3.3 Имплементација софтвера - Vivado SDK

IDE за развој софтвера, намењеног Zynq FPGA чиповима, је Xilinx SDK. Овај IDE базиран је на Eclipse платформи, стога је рад са Xilinx SDK идентичан раду са Eclipse-ом.

Када се SDK покрене помоћу Vivado алата, аутоматски се учита опис извезеног хардвера и направи се један пројекат `system_wrapper_hw_platform_0`. Овај пројекат садржи софтверске компоненте које иницијализују PS део FPGA чипа: све датотеке чије име почиње са `ps7_`. Додатно, овај пројекат садржи и `*.bit` датотеку са којом је могуће конфигурисати PL део чипа, помоћу SDK IDE-а.

3.3.1 Прављење апликационог пројекта

Као и у Eclipse окружењу, и у Xilinx SDK потребно је направити пројекат.

1. Направити пројекат акцијом: `File -> New -> Application Project`.

2. У добијеном дијалогу поставити `system_test` као име пројекта. За остала подешавања оставити подразумеване вредности. Даље је потребно прећи на наредна подешавања акцијом `Next`.

3. На последњем дијалогу за прављење пројекта потребно је одабрати почетни саджај. У овом примеру почетни садржај није потребан, па је стога потребно одабрати опцију `Empty Application`. Акцијом `Finish` добија се прјекат са именом `system_test` као и додати пројекат `system_test_bsp`. Све даље акције раде се у оквиру пројекта `system_test`.

4. У оквиру пројекта `system_test` потребни је направити датотеку у којој ће бити написан софтвер намењен тестирању хардверске платформе. Потребно је одабрати директоријум `src` у оквиру пројекта, па потом урадити акцију `New -> Source File` након десног клика.

5. У добијеном дијалогу потребно је подесити име главне датотеке, на пример `main.c`. Када се уради акција `Finish` добија се празна датотека.

6. Садржај датотеке попунити одговарајућим садржајем (листинг 3.11). SDK аутоматски компајлира пројекат и прави Executable and

Linkable Format (ELF) датотеку намењену извршавању на циљном процесору.

