Programowanie procesora Microblaze w środowisku SDK

Zespół Rekonfigurowalnych Systemów Obliczeniowych
AGH Kraków
http://www.fpga.agh.edu.pl/

9 kwietnia 2010

1.Wstęp

Celem niniejszego ćwiczenia jest:

- zapoznanie z narzędziami umomożliwościami debuggowania kodu uruchomionego na platformie MicroBlaze
- zapoznanie ze środowiskiem graficznym wspomagającym prace programisty: Xilinx Platform SDK opartym na narzędziu Eclipse.

Wymagania:

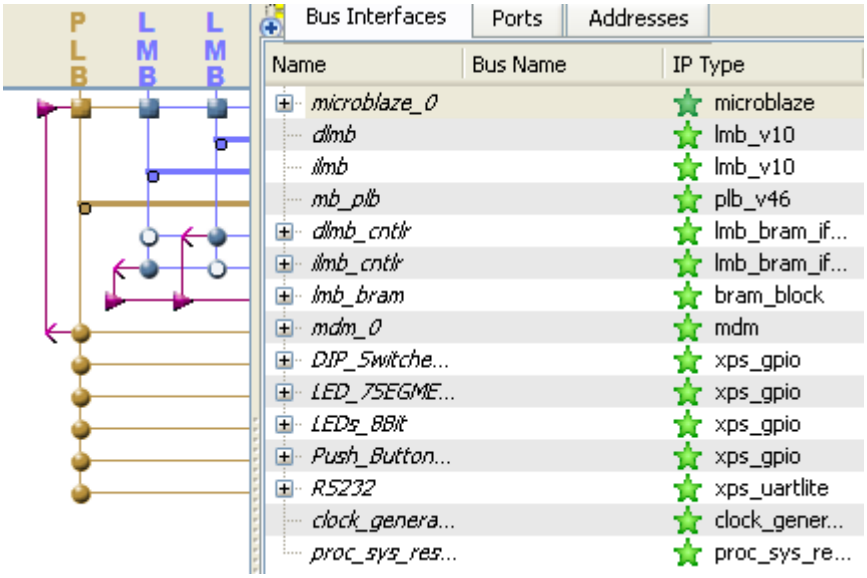
- Xilinx ISE Design Suite 11.2.
- Płytka uruchomieniowa Spartan3E-200 - Digilent Board.

2.Zapoznanie z platformą sprzętowa

Do wykonania poniższego ćwiczenia należy posłużyć się platformą sprzętową Microblaze wykonaną w poprzednich ćwiczenia:
„EDK MicroBlaze Tutorial Documentation z wykorzystaniem platformy FPGA z rodziny SPARTAN3E”

a) Komponenty sprzętowe

W skład platformy uruchomieniowej powinny wchodzić następujące komponenty



Rysunek 1 Moduły sprzętowe

- **microblaze_0** – procesor MicroBlaze
- **mb_plb** – magistrala i arbiter OPB
- **mdm0** – moduł interfejsu umożliwiający sprzętowe debuggowanie poprzez JTAG
- **RS232** – interfejs UART
- **dcm_0** – moduł generatrora zegara
- **dlmb_cntlr** – interfejs danych BRAM
- **ilmb_cntlr** – interfejs instrukcji BRAM
- **lmb_bram** – pamięć Block RAM
- peryferia GPIO

