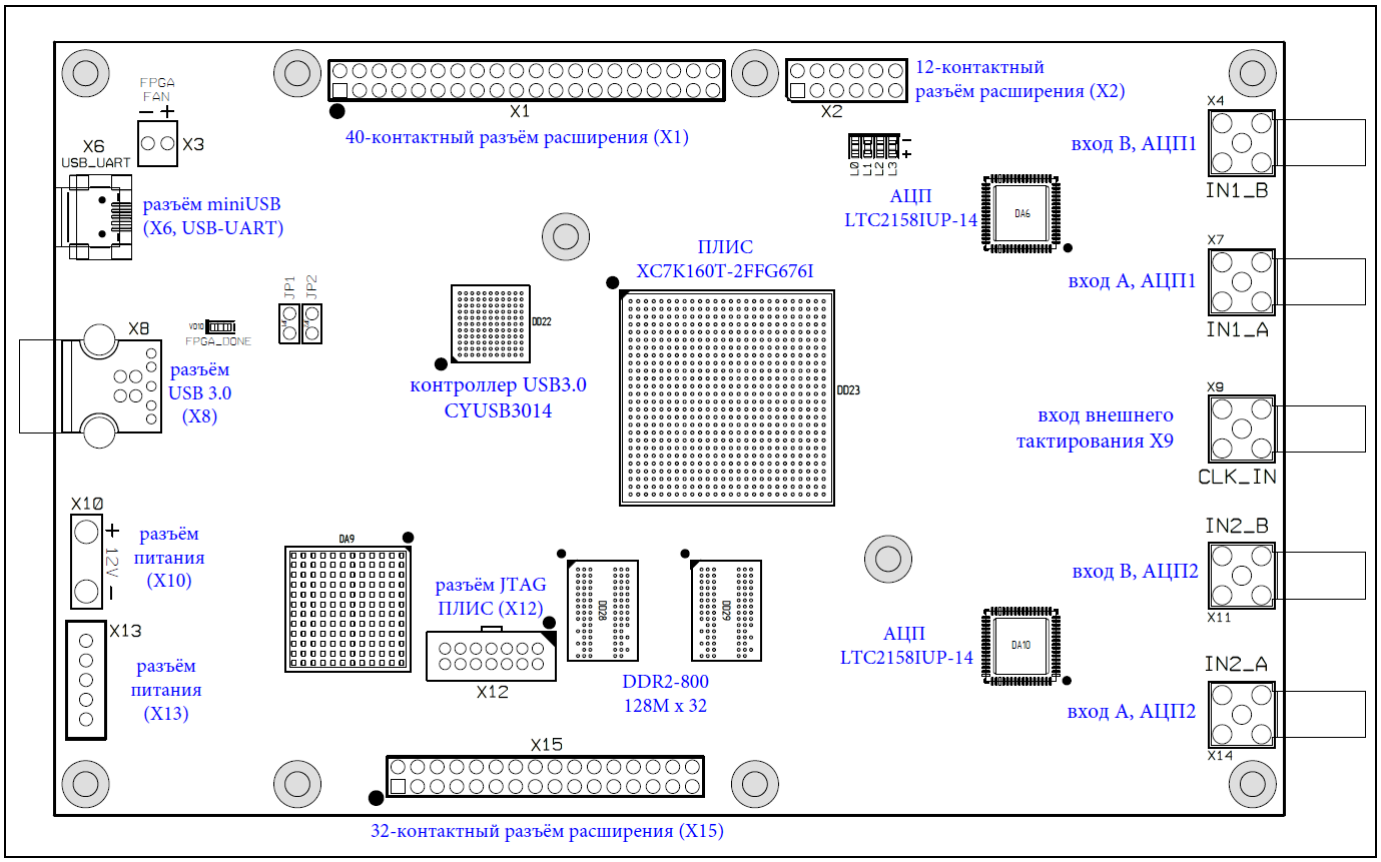


Рис. 3. Внешние интерфейсы платы ЦОС.

Для подключения к другим устройствам и взаимодействия с пользователем на плате предусмотрены следующие внешние интерфейсы:

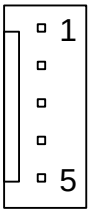
- Разъём питания,
- Разъём внутрисхемного программирования и отладки ПЛИС (JTAG),
- 4 разъёма аналоговых входов АЦП,
- 1 разъём внешнего тактирования,
- Порт USB 3.0
- Порт USB 2.0 (USB-UART)
- 4 индикаторных светодиода,
- 40 внешних буферизированных логических выводов общего назначения, подключённых к ПЛИС через двунаправленные буферы 74LCX245MTC,
- 24 внешних буферизированных логических вывода общего назначения, подключённых к ПЛИС через сдвиговые регистры 74LVC595APW.
- 3 внешних небуферизированных логических вывода общего назначения, подключённых напрямую к ПЛИС,
- 1 внешний буферизированный логический вход, подключённый к ПЛИС через микросхему NC7SZ18P6X.

Расположение внешних интерфейсов на плате, с указанием ориентации разъёмов, показано на рисунке 3. Подробное описание интерфейсов приведено ниже.

Разъём питания (X13)

Разъём питания X13 представляет собой вилку типа WF-5 (ответная часть — розетка HU-5, согласно каталогу www.chipdip.ru), и служит для подачи электропитания. Для работы устройства необходимо однополярное питание напряжением от +9 до +14 Вольт. Контакты разъёма соединены параллельно, для повышения надёжности. Расположение и назначение контактов

приведено ниже.



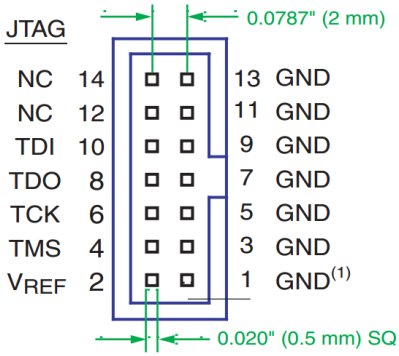
1	Общий (GND)
2	Питание (+9...+14 В)
3	Общий (GND)
4	Питание (+9...+14 В)
5	Общий (GND)

Разъём питания (X10)

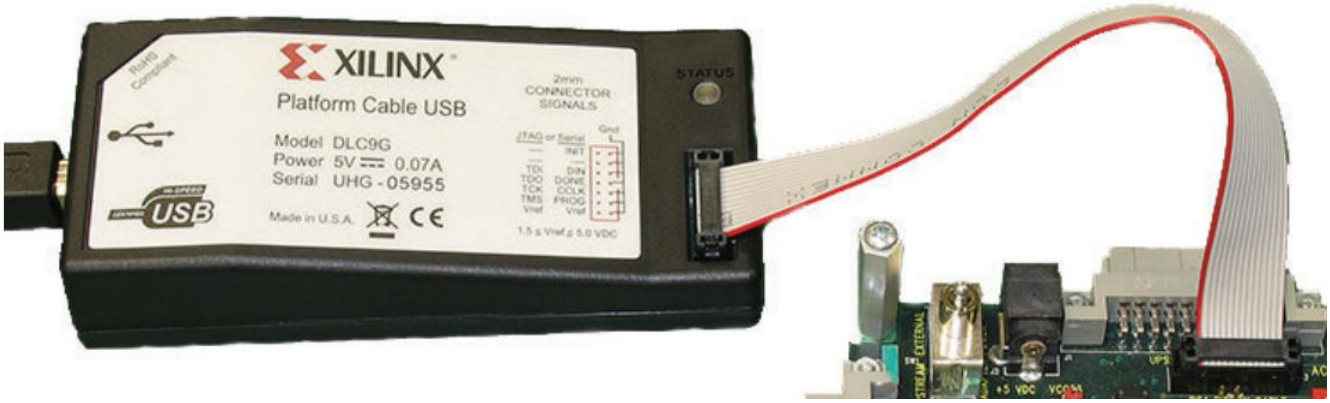
Разъём питания X10 предназначен под запайку проводов и соединён параллельно с разъёмом питания X13.

Разъём JTAG ПЛИС (X12)

Разъём JTAG используется для внутрисхемного программирования ПЛИС и внешней SPI Flash-памяти ПЛИС (M25P128-VMF6TPB), а также для отладки загруженного в ПЛИС ПО. Разъём представляет собой двухрядную вилку, с шагом контактов 2 мм типа Molex 87831-1420. Расположение и назначение контактов приведено ниже.



Разъём JTAG аналогичен по цоколёвке разъёму на программаторах Xilinx Platform Cable USB и Xilinx Platform Cable USB II и предназначен для подключения к ним с помощью кабеля входящего в комплект поставки программаторов (см. рисунок ниже):



Разъёмы аналоговых входов АЦП (X4,X7,X11,X14)

Аналоговые входы АЦП выведены на 4 коаксиальных ВЧ-разъёма типа SMA. Все разъёмы имеют согласованное сопротивление 50 Ом. Для согласования коаксиального подключения с дифференциальными входами АЦП применена трансформаторная развязка, которая ограничивает диапазон частот сигналов снизу.