Листинг 3.11: Тест софтвер за хардверску платформу

```
#include "xparameters.h"
#include "xtmrctr.h"
#include "xil_exception.h"
#include "xscugic.h"
#include "xil_printf.h"
#include "xgpio.h"

#define TIMER_CNTR_0     0
#define LED_CHANNEL 1

#define RESET_VALUE      0xF0000000

typedef struct IRQData_t
{
        XGpio* pGpio;
        u32 val;
} IRQData;

void TimerCounterHandler(void *CallBackRef, u8 TmrCtrNumber);

int InitIPGpio(XGpio* pHandler, u16 id, unsigned channel);
int InitIPTimer(XTmrCtr* pHandler, u16 id);
int InitIntrSys(XScuGic* pHandler, u16 id);
int ConnectTimer2Gic(XScuGic* pGic, XTmrCtr* pTimer, u16 irqID);
void SetTimerOptions(XTmrCtr* pTimer, XTmrCtr_Handler irqRoutine, void* dataArg, u8 timerNumber, u32 initVal);

int main()
{
        int status;
        IRQData irqdata;
        /*
         * Interrupt controller
         */
        XScuGic Gic;
        /*
         * Timer software handler
         */
        XTmrCtr Timer;
        /*
         * GPIO software handler
         */
        XGpio Gpio;

        xil_printf("Starting...\n\r");

        status = InitIPGpio(&Gpio, XPAR_GPIO_0_DEVICE_ID, LED_CHANNEL);
        irqdata.pGpio = &Gpio;
        irqdata.val = 0;

        if (status != XST_SUCCESS)
        {
                xil_printf("GPIO failed\r\n");
                return XST_FAILURE;
        }
        xil_printf("Gpio good.\n\r");

        // interrupt controller ID is 0
        status = InitIntrSys(&Gic, 0);

        if (status != XST_SUCCESS)
        {
                xil_printf("Interrupt system failed\r\n");
                return XST_FAILURE;
        }
        xil_printf("Interrupt system good.\n\r");

        status = InitIPTimer(&Timer, XPAR_TMRCTR_0_DEVICE_ID);

        if (status != XST_SUCCESS)
        {
                xil_printf("Timer failed\r\n");
                return XST_FAILURE;
        }
        xil_printf("Timer good.\n\r");
```

```
        status = ConnectTimer2Gic(&Gic, &Timer, XPAR_FABRIC_TMRCTR_0_VEC_ID);

        if (status != XST_SUCCESS)
        {
                xil_printf("Connection failed\r\n");
                return XST_FAILURE;
        }
        xil_printf("Connection good.\n\r");

        SetTimerOptions(&Timer, TimerCounterHandler, &irqdata, 0, RESET_VALUE);

        xil_printf("Timer options good.\n\r");

        XTmrCtr_Start(&Timer, 0);

        xil_printf("Timer started.\n\r");

        while(irqdata.val != 5)
        {

        }

        XTmrCtr_Stop(&Timer, 0);
        XScuGic_Disable(&Gic, XPAR_FABRIC_TMRCTR_0_VEC_ID);
        XScuGic_Disconnect(&Gic, XPAR_FABRIC_TMRCTR_0_VEC_ID);

        xil_printf("Test finished\r\n");
        return XST_SUCCESS;
}

int InitIPGpio(XGpio* pHandler, u16 id, unsigned channel)
{
        int status = XGpio_Initialize(pHandler, id);
        if (status != XST_SUCCESS)
        {
                return XST_FAILURE;
        }
        else
        {
                // All Gpio pins are output
                XGpio_SetDataDirection(pHandler, channel, 0);
                return XST_SUCCESS;
        }
}

int InitIntrSys(XScuGic* pHandler, u16 id)
{
        int status;
        XScuGic_Config *pCfg;

        pCfg = XScuGic_LookupConfig(id);
        if (NULL == pCfg) return XST_FAILURE;

        status = XScuGic_CfgInitialize(pHandler, pCfg,
                                        pCfg->CpuBaseAddress);
        if(status != XST_SUCCESS)
                return XST_FAILURE;

        /*
         * Initialize the exception table.
         * Enable non-critical exceptions.
         */
        Xil_ExceptionInit();
        Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,
                                        (Xil_ExceptionHandler)
                                        XScuGic_InterruptHandler,
                                        pHandler);
        Xil_ExceptionEnable();
        return XST_SUCCESS;
}

int InitIPTimer(XTmrCtr* pHandler, u16 id)
{
        int status;

        status = XTmrCtr_Initialize(pHandler, id);
        if (status != XST_SUCCESS)
                return XST_FAILURE;

        status = XTmrCtr_SelfTest(pHandler, 0);
        return status;
}
```

```
int ConnectTimer2Gic(XScuGic* pGic, XTmrCtr* pTimer, u16 irqID)
{
        int status;

        /*
         * Connect the interrupt handler that will be called when an
         * interrupt occurs for the device.
         */
        status = XScuGic_Connect(pGic, irqID,
                                 (Xil_ExceptionHandler)XTmrCtr_InterruptHandler,
                                 pTimer);
        if (status != XST_SUCCESS)
                return status;

        /*
         * Enable the interrupt for the Timer device.
         */
        XScuGic_Enable(pGic, irqID);

        return XST_SUCCESS;
}

void SetTimerOptions(XTmrCtr* pTimer, XTmrCtr_Handler irqRoutine, void* dataArg, u8 timerNumber, u32 initVal)
{
        XTmrCtr_SetHandler(pTimer, irqRoutine, dataArg);
        XTmrCtr_SetOptions(pTimer, timerNumber, XTC_INT_MODE_OPTION | XTC_AUTO_RELOAD_OPTION);
        XTmrCtr_SetResetValue(pTimer, timerNumber, initVal);
}

/*
 * Interrupt routine
 */
void TimerCounterHandler(void *CallBackRef, u8 TmrCtrNumber)
{
        IRQData* pData = (IRQData *)CallBackRef;

        pData->val++;

        xil_printf("Timer interrupt happened val = %d\n\r", pData->val);

        XGpio_DiscreteWrite(pData->pGpio, LED_CHANNEL, pData->val);
}
```

3.3.2 Програмирање FPGA и покретање апликације

Све потребне бинарне датотеке су на располагању потребне за пуштање софтверског теса на циљаној хардверској платформи.

1. Потребно је програмирати FPGA чип акцијом `Xilinx Tools -> Program FPGA`. Сва подешавања требају да остану на подразумеваним вредностима. У добијеном дијалогу, акцијом `Program` програмира се FPGA чип одговарајућом `bit` датотеком.

2. Потребно је покренути апликацију на ARM процесору FPGA чипа. То се може постићи на више начина од којих је један акцијом: десни клик на пројекат па `Run As -> Launch on Hardware (GDB)`

Програм ће се покренути на процесору. LED на плочи би требао да почне да се мења, а на UART терминалу би требао да се добије следећи испис:

```
Starting...
```

```
Gpio good.
Interrupt system good.
Timer good.
Connection good.
Timer options good.
Timer started.
Timer interrupt happened val = 1
Timer interrupt happened val = 2
Timer interrupt happened val = 3
Timer interrupt happened val = 4
Timer interrupt happened val = 5
Test finished
```