Lista 4: Writing Basic Software Applications

Przemysław Dragańczuk, Marcin Serafin, Aksel Stankiewicz

33INF–SSI/A

1 Wstęp

1.1 Cel ćwiczenia

Celem ćwiczenia jest pokazanie, jak można napisać prostą aplikację softwareową dla płytki Spartan-3E. Aplikacja ta ma za zadanie włączanie odpowiednich diod LED w zależności od stanu przycisków i przełączników. Sekcja .text oprogramowania będzie przechowywana w pamięci BRAM.

2 Przebieg ćwiczenia

2.1 Modyfikacja projektu z laboratorium 3

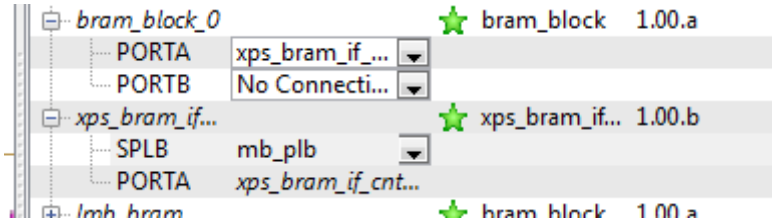
Skopiowano folder lab3 do nowego folderu lab4, a następnie otwarto go w programie XPS.

2.2 Dodanie BRAM i kontroler BRAM

Z katalogu IP dodano następujące IP do projektu systemu:

- XPS BRAM Controller 1.00.b
- Block RAM (BRAM) block 1.00.a

Następnie połączono kontroler BRAM do PLB i BRAM do kontrolera BRAM.



Rysunek 1: Dodane i połączone IP BRAM i BRAM controller

W zakładce ”Adresses” ustawiono rozmiar pamięci BRAM na 8K i kliknięto na przycisk ”Generate Adresses”

W menu ”Hardware” kliknięto na ”Generate Bitstream”. Później w menu ”Project” kliknięto ”Export Hardware Design to SDK” a następnie ”Export & Launch SDK”.

2.3 Tworzenie aplikacji

Po otworzeniu SDK w odpowiednim folderze, utworzono nowy projekt typu ”Xilinx C Project”. Wybrano szablon ”Empty Application” a jako nazwę podano ”TestApp”. Następnie wybrano odpowiedni BSP. Zaimportowano plik lab4.c.

```
#include "xparameters.h"
#include "xgpio.h"
#include "xutil.h"

//----------------------------------------------------

int main (void)
{

   XGpio dip, push, led;
    int i, psb_check, dip_check;
    // define instance pointer for LEDs_8Bit device


   xil_printf("-- Start of the Program --\r\n");

   XGpio_Initialize(&dip, XPAR_DIP_DEVICE_ID);
    XGpio_SetDataDirection(&dip, 1, 0xffffffff);

    XGpio_Initialize(&push, XPAR_PUSH_DEVICE_ID);
    XGpio_SetDataDirection(&push, 1, 0xffffffff);
    // initialize and set data direction for LEDs_8Bit device
    XGpio_Initialize(&led, XPAR_LEDS_8BIT_DEVICE_ID);
    XGpio_SetDataDirection(&led, 1, 0xffffffff);

    while (1)
    {
      psb_check = XGpio_DiscreteRead(&push, 1);
      xil_printf("Push Buttons Status %x\r\n", psb_check);
      dip_check = XGpio_DiscreteRead(&dip, 1);
      xil_printf("DIP Switch Status %x\r\n", dip_check);

      // output dip switches value on LEDs_8Bit device
      XGpio_DiscreteWrite(&led, 1, psb_check);

      for (i=0; i<999999; i++);
    }
}
```

Rysunek 2: Zaimporowany program

Po przeczytaniu dokumentacji zmodyfikowano kod tak, aby na diodach LED wyświetlany był stan przycisków i przełączników.

```
#include "xparameters.h"
#include "xgpio.h"
#include "xutil.h"

int main (void)
{
    XGpio dip, push, led;
    int i, psb_check, dip_check;
```

