

# Разработка IP-блока с помощью инструментов высокоуровнего синтеза: HLS

Часть 3

Автор: PointPas

Рецензент: KeisN13





#### Оглавление

Аннотация	. 3
Работаем с IP-блоками HLS в Xilinx SDK	. 3
Шаг 1: Экспортируем файлы в SDK	. 3
Шаг 2: Подготовка к работе с IP	. 4
Шаг 3: Разработка ПО и запуск на системе с ZYNQ	. 7
Заключение	. 9
Список литературы	. 9





#### Аннотация

В этой части будут показаны основные шаги необходимые для запуска и работы разработанной в прошлых частях системы [1, 2]. Будет показано, как можно использовать API, которое автоматически генерируется для IP-блоков с интерфейсом AXI4-Lite.

# Работаем с IP-блоками HLS в Xilinx SDK

### Шаг 1: Экспортируем файлы в SDK

Начнем с экспорта в SDK. Для этого в Vivado идем в меню File→Export→Export Hardware (Рисунок 1). Отмечаем галочку "Include bitstream". Затем снова идем в меню File→Launch SDK.



Рисунок 1 – Экспорт в SDK





#### Шаг 2: Подготовка к работе с ІР

Для начала убедимся, что мы все правильно настроили. Для этого сделаем новый проект с "Hello World". Идем File→New→Application Project вводим название, какое хотите, больше ничего не заполняем (Рисунок 2). Нажимаем "Next", выбираем "Hello World" и нажимаем "Finish". На отладочной плате MiniZed на преобразователь UART→USB выведен UART1 процессорной системы. Нужно выбрать именно его, для этого идем в папку Hello\_bsp (либо ваше название вместо Hello) нажимаем модифицировать BSP (Рисунок 3). Выбираем UART1 (Рисунок 4). Нажимаем "OK".

New Project		
<b>Application Project</b> Create a managed mak	e application project.	G
Project name: Hello		
Use default locatio	n	
Location: D:\XilinxPrj	PWM_Minized\PWM_Minized.sdk\Hello	Browse
Choose file s	ystem: default 💌	
OS Platform: standa	one	
Target Hardware		
Hardware Platform:	Subsys_wrapper_hw_platform_0	▼ New
Processor	ns7 corteva0 0	
. Totesson	ps/_conceas_o	
Target Software		
Language:		
Compiler:	32-bit	
Human ison Guarta		
Record Support Dealer	Crosto Now Halla han	
воаго заррон Раска		
	O Use existing	· · ·
?	< Back Next > Finis	h Cancel

Рисунок 2 - Создаем проект для проверки



Рисунок 3 – Модифицируем BSP

<sup>sok</sup> Board Support Package Sett	tings		
Board Support Package Set Control various settings of yo	<b>ttings</b> our Board Support Package.		
<ul> <li>Overview</li> <li>standalone</li> <li>drivers</li> <li>ps7_cortexa9_0</li> </ul>	Configuration for OS: standal	one Value	Def
	hypervisor_guest lockstep_mode_debug sleep_timer	false false	fals fals
	stdin stdout ttc select cntr	ps7_uart_1 ps7_uart_1 7	r or r or

www.FPGA-Systems.ru

Рисунок 4 – Выбираем UART1

Подключаем отладочную плату к компьютеру. Нажимаем Ctrl + B (Build Project). Идем во вкладку "SDK Terminal" и нажимаем на кнопку добавления порта. В открывшемся окне нужно указать COM порт, к которому подключен USB UART преобразователь и скорость обмена данными (Рисунок 5). Узнать номер порта можно в диспетчере устройств во вкладке COM и LPT.



Рисунок 5 – Добавляем соединение и настраиваем параметры

Теперь можно прошить нашу СнК. Идем в меню Run -> Run Configurations. Заполняем как на рисунке (Рисунок 6). Нажимаем "Run".

w	You Tube	f	
---	-------------	---	--



Run or Debug a program using System	Debugger.			
	Name: System Debugge	r using Debug_Hello.elf on Local		
type filter text  Performance Analysis  Target Communication Frami  Kilinx C/C++ Application (GD)  Kilinx C/C++ application (Syst)  Kilinx C/C++ application (Syst)	<ul> <li>Target Setup</li> <li>Debug Type: Standalo</li> <li>Connection: Local</li> </ul>	Application 🕬= Arguments 🚾 Environn ne Application Debug 👻 👻 New	nent 🔤 Symbol Files	Ey Source ≫2
	Hardware Platform: Bitstream File: Initialization File: FPGA Device: PS Device:	Subsys_wrapper_hw_platform_0   Subsys_wrapper.bit ps7_init.tcl Auto Detect Auto Detect	Search Search Select Select	Browse Browse
	<ul> <li>✓ Reset entire system</li> <li>✓ Program FPGA</li> <li>✓ Run ps7_init</li> <li>✓ Run ps7_post_conf</li> </ul>	Summary of operations to be performed         Following operations will be performed before launching the debugger.         1. Resets entire system. Clears the FPGA fabric (PL).         2. Program FPGA fabric (PL).         3. Runs ps7_init to initialize PS.         4. Runs ps7_post_config. Enables level shifters from PL to PS. (Recommended to use this option only after system reset or board power ON).         5. All processors in the system vill be suspended, and Applications will be download to the following processors as specified in the Applications tab.         1) ps7_cortexa9_0         (D:\XilinxPrj\PWM_Minized\PWM_Minized.sdk\Hello\Debug\Hello.elf)		
← III ► Filter matched 6 of 9 items			[	Revert Apply
?				Run Close

Рисунок 6 – Окно "Run Configurations"

Переключаемся на вкладку "SDK Terminals". Мы должны увидеть сообщение "Hello World" (Рисунок 7).