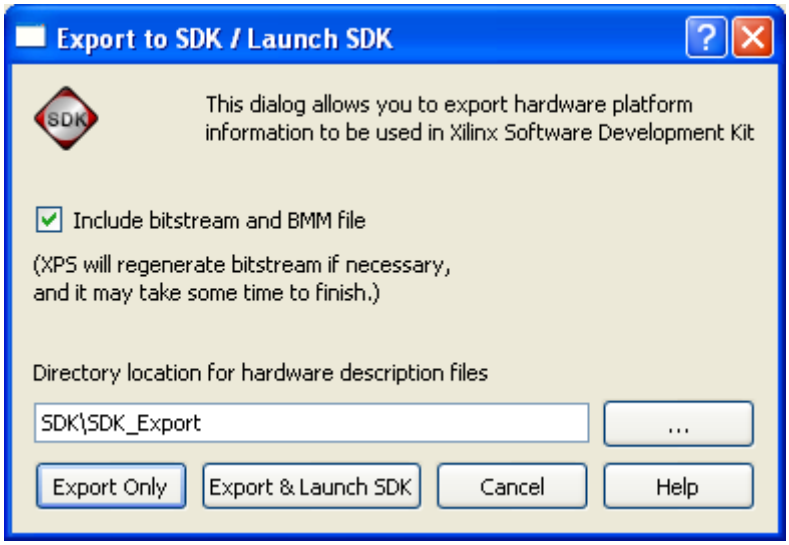
b) Eksport ustawień do środowiska SDK

Aby możliwe było korzystanie ze środowiska programistycznego GUI należy dokonać eksportu parametrów i ustawień utworzonej w Xilinx Platform Studio platformy sprzętowej. Ustawienia te są eksportowane w formacie XML

W celu wygenerowania pliku z ustawieniami należy w środowisku XPS w którym otwarty jest nasz projekt hardwarowy wykonać:

Projekt → Export Hardware Design To SDK

Pojawi się okno



Wybrać polecenie „Export Only”

4. Środowisko Eclipse

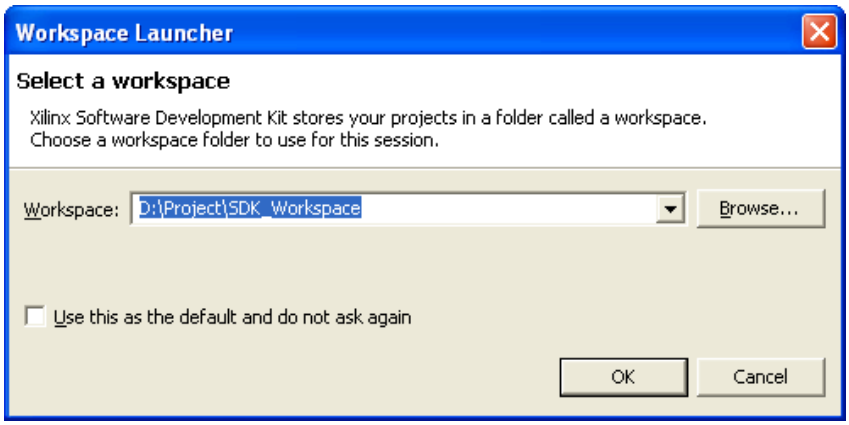
Eclipse jest rozbudowanym środowiskiem programistycznym napisanym w Javie. Projekt został stworzony przez firmę IBM a następnie udostępniony społeczności Open Source.

W naszym przypadku jest ono dostosowane przez firmę Xilinx m.in. w celu łatwiejszego tworzenia i debuggowania oprogramowania na platformę MicroBlaze.

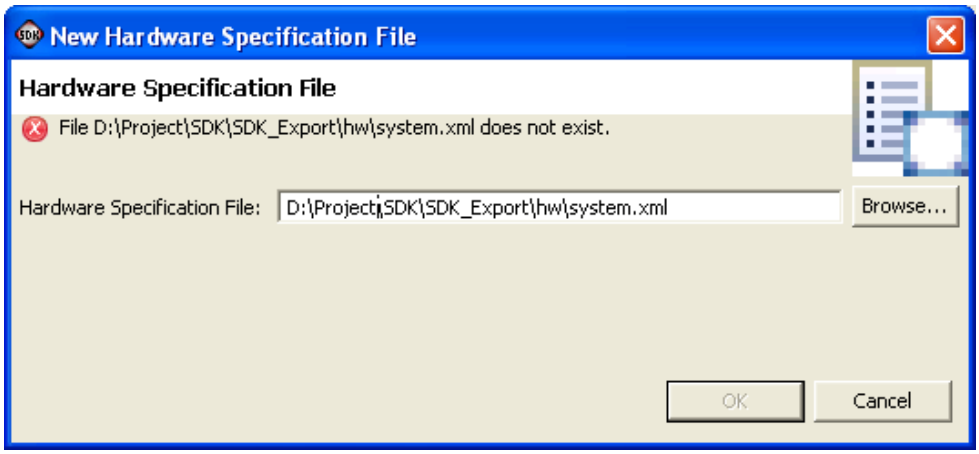
a) Tworzenie projektu

Otworzyć z menu START Windows aplikacje Xilinx Platform SDK.