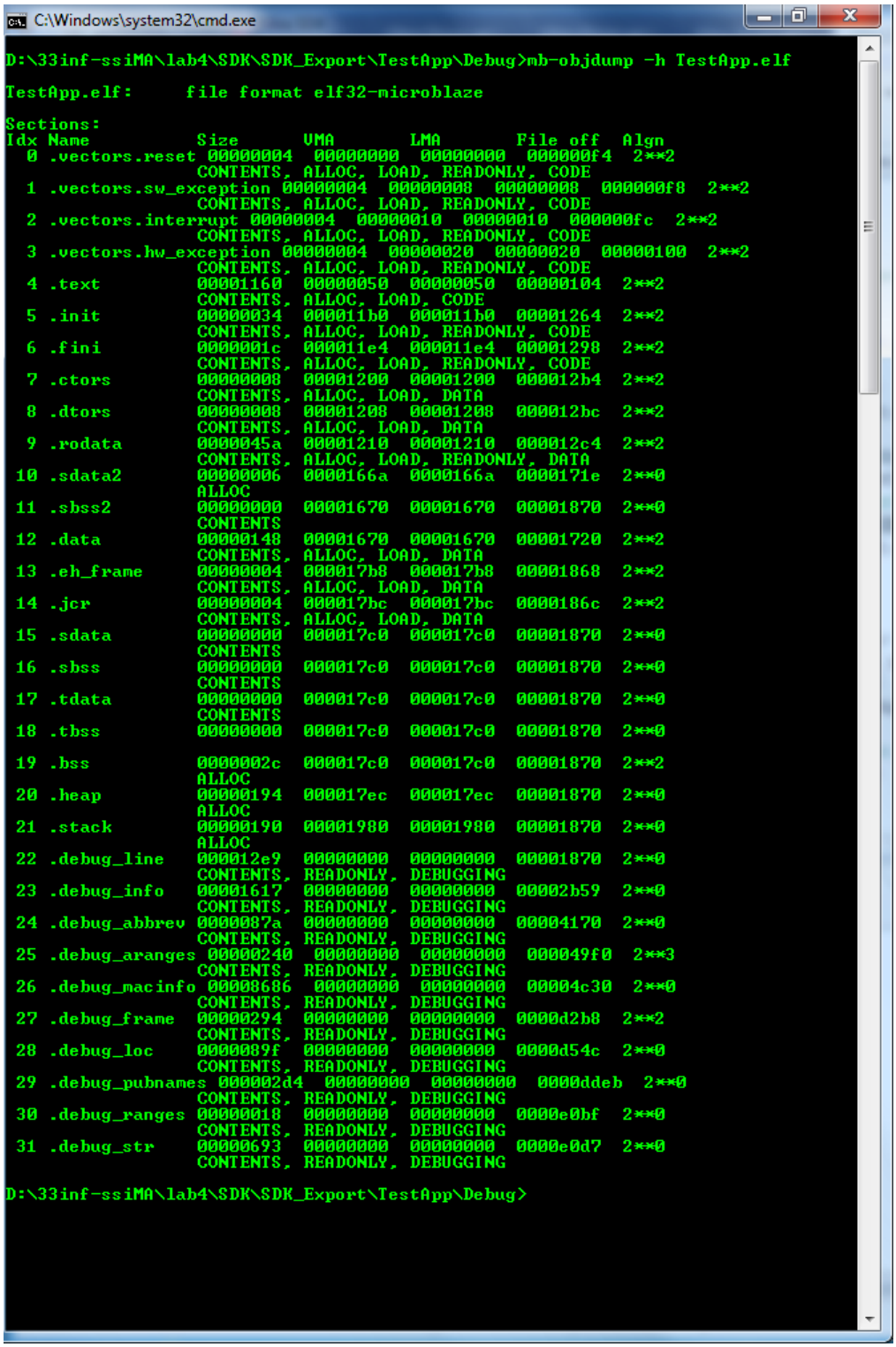
```
9
10      xil_printf("-- Start of the Program --\r\n");
11
12      XGpio_Initialize(&dip, XPAR_DIP_DEVICE_ID);
13      XGpio_SetDataDirection(&dip, 1, 0xffffffff);
14
15      XGpio_Initialize(&push, XPAR_PUSH_DEVICE_ID);
16      XGpio_SetDataDirection(&push, 1, 0xffffffff);
17
18      XGpio_Initialize(&led, XPAR_LEDS_8BIT_DEVICE_ID);
19      XGpio_SetDataDirection(&led, 1, 0xffffffff);
20
21      while (1)
22      {
23          psb_check = XGpio_DiscreteRead(&push, 1);
24          xil_printf("Push Buttons Status %x\r\n", psb_check);
25          dip_check = XGpio_DiscreteRead(&dip, 1);
26          xil_printf("DIP Switch Status %x\r\n", dip_check);
27
28          int status = (psb_check << 4) | dip_check;
29
30          XGpio_DiscreteWrite(&led, 1, status);
31
32          for (i=0; i<999999; i++);
33      }
34  }
```

2.4 Analiza zbudowanych obiektów

W menu ”Xilinx Tools” kliknięto na ”Generate Linker Script”. Następnie zmieniono rozmiar stosu i stogu na 400 bajtów i kliknięto ”Generate”.

Zrekompilowano plik lab4.c, a następnie w menu ”Xilinx Tools” wybrano ”Launch shell”. Komenda `cd` została użyta aby przejść do folderu `TestApp/Debug`.

Użyto komendy `mb-objdump -h TestApp.elf` do wyświetlenia poszczególnych sekcji programu.



Ponownie otwarto okno ”Generate Linker Script”. Rozmiary stosu i stogu zostały zmienione na 1KB. Następnie w liście rozwijanej ”Place Code Sections in: ” wybrano kontroler BRAM.

Po ponownej rekompilacji pliku lab4.c i uruchomieniu komendy `mb-objdump` zauważono, że sekcja .text przesunęła się do pamięci BRAM.

2.5 Weryfikacja w sprzęcie

Używając narzędzia "Program FPGA" wgrano aplikację na płytkę i zweryfikowano, że działa

3 Wnioski

Oprogramowanie Xilinx Platform Studio pozwala w bardzo prosty sposób projektować oprogramowanie, które można wgrać na płytki FPGA. Pozwala również na zobaczenie wszystkich informacji o zainstalowanym sprzęcie, jak połączenia między urządzeniami czy przypisane adresy pamięci.