Problems	Tasks	📃 Console	Properties	📃 SDK Terminal 🔀	-	×	<u> </u>	
Connected to	: Serial ( CC	OM11, 115200,	0, 8)					
Connected to Hello World	o COM11 at	115200						*
								-
							•	
						Sei	nd	lear

Рисунок 7 – Вывод консоли





#### Шаг 3: Разработка ПО и запуск на системе с ZYNQ

Открываем файл helloword.c

Добавляем заголовочные файлы, которые нам понадобятся

```
#include <stdio.h>
/*include libraries from Xilinx*/
#include "platform.h"
#include "xpwm_ctrl.h" // Device driver for HLS HW block
#include "xparameters.h" // Parameter definitions for processor peripherals
#include "sleep.h"
```

 Добавим строку #define PWM\_TERMINAL Ø. Наше ПО можно будет собрать в двух исполнениях. В первом исполнении управлять яркостью светодиода будем сами, отправляя ему значение рабочего цикла с помощью консоли. Во втором яркость будет меняться от 0 до 100% автоматически (PWM\_TERMINAL 0).

Напишем функцию, которая будет инициализировать наш IP-блок в PL части.

```
int PWM_init(XPwm_ctrl *PwmPtr){
    XPwm_ctrl_Config *CfgPwmPtr;
    int status;
    CfgPwmPtr = XPwm_ctrl_LookupConfig(XPAR_XPWM_CTRL_0_DEVICE_ID);
    if (!CfgPwmPtr) {
        print("ERROR: Configuration failed. DeviceId is not found. \n\r");
        return XST_FAILURE;
    }
    status = XPwm_ctrl_Initialize(PwmPtr, CfgPwmPtr->DeviceId);
    if (status != XST_SUCCESS) {
        print("ERROR: Could not initialize device.\n\r");
    return XST_FAILURE;
    }
    return xST_FAILURE;
}
```

- 2. Функции и типы данных используемые для написания функции инициализации в этом пункте были автоматически сгенерированы Vivado HLS и доступны при подключении заголовочного файла "xpwm\_ctrl.h".
- 3. Дополним main(), используя написанную ранее функцию и API сгенерированное Vivado HLS.

```
int main()
{
    XPwm_ctrl PWM_CTRL;
    int status;
    u32 EN = 1;
    u32 Rst = 0;
    u32 NumOfTicks = 65535/100; //the number of ticks in 1%
#ifPWM_TERMINAL
    u32 LoadValPer;
#endif
    u32 LoadVal;
    init_platform();
```





```
print("Program to test communication with HLS PWM peripheral in PL\n\r");
   //check
   status = PWM_init(&PWM_CTRL);
   if(status != XST_SUCCESS){
         print("HLS peripheral setup failed\n\r");
   } else {
         print("HLS peripheral check done\n\r");
   }
   printf("XPwm ctrl IsReady = %x\n\r", XPwm ctrl IsReady(&PWM CTRL));
   XPwm_ctrl_Set_EN_V(&PWM_CTRL, EN); //enable counter
   XPwm_ctrl_Set_Rst_V(&PWM_CTRL, Rst);//assert soft rst to low
   while(1){
#ifPWM TERMINAL
     print("Enter Duty cycle in percents \n\r");
     scanf("%d", &LoadValPer);
     LoadVal = LoadValPer*NumOfTicks; //get the number of ticks
     printf("LoadVal = %d\n\r", LoadVal);
     XPwm_ctrl_Set_LoadValCnt_V(&PWM_CTRL, LoadVal);
#else
     for(int i = 0; i<=100; i++){</pre>
     printf("LoadVal = %d\n\r", i);
     XPwm_ctrl_Set_LoadValCnt_V(&PWM_CTRL, i*NumOfTicks);
     usleep(100000); //0.1 second delay
   }
#endif
}
   cleanup_platform();
   return 0;
}
```

4. Сохраняем файл. При сохранении SDK автоматически пересоберет проект.

Прошиваем нашу СнК, т.к. мы уже настроили параметры запуска при запуске "Hello World", то можно просто щелкнуть право кнопкой мыши по проекту→Run As→Launch On Hardware (System Debugger) (Рисунок 8). Если вы отключали плату от компьютера, то необходимо заново подключиться в терминале. После прошивки мы должны увидеть, как яркость светодиода изменяется от 0 до 100%. Чтобы менять яркость вручную, нужно пересобрать проект, изменив значение с 0 на 1 для РWM\_TERMINAL.





Рисунок 8 – Прошиваем СнК

## Заключение

Вы прочитали третью и заключительную часть по разработке IP-блоков с помощью инструментов высокоуровневого синтеза. В этом цикле был рассмотрен маршрут проектирования в среде VIvado HLS, интеграция разработанных IP-блоков в систему на базе процессорной системы ZYNQ в Vivado и работа с получившейся системой в SDK. Многое не было рассмотрено, но целью не ставилось рассказать обо всем. Целью было показать основные этапы и подход к такой разработке.

#### Список литературы

- 1. Разработка IP-блока с помощью инструментов высокоуровнего синтеза: HLS. <u>Часть 1</u>.
- 2. Разработка IP-блока с помощью инструментов высокоуровнего синтеза: HLS. <u>Часть 2</u>
- 3. MiniZed<sup>TM</sup>: a single-core Zynq 7Z007S development board
- 4. Осваиваем Zynq-7000S с бесплатной отладкой: видео
- 5. Осваиваем Zynq-7000: видеоуроки (доп. ссылка)
- 6. Добавление MiniZed в Vivado: архив с инструкцией
- 7. Vivado Design Suite Evaluation and WebPACK // Ссылка
- 8. <u>UG973</u>. Release Notes, Installation, and Licensing

www.FPGA-Systems.ru