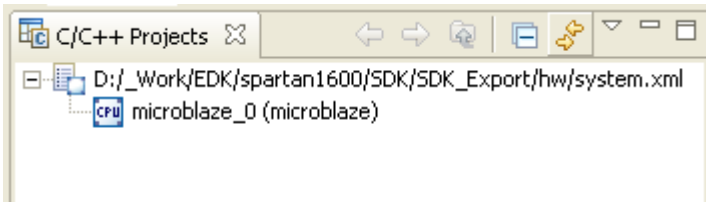
Na początku należy wskazać miejsce na dysku gdzie będzie zapisywany projekt. Można wybrać dowolne miejsce, ale w podanej ścieżce **nie może być spacji.**



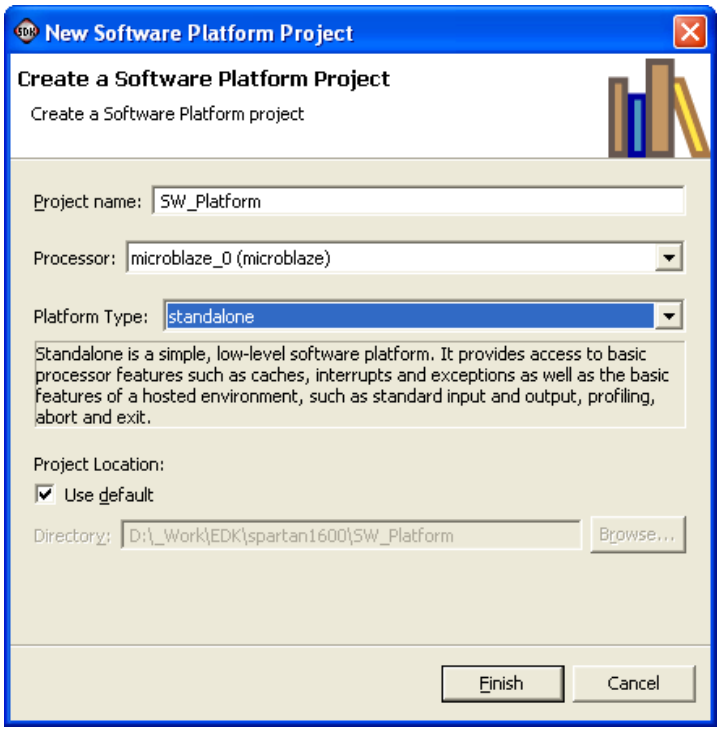
Również podczas otwierania aplikacja zapyta o plik XML z ustawieniami platformy sprzętowej. Należy podąć ścieżkę do wygenerowanego wcześniej pliku XML:



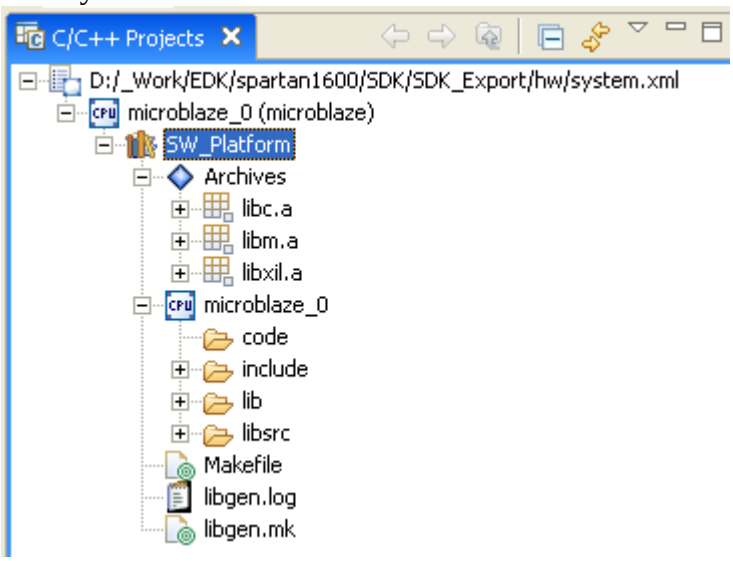
Wybrana platforma sprzętowa będzie widoczna w oknie nawigacji projektu:



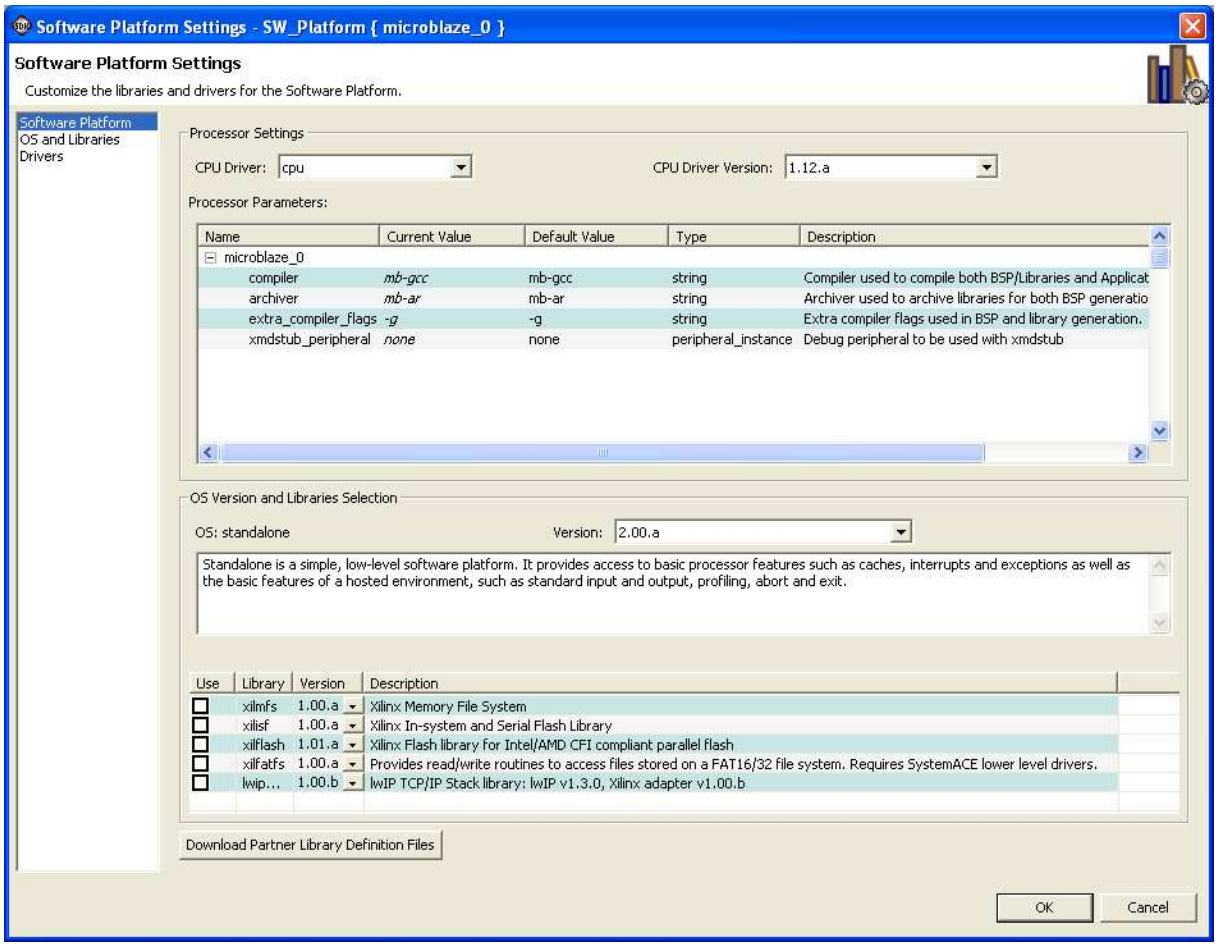
Dla wyeksportowanej z XPS platformy sprzętowej musimy w SDK utworzyć odpowiadającą platformę softwarową:
Wybieramy w SDK **File→ New → Software Platform**



Należy wybrać nazwę platformy softwarowej i typ platformy „*standalone*”. Wciskamy „Finish”
Powstaje platforma sprzętowa ze wszystkimi bibliotekami i driverami potrzebnymi do programowania platformy.