Разъём внешнего тактирования (X9)

Для тактирования АЦП и ПЛИС на плате имеется программируемый синтезатор частот

LMX2581, и кварцевый генератор опорной частоты 10МГц Geyer KXO-V97T для него.

Для реализации возможности внешнего тактирования на плате предусмотрен разъём типа SMA (**X9**). Входное сопротивление внешнего тактового входа составляет 1 кОм, и определяется резистором **R63**. При необходимости изменения входного сопротивления необходимо перепаять резистор. Имеется два варианта использования разъёма внешнего тактирования:

- подача внешней опорной частоты
- подача внешней тактовой частоты.

Для режима подачи внешней опорной частоты необходимо отключить встроенный опорный генератор, выпаяв резистор **R181** номиналом 22Ом и запаяв резистор **R189** номиналом 22Ом или менее). После этого опорную частоту можно подавать через внешний разъём.

Значение опорной частоты, совместимое с тестовым ПО ПЛИС, предполагается 10МГц. Для других значений опорной частоты, кратных 10МГц, потребуется изменение значений регистров синтезатора LMX2581, которые загружаются в него при старте тестового ПО ПЛИС по SPI интерфейсу. При некратных 10МГц опорных частотах возможно потребуется перерасчёт и перепайка элементов петлевого фильтра синтезатора LMX2581, рассчитанного на частоту сравнения 10МГц. Параметры петлевого фильтра синтезатора, для которых выполнялся его расчёт, указаны на принципиальной схеме.

Для режима подачи внешней тактовой частоты необходимо запаять компоненты **C266**, (например, номиналом 0.1мкФ) и **R192** (например, номиналом 0Ом) и изменить значения регистров синтезатора LMX2581, чтобы использовать его для разветвления внешней тактовой частоты на АЦП и ЦАП.

USB-порты и светодиоды

Установленный на плате разъём USB 3.0 типа B (**X8**) подключен непосредственно к универсальному контроллеру интерфейса USB (Cypress CYUSB3014-BZXI), и предназначен для подключения платы к управляющему компьютеру. Функционирование интерфейса USB 3.0 полностью определяется программным обеспечением, загруженным в ПЛИС и контроллер USB 3.0.

Также на плате установлен разъём mini-USB (**X6**), подключенный к мосту USB-UART FTDI FT230XQ, позволяющий обмениваться данными между ПЛИС и ПК через виртуальный COM-порт.

Светодиоды **VD6, VD7, VD8, VD9** подключены через токоограничивающие резисторы к выводам ПЛИС **LED0, LED1, LED2, LED3** соответственно (см. UCF-файл). Установка логической «1» на данных выводах ПЛИС вызывает свечение соответствующего светодиода. Соответственно, светодиоды могут быть использованы для оперативного вывода пользователю информации о текущем состоянии работы ПО ПЛИС.

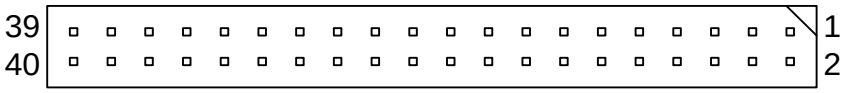
40-контактный разъём расширения (X1)

На плате предусмотрен разъём, на который выведены 32 логических сигнала, а также напряжение питания 3,3 вольта, и «земля» (GND). Все 32 логических сигнала подключены к ПЛИС через буферы 74LCX245 и могут быть как входами так и выходами. При этом направление работы буферов задаётся резисторами на плате **R121, R122, R123, R124, R128, R129, R130, R131**. По умолчанию все буферы и, соответственно, все 32 вывода настроены на выход.

По умолчанию разъём на плату не запаивается. Посадочное место на плате представляет собой 2 ряда по 20 отверстий со стандартным шагом 2,54 мм и позволяет установить любой разъём (штыри, гнёзда, в 1 или 2 ряда, и т.д.) с таким шагом контактов.

Нумерация контактов и их назначение приведены ниже. Названия выводов ПЛИС приведены в соответствии с прилагаемым UCF-файлом. Направление «→» соответствует передаче сигнала из платы ЦОС наружу – “выход”. Направление «←» – “вход”.

Логические уровни сигналов соответствуют стандарту КМОП 3.3В.



№	Направление	Цепь/вывод ПЛИС		№	Направление	Цепь/вывод ПЛИС
1	→	+3,3 В		2		GND (общий)
3	→	+3,3 В		4		GND (общий)
5	↔ буф.	CONN<31>		6	↔ буф.	CONN<0>
7	↔ буф.	CONN<30>		8	↔ буф.	CONN<1>
9	↔ буф.	CONN<29>		10	↔ буф.	CONN<2>
11	↔ буф.	CONN<28>		12	↔ буф.	CONN<3>
13	↔ буф.	CONN<27>		14	↔ буф.	CONN<4>
15	↔ буф.	CONN<26>		16	↔ буф.	CONN<5>
17	↔ буф.	CONN<25>		18	↔ буф.	CONN<6>
19	↔ буф.	CONN<24>		20	↔ буф.	CONN<7>
21	↔ буф.	CONN<23>		22	↔ буф.	CONN<8>
23	↔ буф.	CONN<22>		24	↔ буф.	CONN<9>
25	↔ буф.	CONN<21>		26	↔ буф.	CONN<10>
27	↔ буф.	CONN<20>		28	↔ буф.	CONN<11>
29	↔ буф.	CONN<19>		30	↔ буф.	CONN<12>
31	↔ буф.	CONN<18>		32	↔ буф.	CONN<13>
33	↔ буф.	CONN<17>		34	↔ буф.	CONN<14>
35	↔ буф.	CONN<16>		36	↔ буф.	CONN<15>
37		GND (общий)		38		GND (общий)
39		GND (общий)		40		GND (общий)

Нумерация контактов для 40-контактного разъёма расширения (X1)

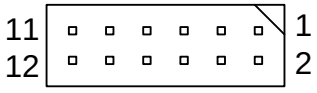
12-контактный разъём расширения (X2)

На плате предусмотрен 12-контактный разъём расширения, на который выведены 8 логических сигналов, подключённых к ПЛИС через двунаправленный логический буфер 74LCX245, а также напряжение питания 3,3 Вольта, и общая «земля» (GND). Все 8 сигналов одновременно могут быть либо входами, либо выходами и направление работы буфера можно переключать из ПЛИС, с помощью вывода ПЛИС “**sys_rst**” (см. UCF файл).

По-умолчанию разъём на плату не запаивается. Посадочное место на плате представляет собой 2 ряда по 6 отверстий со стандартным шагом 2,54 мм и позволяет установить любой разъём с таким шагом контактов.

Нумерация контактов и их назначение приведены ниже. В графе «Цепь/вывод ПЛИС» приведены названия выводов ПЛИС, соответствующих контактам разъёма (до буфера), в соответствии с прилагаемым UCF-файлом.

Логические уровни сигналов соответствуют стандарту КМОП 3.3В.



№	Направление	Цепь/вывод ПЛИС		№	Направление	Цепь/вывод ПЛИС
1	→	+3,3 В		2		GND (общий)
3	↔ буф.	CONN<32>		4	↔ буф.	CONN<33>
5	↔ буф.	CONN<34>		6	↔ буф.	CONN<35>
7	↔ буф.	CONN<36>		8	↔ буф.	CONN<37>
9	↔ буф.	CONN<38>		10	↔ буф.	CONN<39>
11		GND (общий)		12		GND (общий)

Нумерация контактов для 12-контактного разъёма расширения (X2)

32-контактный разъём расширения (X15)

На плате предусмотрен разъём, на который выведены 28 логических сигналов, а также напряжение питания 3,3 вольта, и «земля» (GND).

24 сигнала подключены к ПЛИС через сдвиговые регистры (74LVC595APW), объединённые в цепочку (daisy chain) и могут быть только выходами (см. принципиальную схему). Запись в регистры осуществляется с помощью SPI интерфейса младшим значащим битом вперед.

Сигнал **CONN_AUX_SYNC_IN** подключен к ПЛИС через буфер NC7SZ18P6X и может быть только входом.

3 сигнала **CONN_AUX_FR_STROBE**, **CONN_AUX_TX_PAUSE** и **CONN_AUX_START** подключены напрямую к ПЛИС без буферов и могут быть как входами так и выходами.

