От инженеров для инженеров

FPGA-Systems.ru

Разработка IP-блока с помощью инструментов высокоуровнего синтеза: HLS

Часть 3

Автор: **PointPas**

Рецензент: KeisN13

2019 г.

Оглавление

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.

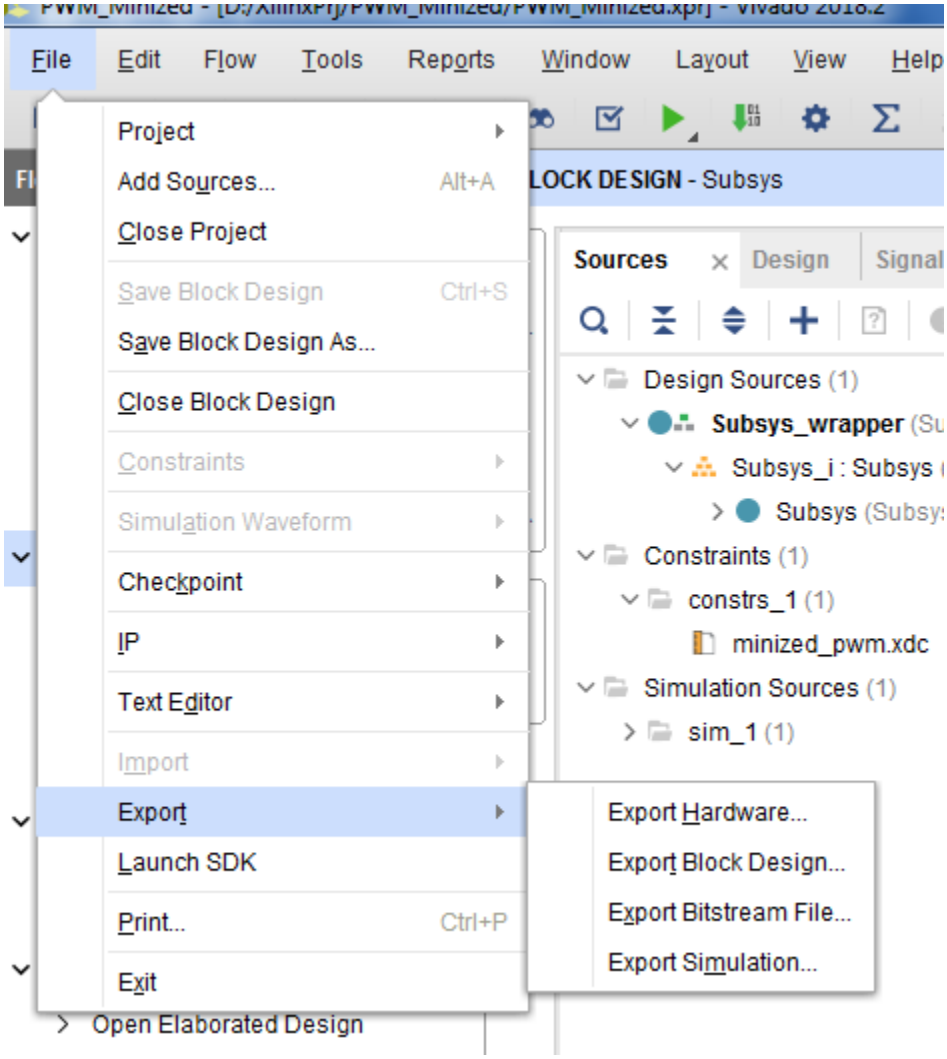

Рисунок 1 – Экспорт в SDK

Шаг 2: Подготовка к работе с IP

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”.