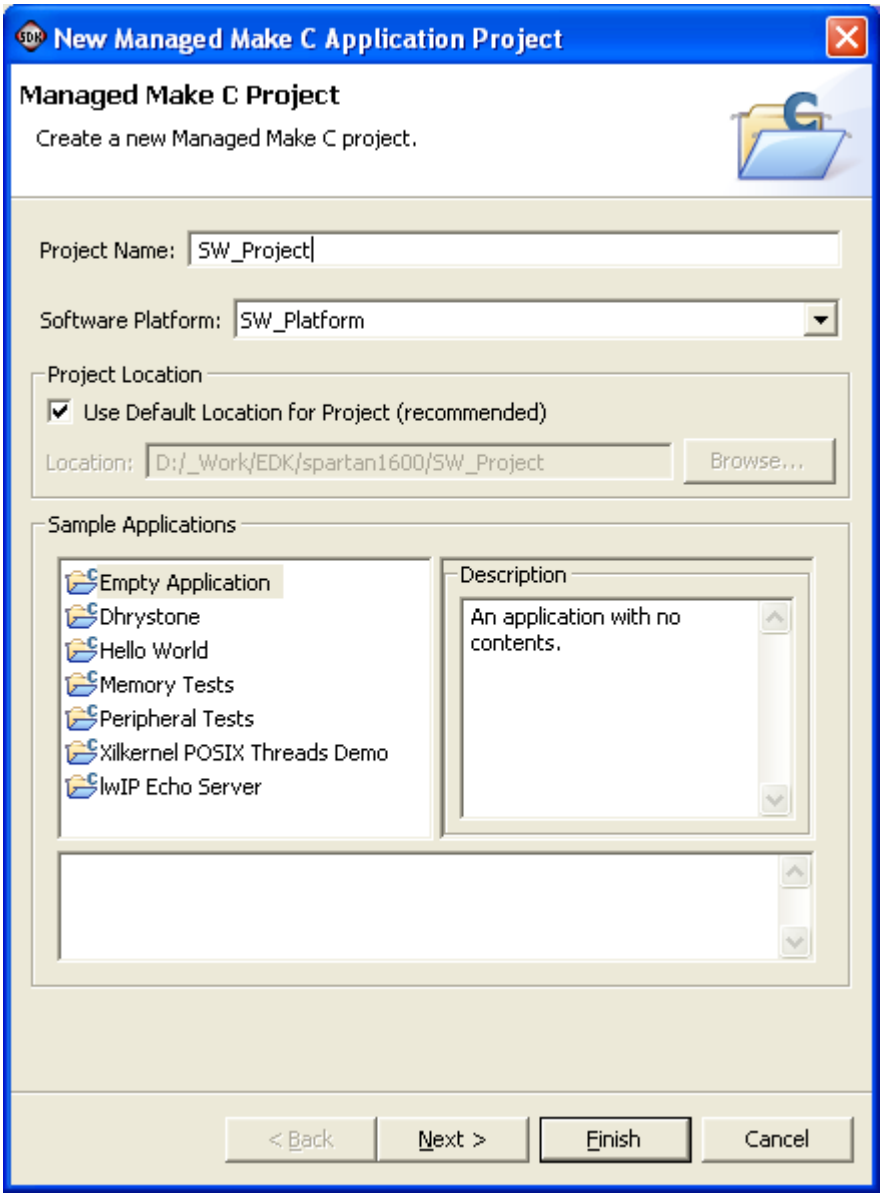


Listę dołączonych bibliotek i sterowników można zobaczyć wybierając „Software Platform Settings” w menu kontekstowym wpisu z nazwą platformy (prawy klawisz myszy na nazwie platformy) .