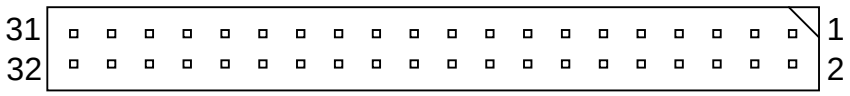
Сигналы SPI интерфейса цепочки сдвиговых регистров:

- **CONN_AUX_SPI_CLK;**
- **CONN_AUX_SPI_DATA;**
- **CONN_AUX_SPI_LE.**

По умолчанию разъём на плату не запаивается. Посадочное место на плате представляет собой 2 ряда по 16 отверстий со стандартным шагом 2,54 мм и позволяет установить любой разъём (штыри, гнёзда, в 1 или 2 ряда, и т.д.) с таким шагом контактов.

Нумерация контактов разъёма и их назначение приведены ниже. Названия выводов ПЛИС приведены в соответствии с прилагаемым UCF-файлом. Направление «→» соответствует передаче сигнала из платы ЦОС наружу – “выход”. Направление «←» – “вход”.

Логические уровни сигналов соответствуют стандарту КМОП 3.3В.



№	Направление	Цепь/вывод ПЛИС (разъёма)	№	Направление	Цепь/вывод ПЛИС (разъёма)
1		GND (общий)	2		GND (общий)
3	→ сдв.рег.	CONN_AUX_SHREG_Q7	4	← буф.	CONN_AUX_SYNC_IN
5	→ сдв.рег.	CONN_AUX_SHREG_Q6	6	→ сдв.рег.	CONN_AUX_SHREG_Q15
7	→ сдв.рег.	CONN_AUX_SHREG_Q5	8	→ сдв.рег.	CONN_AUX_SHREG_Q14
9	→ сдв.рег.	CONN_AUX_SHREG_Q4	10	→ сдв.рег.	CONN_AUX_SHREG_Q13
11	→ сдв.рег.	CONN_AUX_SHREG_Q3	12	→ сдв.рег.	CONN_AUX_SHREG_Q12
13	→ сдв.рег.	CONN_AUX_SHREG_Q2	14	→ сдв.рег.	CONN_AUX_SHREG_Q11
15	→ сдв.рег.	CONN_AUX_SHREG_Q1	16	→ сдв.рег.	CONN_AUX_SHREG_Q10
17	→ сдв.рег.	CONN_AUX_SHREG_Q0	18	→ сдв.рег.	CONN_AUX_SHREG_Q9
19	→ сдв.рег.	CONN_AUX_SHREG_Q22	20	→ сдв.рег.	CONN_AUX_SHREG_Q8
21	→ сдв.рег.	CONN_AUX_SHREG_Q21	22	→ сдв.рег.	CONN_AUX_SHREG_Q20
23	↔	CONN_AUX_FR_STROBE	24	→ сдв.рег.	CONN_AUX_SHREG_Q19
25	→ сдв.рег.	CONN_AUX_SHREG_Q23	26	→ сдв.рег.	CONN_AUX_SHREG_Q18
27	↔	CONN_AUX_TX_PAUSE	28	→ сдв.рег.	CONN_AUX_SHREG_Q17
29	↔	CONN_AUX_START	30	→ сдв.рег.	CONN_AUX_SHREG_Q16
31		GND (общий)	32	→	+3,3 В

Нумерация контактов для 32-контактного разъёма расширения (X15)

Прилагаемое тестовое программное обеспечение

Прилагаемое тестовое программное обеспечение состоит из:

1) Программного обеспечения с исходными кодами на языке “C” для контроллера USB 3.0 CYUSB3014, в виде проекта для EZ-USB FX3 SDK 1.2.2 (установочные файлы прилагаются).
2) Тестовый проект ПЛИС, задействующий все основные компоненты платы ЦОС, разработанный в среде Xilinx ISE 14.7 с исходными кодами на языке VHDL.
3) Тестовая программа для ПК, в виде проекта для Borland C++ Builder 6.0 с исходными кодами на языке “C++”.
4) Драйвера для ОС 32-бит и 64-бит Windows XP, Windows 7 (совместимость с Windows 8 не проверялась).

Плата поставляется с уже “прошитыми” загрузочными микросхемами Flash-памяти для контроллера USB и ПЛИС, поэтому для тестирования достаточно подключить плату к ПК, поставить соответствующий драйвер и запустить тестовую программу на ПК.

ПО контроллера USB 3.0

Программное обеспечение (ПО) для контроллера USB 3.0 CYUSB3014 прилагается в архиве cyusb_*soft.7z*. В архиве содержатся исходные файлы ПО на языке “C” для контроллера USB 3.0 CYUSB3014, в виде проекта для EZ-USB FX3 SDK 1.2.2 (установочные файлы прилагаются).

ПО контроллера USB создаёт два Endpoint (“Endpoint IN” и “Endpoint OUT”) в режиме BULK, с размером пакета 1024 Байта в режиме SuperSpeed (512 Байт в режиме HighSpeed) - см. рис.4. Интерфейс между контроллером USB и ПЛИС работает на частоте 100МГц, которая формируется внешним кварцевым генератором и поступает на ПЛИС.

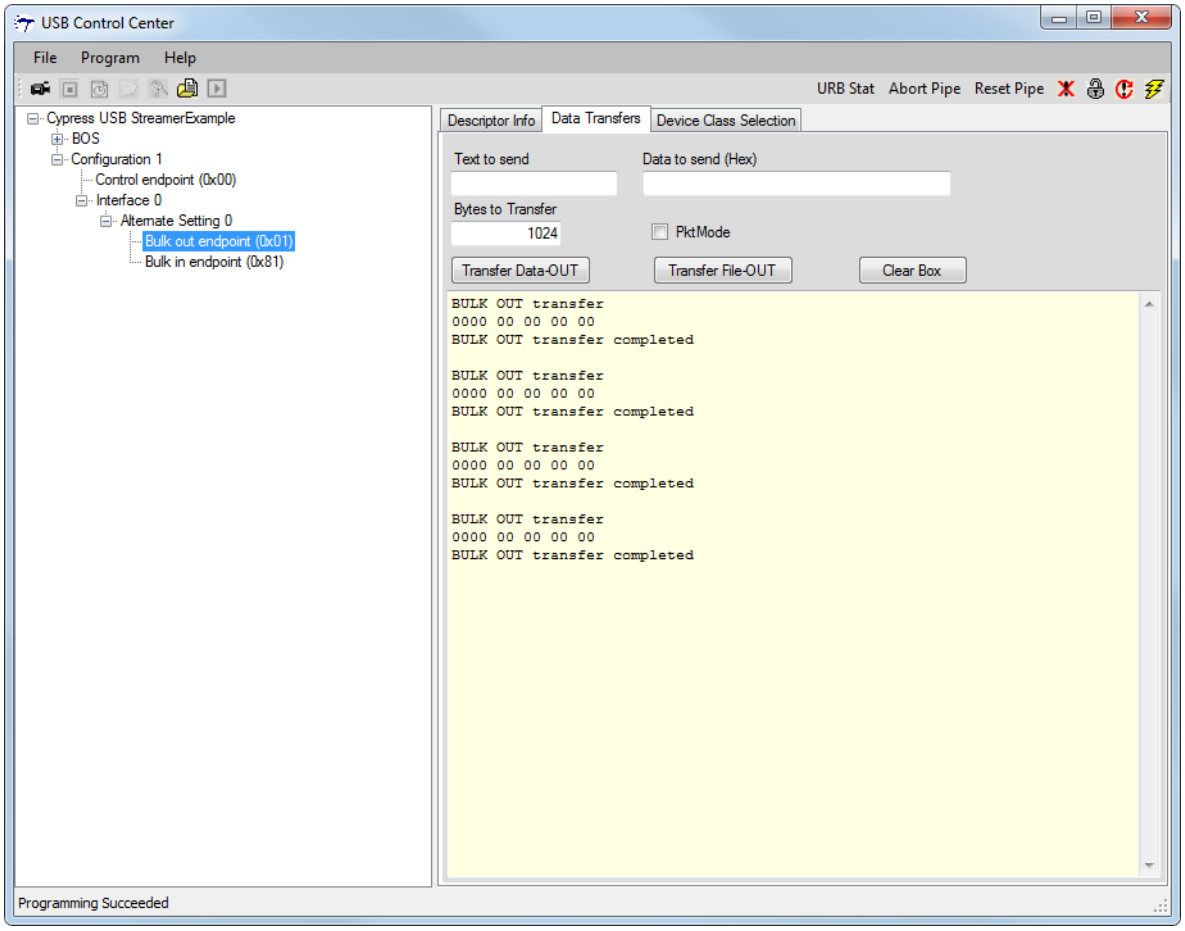


Рис.4. Отображение платы ЦОС-310-4К-АЦП в программе Cypress USB Console с тестовым программным обеспечением.