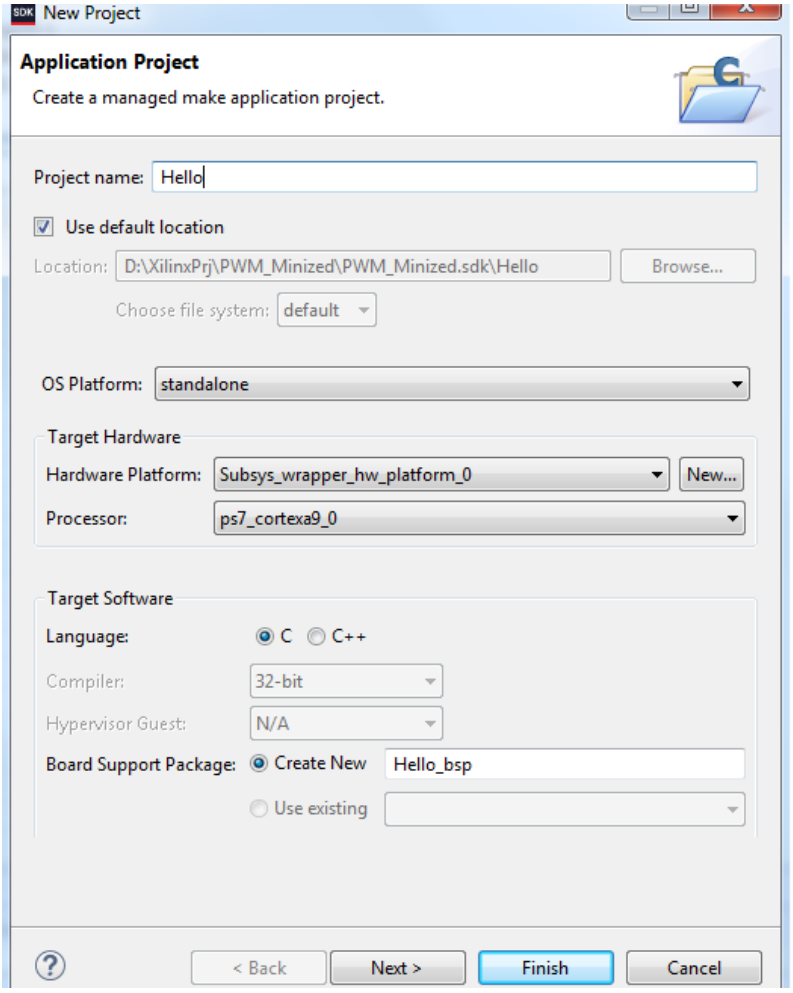

Рисунок 2 – Создаем проект для проверки

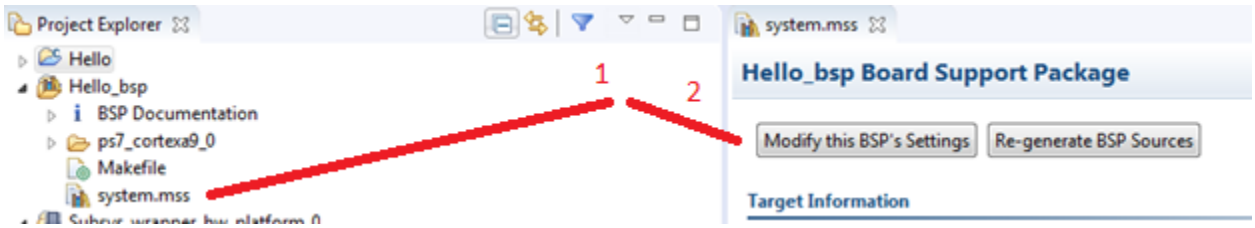

Рисунок 3 – Модифицируем BSP

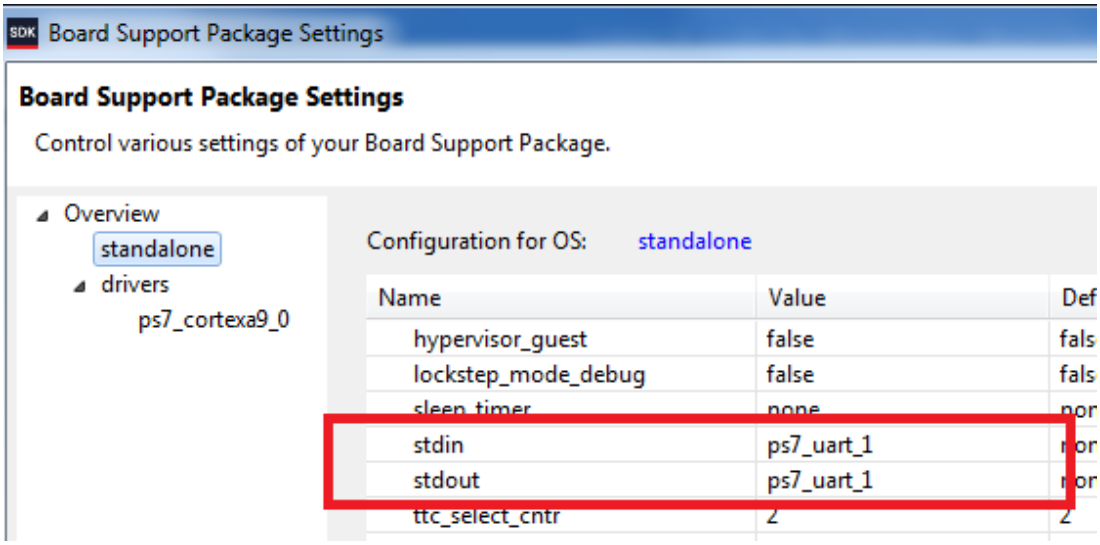

Рисунок 4 – Выбираем UART1

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.