Плата поставляется с прошитым во внешнюю флеш-память тестовым ПО контроллера CYUSB3014. Для прошивки другого или изменённого ПО контроллера CYUSB3014 необходимо установить пакет программ, который содержится в комплекте поставки (установочный файл “FX3DVKSetup.exe”). Далее замкнуть (установить) джамперы **JP1**, **JP2** (см. рис. 3). Джамперы прилагаются в комплекте поставки. Далее подключить плату к компьютеру по интерфейсу USB 3.0 и включить питание платы. Запустить программу “USB Control Center”, в которой будет отображаться подключенный контроллер в режиме загрузчика (см. рис. 5). Далее выделить отображаемый контроллер левой кнопкой мыши и последовательно выбрать пункты меню “Program -> FX3 -> SPI FLASH ” и выбрать бинарный img-файл, содержащий скомпилированное ПО (см. рис. 6). ПО записывается во внешнюю флеш-память контроллера и будет автоматически загружаться в контроллер при каждом включении платы со снятыми джамперами **JP1, JP2**. Для того чтобы ПО контроллера USB начало работу необходимо выключить и включить питание платы.

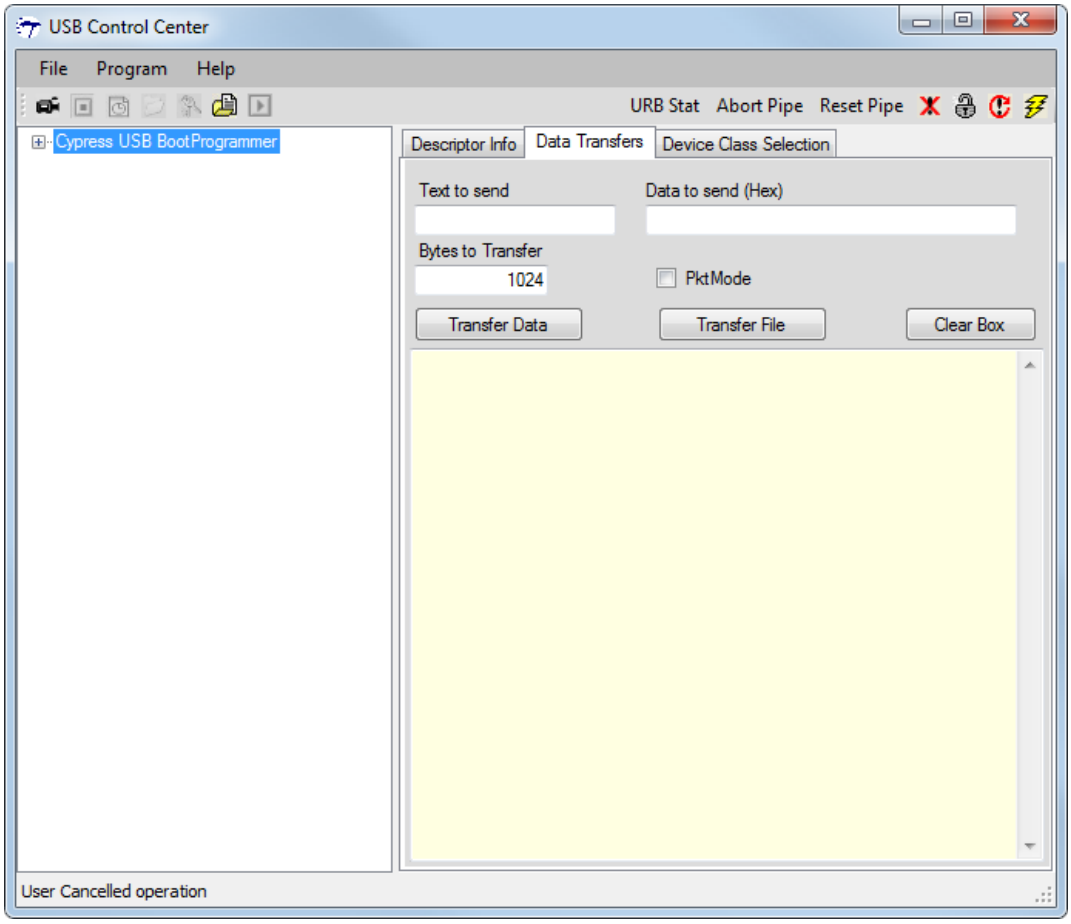


Рис.5. Отображение платы ЦОС-310-4К-АЦП в программе Cypress USB Console в режиме загрузчика.

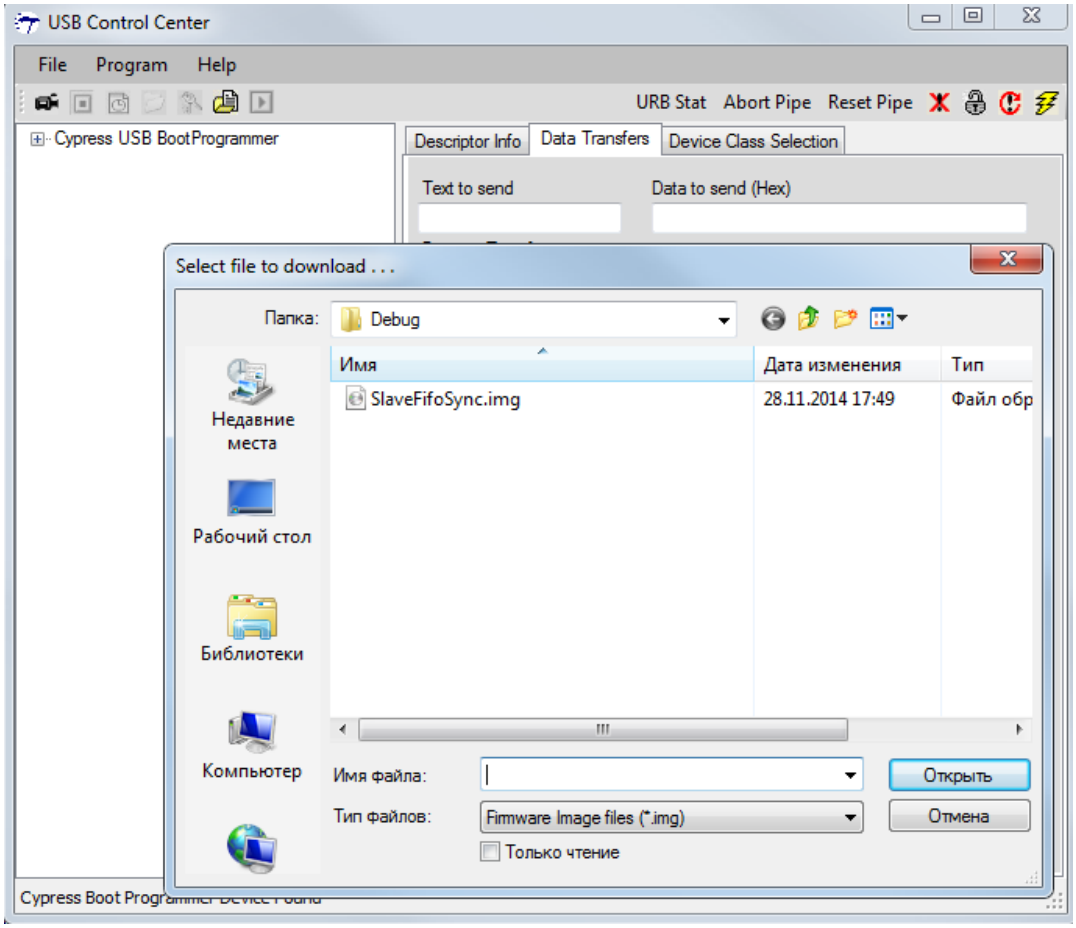


Рис.6. Выбор файла для прошивки CYUSB3014 в программе Cypress USB Console в режиме загрузчика.

Тестовый проект ПЛИС

Тестовый проект ПЛИС прилагается в архиве *ISE_14_7_test_prj.7z* и предназначен для среды Xilinx ISE версии 14.7. В тестовом проекте задействованы все основные функциональные возможности платы: работа с АЦП, взаимодействие с контроллером USB, работа с внешней памятью DDR2, программирование АЦП1, АЦП2 и синтезатора тактовых частот по SPI. Блок схема основного файла тестового проекта microb_stub.vhd приведена на рис. 7.

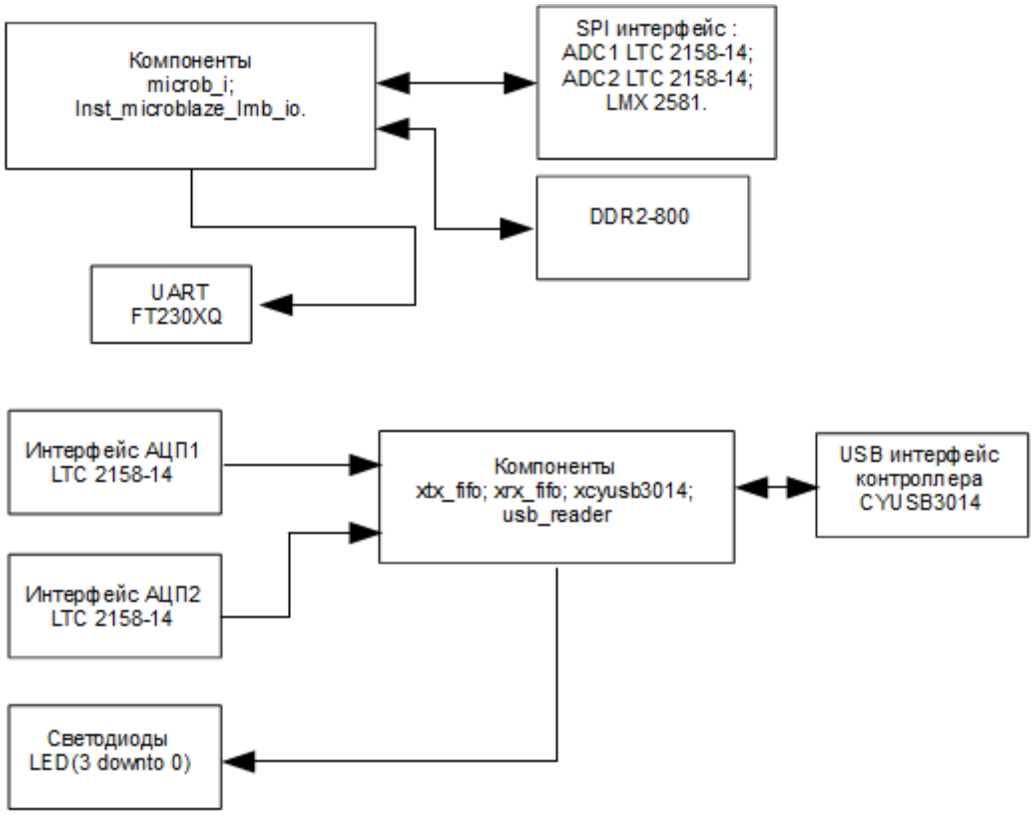


Рис.7. Блок схема тестового проекта ПЛИС.

Для работы с USB, памятью, программирования АЦП и синтезатора частот предназначены компоненты ***microb_i : entity work.microb; Inst_microblaze_lmb_io: entity work.microblaze_lmb_io.***

По включению питания они осуществляют программирование синтезатора тактовых частот LMX2581 по SPI для подачи тактовой частоты 300 МГц на АЦП1, АЦП2 и ПЛИС. Также компоненты тестируют внешнюю память DDR2 и выводят результаты в интерфейс UART.

Интерфейс контроллера USB UART подключен через ПЛИС к мосту FT230XQ. Через него результаты работы контроллера выводятся в компьютер. Скорость работы UART равна 115200 бод.

Проверка работы АЦП и USB осуществляется передачей сигнала с **канала_А, канала_В** АЦП1 и **канала_А, канала_В** АЦП2 по USB в ПК. По управляющему сигналу с контроллера начинается запись массива данных размером 4096 выборок (16384 байта, 2 канала по 16 бит). Данные для записи берутся с 14-разрядных портов ***ADC1_DA, ADC1_DB, ADC2_DA, ADC2_DB*** с частотой 300 МГц. После окончания записи массива данных в память начинается его передача в ПК через USB (Endpoint IN).

Выдачу управляющих пакетов для начала записи массива данных в память и приём массива данных по USB осуществляет тестовая программа для ПК. Таким образом, в исходном варианте тестового ПО программа для ПК будет принимать и отображать данные с **канала_А** и **канала_В** АЦП.

Компонент чтения с USB ***usb_reader*** имеет 2 выхода:

```
REG0_OUT(7 downto 0),
REG1_OUT(7 downto 0),
```

которые могут управляться по USB (посредством тестовой программы для ПК).

В данном случае младший бит порта ***REG0_OUT***, подключен к светодиоду LED3, что позволяет управлять его включением или выключением из тестовой программы для ПК путём установки в ‘0’ или ‘1’ младших битов регистра REG0. Младший бит порта ***REG1_OUT*** служит для команды записи массива выборок с АЦП для отправки по интерфейсу USB на ПК.

Светодиод ***LED0*** подключен к сигналу ***CG_MUXOUT***, который показывает состояние захвата частоты петли ФАПЧ микросхемы LMX2581. В состоянии синтеза заданного значения тактовой частоты данный сигнал равен ‘1’ и светодиод должен гореть.

Также тестовому к проекту в среде Xilinx ISE 14.7 прилагается файл ограничений kintex_green_board_0.ucf, в котором приведено соответствие номеров выводов ПЛИС и наименований сигналов (как в принципиальной схеме), необходимые стандарты ввода-вывода и ряд временных ограничений.