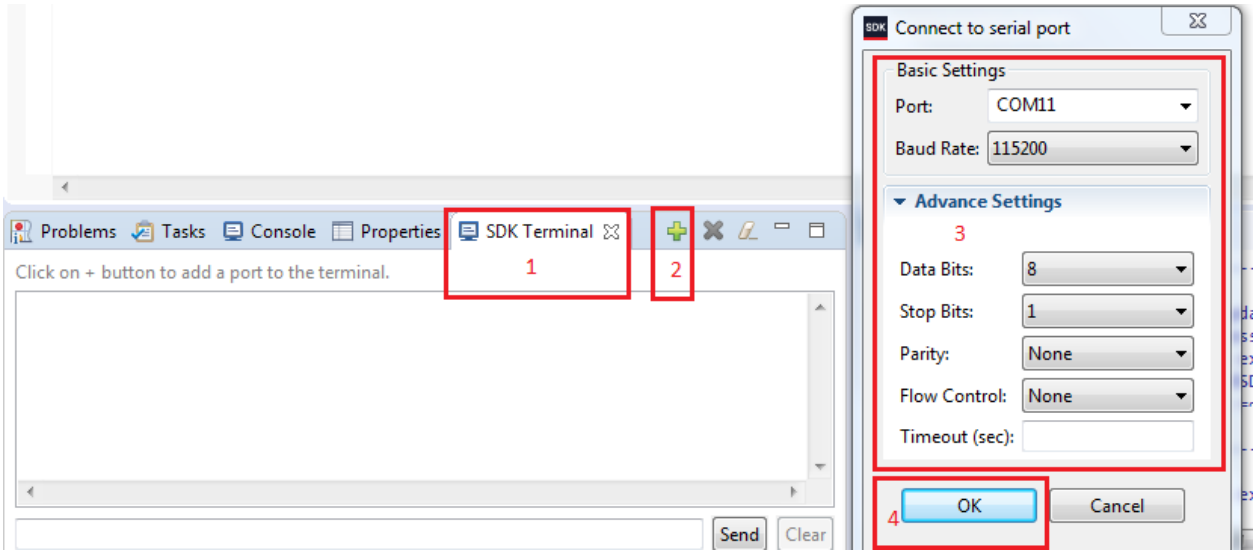

Рисунок 5 – Добавляем соединение и настраиваем параметры