********** **файл ограничений kintex_green_board_0.ucf** **********

```
NET "sys_clk_p" TNM_NET = "TNM_sys_clk";
TIMESPEC TS_sys_clk = PERIOD "TNM_sys_clk" 100 MHz;

NET "sys_clk_p" LOC = AB11;
NET "sys_clk_p" IOSTANDARD = LVDS;
NET "sys_clk_p" DIFF_TERM = "TRUE";
NET "sys_clk_p" VCCAUX_IO = DONTCARE;

NET "ADC1_DCLKO_P" PERIOD = 300 MHz;
NET "ADC2_DCLKO_P" PERIOD = 300 MHz;

# Pad function: IO_L13N_T2_MRCC_33
NET "sys_clk_n" LOC = AC11;
NET "sys_clk_n" IOSTANDARD = LVDS;
NET "sys_clk_n" DIFF_TERM = "TRUE";
NET "sys_clk_n" VCCAUX_IO = DONTCARE;
NET "FPGA_RXD_IN" LOC = G20;
NET "FPGA_RXD_IN" IOSTANDARD = LVCMOS33;
NET "FPGA_TXD_OUT" LOC = G19;
NET "FPGA_TXD_OUT" IOSTANDARD = LVCMOS33;
NET "LED[0]" LOC = B26;
NET "LED[0]" IOSTANDARD = LVCMOS33;
NET "LED[1]" LOC = D25;
NET "LED[1]" IOSTANDARD = LVCMOS33;
NET "LED[2]" LOC = D26;
NET "LED[2]" IOSTANDARD = LVCMOS33;
NET "LED[3]" LOC = E25;
NET "LED[3]" IOSTANDARD = LVCMOS33;


# ADC LTC 2158-14
NET "ADC1_CSn" LOC = F24;
NET "ADC1_CSn" IOSTANDARD = LVCMOS33;
NET "ADC1_DA_N[0]" LOC = M26;
NET "ADC1_DA_N[0]" IOSTANDARD = LVDS_25;
```

```
NET "ADC1_DA_N[1]" LOC = R20;
NET "ADC1_DA_N[1]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_N[2]" LOC = P25;
NET "ADC1_DA_N[2]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_N[3]" LOC = P26;
NET "ADC1_DA_N[3]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_N[4]" LOC = T25;
NET "ADC1_DA_N[4]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_N[5]" LOC = T23;
NET "ADC1_DA_N[5]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_N[6]" LOC = U20;
NET "ADC1_DA_N[6]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[0]" LOC = N26;
NET "ADC1_DA_P[0]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[1]" LOC = T20;
NET "ADC1_DA_P[1]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[2]" LOC = R25;
NET "ADC1_DA_P[2]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[3]" LOC = R26;
NET "ADC1_DA_P[3]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[4]" LOC = T24;
NET "ADC1_DA_P[4]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[5]" LOC = T22;
NET "ADC1_DA_P[5]" IOSTANDARD = LVDS_25;
NET "ADC1_DA_P[6]" LOC = U19;
NET "ADC1_DA_P[6]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[0]" LOC = L24;
NET "ADC1_DB_N[0]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[1]" LOC = L25;
NET "ADC1_DB_N[1]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[2]" LOC = N23;
NET "ADC1_DB_N[2]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[3]" LOC = P20;
NET "ADC1_DB_N[3]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[4]" LOC = K26;
NET "ADC1_DB_N[4]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[5]" LOC = M19;
NET "ADC1_DB_N[5]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_N[6]" LOC = P18;
NET "ADC1_DB_N[6]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[0]" LOC = M24;
NET "ADC1_DB_P[0]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[1]" LOC = M25;
NET "ADC1_DB_P[1]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[2]" LOC = P23;
NET "ADC1_DB_P[2]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[3]" LOC = P19;
NET "ADC1_DB_P[3]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[4]" LOC = K25;
NET "ADC1_DB_P[4]" IOSTANDARD = LVDS_25;
NET "ADC1_DB_P[5]" LOC = N18;
NET "ADC1_DB_P[5]" IOSTANDARD = LVDS_25;
```

```
NET "ADC1_DB_P[6]" LOC = R18;
NET "ADC1_DB_P[6]" IOSTANDARD = LVDS_25;

NET "ADC1_OF_N" LOC = M22;
NET "ADC1_OF_N" IOSTANDARD = LVDS_25;
NET "ADC1_OF_P" LOC = M21;
NET "ADC1_OF_P" IOSTANDARD = LVDS_25;
NET "ADC1_SCK" LOC = F25;
NET "ADC1_SCK" IOSTANDARD = LVCMOS33;
NET "ADC1_SDI" LOC = H23;
NET "ADC1_SDI" IOSTANDARD = LVCMOS33;
NET "ADC1_SDO" LOC = H24;
NET "ADC1_SDO" IOSTANDARD = LVCMOS33;



NET "ADC1_DCLKO_N" IOSTANDARD = LVDS_25;
NET "ADC1_DCLKO_N" LOC = P21;
NET "ADC1_DCLKO_P" IOSTANDARD = LVDS_25;
NET "ADC1_DCLKO_P" LOC = R21;

# adc 2
NET "ADC2_CSn" IOSTANDARD = LVCMOS33 | LOC = G24;
NET "ADC2_DA_N<0>" IOSTANDARD = LVDS_25 | LOC = AE25;
NET "ADC2_DA_N<1>" IOSTANDARD = LVDS_25 | LOC = AE21;
NET "ADC2_DA_N<2>" IOSTANDARD = LVDS_25 | LOC = AD24;
NET "ADC2_DA_N<3>" IOSTANDARD = LVDS_25 | LOC = AC21;
NET "ADC2_DA_N<4>" IOSTANDARD = LVDS_25 | LOC = AF25;
NET "ADC2_DA_N<5>" IOSTANDARD = LVDS_25 | LOC = AF23;
NET "ADC2_DA_N<6>" IOSTANDARD = LVDS_25 | LOC = AF22;
NET "ADC2_DA_P<0>" IOSTANDARD = LVDS_25 | LOC = AD25;
NET "ADC2_DA_P<1>" IOSTANDARD = LVDS_25 | LOC = AD21;
NET "ADC2_DA_P<2>" IOSTANDARD = LVDS_25 | LOC = AD23;
NET "ADC2_DA_P<3>" IOSTANDARD = LVDS_25 | LOC = AB21;
NET "ADC2_DA_P<4>" IOSTANDARD = LVDS_25 | LOC = AF24;
NET "ADC2_DA_P<5>" IOSTANDARD = LVDS_25 | LOC = AE23;
NET "ADC2_DA_P<6>" IOSTANDARD = LVDS_25 | LOC = AE22;
NET "ADC2_DB_N<0>" IOSTANDARD = LVDS_25 | LOC = W26;
NET "ADC2_DB_N<1>" IOSTANDARD = LVDS_25 | LOC = V24;
NET "ADC2_DB_N<2>" IOSTANDARD = LVDS_25 | LOC = Y26;
NET "ADC2_DB_N<3>" IOSTANDARD = LVDS_25 | LOC = W24;
NET "ADC2_DB_N<4>" IOSTANDARD = LVDS_25 | LOC = AC26;
NET "ADC2_DB_N<5>" IOSTANDARD = LVDS_25 | LOC = AB25;
NET "ADC2_DB_N<6>" IOSTANDARD = LVDS_25 | LOC = AE26;
NET "ADC2_DB_P<0>" IOSTANDARD = LVDS_25 | LOC = W25;
NET "ADC2_DB_P<1>" IOSTANDARD = LVDS_25 | LOC = V23;
NET "ADC2_DB_P<2>" IOSTANDARD = LVDS_25 | LOC = Y25;
NET "ADC2_DB_P<3>" IOSTANDARD = LVDS_25 | LOC = W23;
NET "ADC2_DB_P<4>" IOSTANDARD = LVDS_25 | LOC = AB26;
NET "ADC2_DB_P<5>" IOSTANDARD = LVDS_25 | LOC = AA25;
NET "ADC2_DB_P<6>" IOSTANDARD = LVDS_25 | LOC = AD26;
NET "ADC2_DCLKO_N" IOSTANDARD = LVDS_25 | LOC = AA22;
```

```
NET "ADC2_DCLKO_P" IOSTANDARD = LVDS_25 | LOC = Y22;
NET "ADC2_OF_N" IOSTANDARD = LVDS_25 | LOC = V26;
NET "ADC2_OF_P" IOSTANDARD = LVDS_25 | LOC = U26;
NET "ADC2_SCK" IOSTANDARD = LVCMOS33 | LOC = H21;
NET "ADC2_SDI" IOSTANDARD = LVCMOS33 | LOC = J25;
NET "ADC2_SDO" IOSTANDARD = LVCMOS33 | LOC = J24;

# SPI CLOCK GEN LMX2581
NET "CG_DATA" LOC = G26;
NET "CG_DATA" IOSTANDARD = LVCMOS33;
NET "CG_LE" LOC = E26;
NET "CG_LE" IOSTANDARD = LVCMOS33;
NET "CG_MUXOUT" LOC = J26;
NET "CG_MUXOUT" IOSTANDARD = LVCMOS33;
NET "CG_SCLK" LOC = H26;
NET "CG_SCLK" IOSTANDARD = LVCMOS33;


# CYUSB 3014. GPIO
#GPIO[0 - 15]
# GPIO[0]
NET "bCY_FDATA[0]" LOC = C12;
NET "bCY_FDATA[0]" IOSTANDARD = LVCMOS33;
# GPIO[1]
NET "bCY_FDATA[1]" LOC = E11;
NET "bCY_FDATA[1]" IOSTANDARD = LVCMOS33;
# GPIO[2]
NET "bCY_FDATA[2]" LOC = F9;
NET "bCY_FDATA[2]" IOSTANDARD = LVCMOS33;
# GPIO[3]
NET "bCY_FDATA[3]" LOC = C11;
NET "bCY_FDATA[3]" IOSTANDARD = LVCMOS33;
# GPIO[4]
NET "bCY_FDATA[4]" LOC = D9;
NET "bCY_FDATA[4]" IOSTANDARD = LVCMOS33;
# GPIO[5]
NET "bCY_FDATA[5]" LOC = F8;
NET "bCY_FDATA[5]" IOSTANDARD = LVCMOS33;
# GPIO[6]
NET "bCY_FDATA[6]" LOC = D10;
NET "bCY_FDATA[6]" IOSTANDARD = LVCMOS33;
# GPIO[7]
NET "bCY_FDATA[7]" LOC = C9;
NET "bCY_FDATA[7]" IOSTANDARD = LVCMOS33;
# GPIO[8]
NET "bCY_FDATA[8]" LOC = C14;
NET "bCY_FDATA[8]" IOSTANDARD = LVCMOS33;
# GPIO[9]
NET "bCY_FDATA[9]" LOC = D14;
NET "bCY_FDATA[9]" IOSTANDARD = LVCMOS33;
# GPIO[10]
NET "bCY_FDATA[10]" LOC = A15;
```

```
NET "bCY_FDATA[10]" IOSTANDARD = LVCMOS33;
# GPIO[11]
NET "bCY_FDATA[11]" LOC = A14;
NET "bCY_FDATA[11]" IOSTANDARD = LVCMOS33;
# GPIO[12]
NET "bCY_FDATA[12]" LOC = B15;
NET "bCY_FDATA[12]" IOSTANDARD = LVCMOS33;
# GPIO[13]
NET "bCY_FDATA[13]" LOC = D13;
NET "bCY_FDATA[13]" IOSTANDARD = LVCMOS33;
# GPIO[14]
NET "bCY_FDATA[14]" LOC = F13;
NET "bCY_FDATA[14]" IOSTANDARD = LVCMOS33;
# GPIO[15]
NET "bCY_FDATA[15]" LOC = D8;
NET "bCY_FDATA[15]" IOSTANDARD = LVCMOS33;


# GPIO[16]
NET "bCY_CLK100" LOC = E10;
NET "bCY_CLK100" IOSTANDARD = LVCMOS33;
# GPIO[17]
NET "bCY_SLCS" LOC = B14;

# GPIO[18]
NET "bCY_SLWR" IOSTANDARD = LVCMOS33;
NET "bCY_SLWR" LOC = D11;
# GPIO[19]
NET "bCY_SLOE" LOC = E12;
NET "bCY_SLOE" IOSTANDARD = LVCMOS33;
# GPIO[20]
NET "bCY_SLRD" LOC = F12;
NET "bCY_SLRD" IOSTANDARD = LVCMOS33;
# GPIO[21]
NET "bCY_FULL" LOC = F10;
NET "bCY_FULL" IOSTANDARD = LVCMOS33;
# GPIO[22]
NET "bCY_ALMOST_FULL" LOC = G10;
NET "bCY_ALMOST_FULL" IOSTANDARD = LVCMOS33;
# GPIO[23]
NET "bCY_EMPTY" LOC = A12;
NET "bCY_EMPTY" IOSTANDARD = LVCMOS33;
# GPIO[24]
NET "bCY_PKTEND" LOC = F14;
NET "bCY_PKTEND" IOSTANDARD = LVCMOS33;
# GPIO[25]
NET "bCY_ALMOST_EMPTY" LOC = H12;
NET "bCY_ALMOST_EMPTY" IOSTANDARD = LVCMOS33;



#GPIO[28 - 29]
```

```
# GPIO[28]
NET "bCY_FADDR[0]" LOC = G14;
NET "bCY_FADDR[0]" IOSTANDARD = LVCMOS33;
# GPIO[29]
NET "bCY_FADDR[1]" LOC = H13;
NET "bCY_FADDR[1]" IOSTANDARD = LVCMOS33;
#GPIO[33 - 44]
# GPIO[33]
NET "bCY_FDATA[16]" LOC = B11;
NET "bCY_FDATA[16]" IOSTANDARD = LVCMOS33;
# GPIO[34]
NET "bCY_FDATA[17]" LOC = H14;
NET "bCY_FDATA[17]" IOSTANDARD = LVCMOS33;
# GPIO[35]
NET "bCY_FDATA[18]" LOC = A10;
NET "bCY_FDATA[18]" IOSTANDARD = LVCMOS33;