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

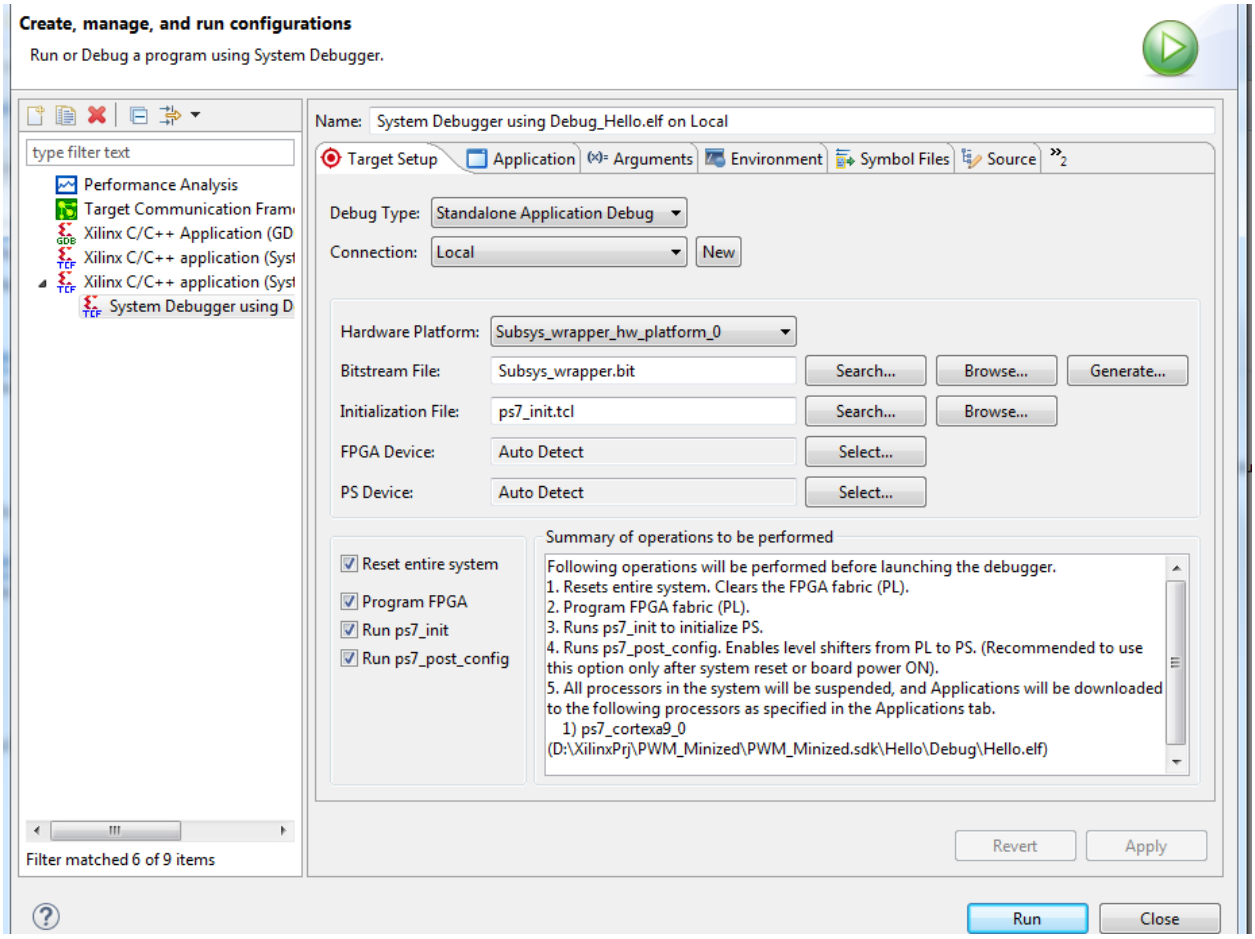

Рисунок 6 – Окно “Run Configurations”