# GPIO[36]
NET "bCY_FDATA[19]" LOC = B10;
NET "bCY_FDATA[19]" IOSTANDARD = LVCMOS33;
# GPIO[37]
NET "bCY_FDATA[20]" LOC = J8;
NET "bCY_FDATA[20]" IOSTANDARD = LVCMOS33;
# GPIO[38]
NET "bCY_FDATA[21]" LOC = A9;
NET "bCY_FDATA[21]" IOSTANDARD = LVCMOS33;
# GPIO[39]
NET "bCY_FDATA[22]" LOC = J14;
NET "bCY_FDATA[22]" IOSTANDARD = LVCMOS33;
# GPIO[40]
NET "bCY_FDATA[23]" LOC = H9;
NET "bCY_FDATA[23]" IOSTANDARD = LVCMOS33;
# GPIO[41]
NET "bCY_FDATA[24]" LOC = H11;
NET "bCY_FDATA[24]" IOSTANDARD = LVCMOS33;
# GPIO[42]
NET "bCY_FDATA[25]" LOC = B9;
NET "bCY_FDATA[25]" IOSTANDARD = LVCMOS33;
# GPIO[43]
NET "bCY_FDATA[26]" LOC = J10;
NET "bCY_FDATA[26]" IOSTANDARD = LVCMOS33;
# GPIO[44]
NET "bCY_FDATA[27]" LOC = J11;
NET "bCY_FDATA[27]" IOSTANDARD = LVCMOS33;
#GPIO[46 - 49]
# GPIO[46]
NET "bCY_FDATA[28]" LOC = G11;
NET "bCY_FDATA[28]" IOSTANDARD = LVCMOS33;
# GPIO[47]
NET "bCY_FDATA[29]" LOC = A8;
NET "bCY_FDATA[29]" IOSTANDARD = LVCMOS33;
# GPIO[48]
NET "bCY_FDATA[30]" LOC = G9;
```

```
NET "bCY_FDATA[30]" IOSTANDARD = LVCMOS33;
# GPIO[49]
NET "bCY_FDATA[31]" LOC = H8;
NET "bCY_FDATA[31]" IOSTANDARD = LVCMOS33;

# # GPIO[48]
# NET "bCY_UART_TX" LOC = G9;
# NET "bCY_UART_TX" IOSTANDARD = LVCMOS33;
# # GPIO[49]
# NET "bCY_UART_RX" LOC = H8;
# NET "bCY_UART_RX" IOSTANDARD = LVCMOS33;

NET "bCY_INT" LOC = A13;
NET "bCY_INT" IOSTANDARD = LVCMOS33;
NET "bCY_SLCS" IOSTANDARD = LVCMOS33;

NET "bCY_GPIO_26" LOC = G12;
NET "bCY_GPIO_26" IOSTANDARD = LVCMOS33;
NET "bCY_GPIO_27" LOC = B12;
NET "bCY_GPIO_27" IOSTANDARD = LVCMOS33;

NET "bCY_RESETn" IOSTANDARD = LVCMOS33 | LOC = C13;

# ограничение временной задержки для контроллера USB
# TIMEGRP CYFlagPads = PADS (bCY_ALMOST_EMPTY | bCY_ALMOST_FULL | bCY_FULL |
bCY_EMPTY);
# TIMEGRP CYCtrlPads = PADS (bCY_SLWR);
# TIMESPEC TS_CY_P2P_DELAY = FROM : CYFlagPads  : TO : CYCtrlPads  : 5 ns;

# X15
NET "CONN_AUX_FR_STROBE" IOSTANDARD = LVCMOS33 | LOC = K17;
NET "CONN_AUX_SPI_OEn" IOSTANDARD = LVCMOS33 | LOC = J18;
NET "CONN_AUX_START" IOSTANDARD = LVCMOS33 | LOC = L18;
NET "CONN_AUX_SYNC_IN" IOSTANDARD = LVCMOS18 | LOC = U1;
NET "CONN_AUX_TX_PAUSE" IOSTANDARD = LVCMOS33 | LOC = K18;
# 74LVC595APW SPI interface
NET "CONN_AUX_SPI_CLK" IOSTANDARD = LVCMOS33 | LOC = H18;
NET "CONN_AUX_SPI_DATA" IOSTANDARD = LVCMOS33 | LOC = J19;
NET "CONN_AUX_SPI_LE" IOSTANDARD = LVCMOS33 | LOC = H19;

# CONN X1 X2
NET "CONN<0>" IOSTANDARD = LVCMOS33 | LOC = E15;
NET "CONN<1>" IOSTANDARD = LVCMOS33 | LOC = F17;
NET "CONN<2>" IOSTANDARD = LVCMOS33 | LOC = F18;
NET "CONN<3>" IOSTANDARD = LVCMOS33 | LOC = F19;
NET "CONN<4>" IOSTANDARD = LVCMOS33 | LOC = E21;
NET "CONN<5>" IOSTANDARD = LVCMOS33 | LOC = E23;
NET "CONN<6>" IOSTANDARD = LVCMOS33 | LOC = D16;
NET "CONN<7>" IOSTANDARD = LVCMOS33 | LOC = C17;
NET "CONN<8>" IOSTANDARD = LVCMOS33 | LOC = C18;
NET "CONN<9>" IOSTANDARD = LVCMOS33 | LOC = C19;
NET "CONN<10>" IOSTANDARD = LVCMOS33 | LOC = D21;
```

```
NET "CONN<11>" IOSTANDARD = LVCMOS33 | LOC = C22;
NET "CONN<12>" IOSTANDARD = LVCMOS33 | LOC = C24;
NET "CONN<13>" IOSTANDARD = LVCMOS33 | LOC = B17;
NET "CONN<14>" IOSTANDARD = LVCMOS33 | LOC = A18;
NET "CONN<15>" IOSTANDARD = LVCMOS33 | LOC = B19;
NET "CONN<16>" IOSTANDARD = LVCMOS33 | LOC = A19;
NET "CONN<17>" IOSTANDARD = LVCMOS33 | LOC = A17;
NET "CONN<18>" IOSTANDARD = LVCMOS33 | LOC = B16;
NET "CONN<19>" IOSTANDARD = LVCMOS33 | LOC = D24;
NET "CONN<20>" IOSTANDARD = LVCMOS33 | LOC = C21;
NET "CONN<21>" IOSTANDARD = LVCMOS33 | LOC = D20;
NET "CONN<22>" IOSTANDARD = LVCMOS33 | LOC = D19;
NET "CONN<23>" IOSTANDARD = LVCMOS33 | LOC = D18;
NET "CONN<24>" IOSTANDARD = LVCMOS33 | LOC = C16;
NET "CONN<25>" IOSTANDARD = LVCMOS33 | LOC = D15;
NET "CONN<26>" IOSTANDARD = LVCMOS33 | LOC = E22;
NET "CONN<27>" IOSTANDARD = LVCMOS33 | LOC = F20;
NET "CONN<28>" IOSTANDARD = LVCMOS33 | LOC = E18;
NET "CONN<29>" IOSTANDARD = LVCMOS33 | LOC = E17;
NET "CONN<30>" IOSTANDARD = LVCMOS33 | LOC = E16;
NET "CONN<31>" IOSTANDARD = LVCMOS33 | LOC = F15;
NET "CONN<32>" IOSTANDARD = LVCMOS33 | LOC = A20;
NET "CONN<33>" IOSTANDARD = LVCMOS33 | LOC = B20;
NET "CONN<34>" IOSTANDARD = LVCMOS33 | LOC = B21;
NET "CONN<35>" IOSTANDARD = LVCMOS33 | LOC = A22;
NET "CONN<36>" IOSTANDARD = LVCMOS33 | LOC = B22;
NET "CONN<37>" IOSTANDARD = LVCMOS33 | LOC = A23;
NET "CONN<38>" IOSTANDARD = LVCMOS33 | LOC = A24;
NET "CONN<39>" IOSTANDARD = LVCMOS33 | LOC = C26;
NET "sys_rst" IOSTANDARD = LVCMOS33 | LOC = F23;


# DDR2
NET "axi_7series_ddrx_0_ddr_addr_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[2]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[3]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[4]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[5]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[6]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[7]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[8]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[9]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[10]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[11]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[12]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_addr_pin[13]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ba_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ba_pin[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ba_pin[2]" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_cas_n_pin" VCCAUX_IO = NORMAL;
```

```
NET "axi_7series_ddrx_0_ddr_ras_n_pin" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_we_n_pin" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_ck_n_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ck_n_pin[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ck_p_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_ck_p_pin[1]" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_cke_pin" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_dm_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dm_pin[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dm_pin[2]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dm_pin[3]" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_dq[0]" LOC = AB2;
NET "axi_7series_ddrx_0_ddr_dq[0]" IOSTANDARD = SSTL18_II_T_DCI;
NET "axi_7series_ddrx_0_ddr_dq[0]" SLEW = FAST;
NET "axi_7series_ddrx_0_ddr_dq[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[2]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[3]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[4]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[5]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[6]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[7]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[8]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[9]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[10]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[11]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[12]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[13]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[14]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[15]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[16]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[17]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[18]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[19]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[20]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[21]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[22]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[23]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[24]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[25]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[26]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[27]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[28]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[29]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[30]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dq[31]" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_dqs_n[0]" VCCAUX_IO = NORMAL;
```

```
NET "axi_7series_ddrx_0_ddr_dqs_n[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_n[2]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_n[3]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_p[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_p[1]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_p[2]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_dqs_p[3]" VCCAUX_IO = NORMAL;

NET "axi_7series_ddrx_0_ddr_odt_pin[0]" VCCAUX_IO = NORMAL;
NET "axi_7series_ddrx_0_ddr_odt_pin[1]" VCCAUX_IO = NORMAL;
```

********** окончание kintex_green_board_0.ucf **********

Тестовая программа ПК

Тестовая программа для ПК находится в архиве *BCB_prog.7z*.

Для запуска тестовой программы её необходимо разархивировать и запустить файл Usb_test.exe. При подключённой плате ЦОС-310-4К-АЦП к USB порту компьютера, поданном питании и правильно установленных драйверах контроллера CYUSB3014 внешний вид окна программы показан на рис.8. Программа автоматически определяет наличие платы ЦОС.

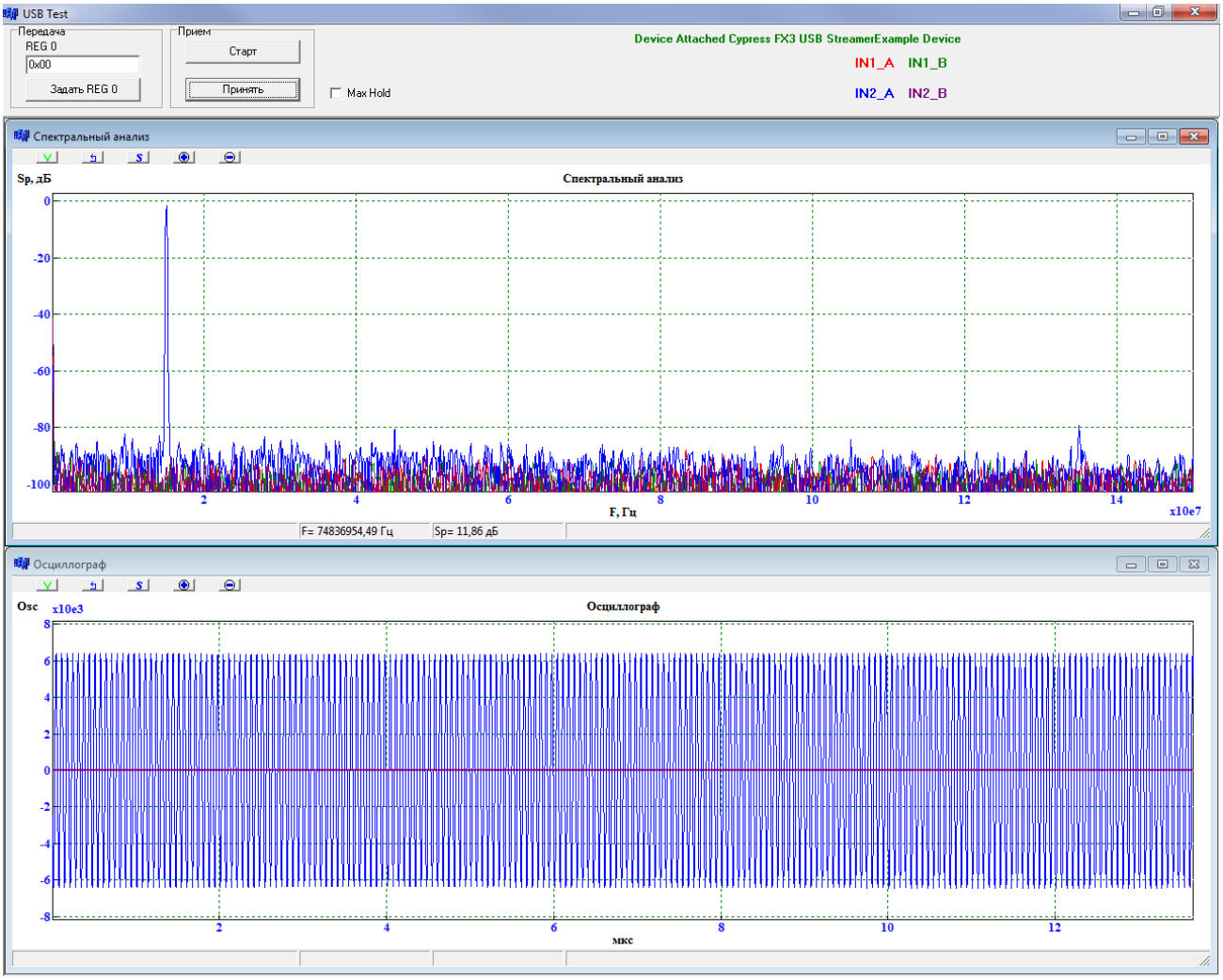


*Рис.8. Внешний вид окна тестовой программы при приёме массивов данных с платы ЦОС-310-4К-АЦП. На **канал_А** АЦП2 подан гармонический сигнал частотой 315 МГц и амплитудой 7dBm с генератора внешним кабелем. Число выборок данных в массиве 4096.*

В случае, если плата ЦОС успешно идентифицирована программой, то для начала периодической отсылки управляющих пакетов для записи и передачи массивов данных на ПК необходимо нажать кнопку “Старт”. Для остановки приема необходимо нажать кнопку “Стоп”. Для приема одного массива данных необходимо нажать кнопку “Принять”. Внешний вид окна тестовой программы при приёме массивов данных с платы ЦОС-310-4К-АЦП показан на рис.8. Тестовая программа отображает осциллограмму и спектр принятого сигнала.

Также тестовая программа для ПК может записывать данные в 2 внутренних 16-разрядных регистра REG0, REG1, которые имеются в тестовой прошивке ПЛИС. Посылка данных в регистр в REG0 осуществляется с помощью полей для ввода данных. Запись в младший бит регистра REG1 осуществляется каждый раз при приеме данных и служит командой начала записи с АЦП.

Компонент ***reader*** имеет 2 выхода:

REG0_OUT(7 downto 0),
REG1_OUT(7 downto 0),

которые являются выходами этих регистров.

Чтобы записать байт данных в регистр нужно записать его в шестнадцатеричном коде с префиксом 0x(например 0x00F5) в соответствующем поле вверху программы и нажать кнопку “Задать REG 0” для передачи данных по интерфейсу USB в плату ЦОС. Например, для записи числа 25000 в регистр REG0, нужно написать 0x61A8 в поле REG0 и нажать кнопку “Задать REG 0”.

Драйвер контроллера FX3

Драйвера прилагаются в архиве cyusb_driver.7z. Для изменения названия устройства в ОС Windows и в случае изменения VID/PID устройства, необходимо отредактировать *.inf файл в папке с драйвером. В строках с существующим VID/PID необходимо написать новое имя устройства, или добавить строчку с новым VID/PID и наименованием устройства.