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

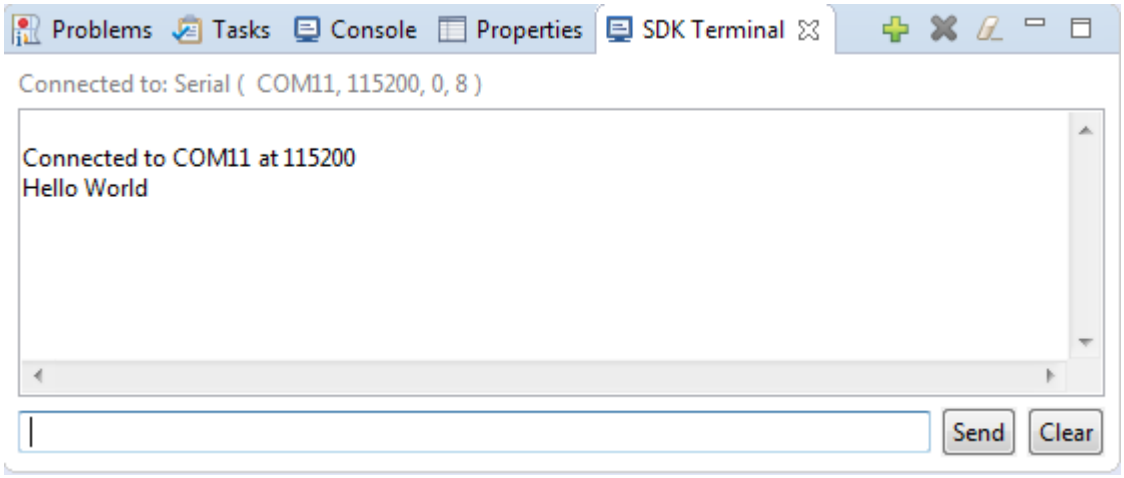

Рисунок 7 – Вывод консоли

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"
```

1. Добавим строку `#define PWM_TERMINAL 0`. 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 status;
}
```

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++){
     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 `PWM_TERMINAL`.

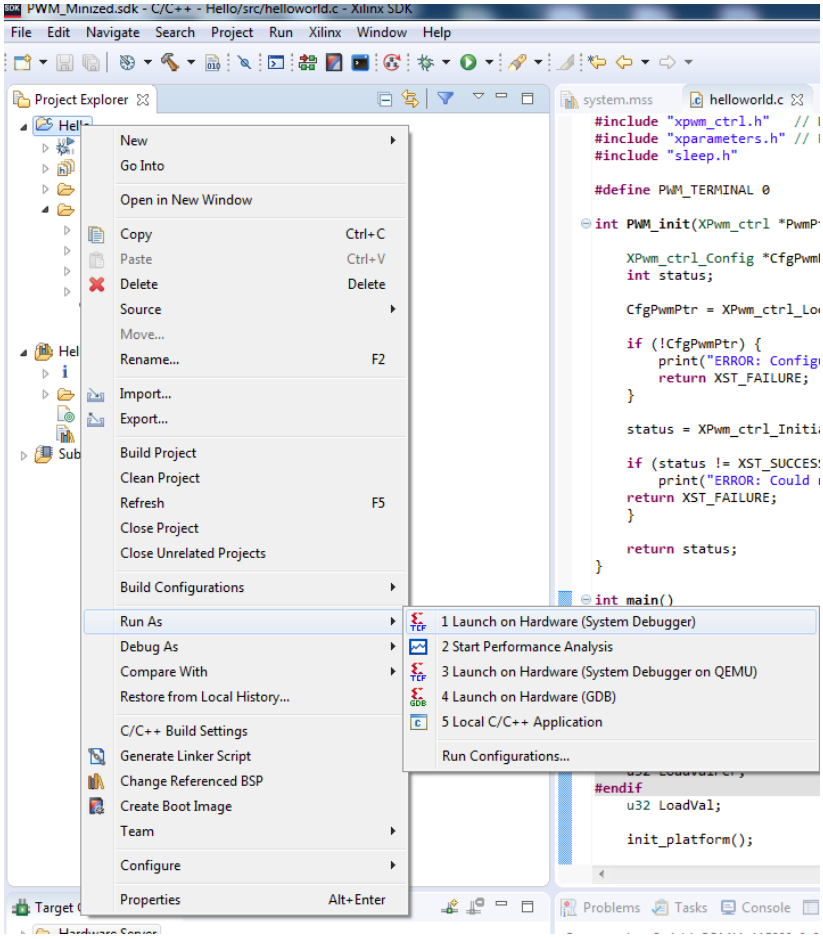

Рисунок 8 – Прошиваем СнК

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. Часть 1.
2. Разработка IP-блока с помощью инструментов высокоуровнего синтеза: HLS. Часть 2
3. MiniZed™: 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. UG973. Release Notes, Installation, and Licensing