b) Pisanie i kompilacja kodu

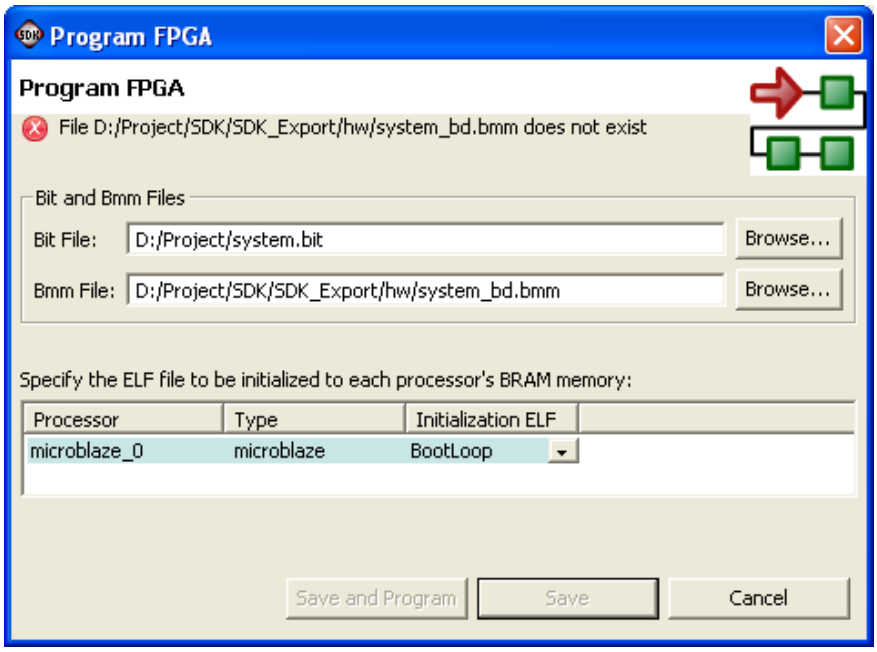
W następnym kroku tworzymy projekt aplikacji:
File→ New → Managed Make C Project



Wybieramy nazwę naszego projektu i klikamy **Finish**.

c) Programowanie układu FPGA

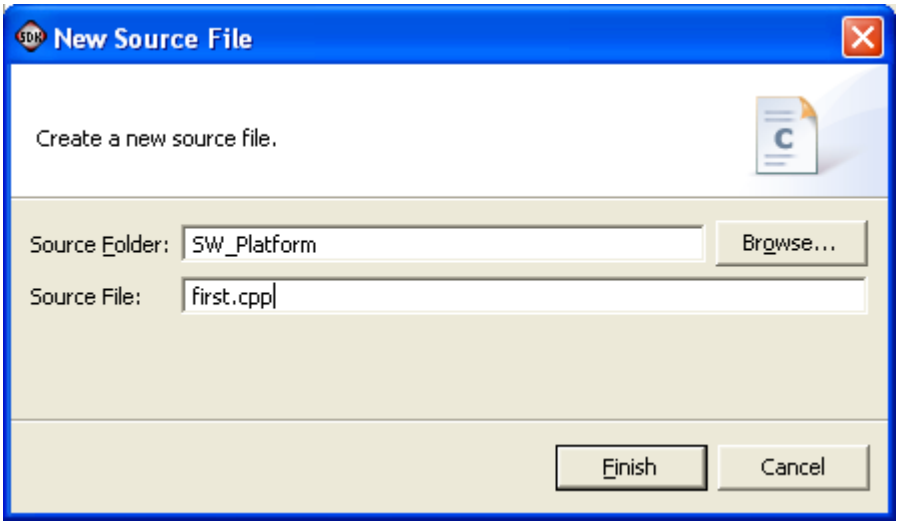
Aby możliwe było uruchamianie kodu na platformie FPGA musi ona zostać wstępnie zaprogramowana konfiguracją sprzętową zdolna do komunikacji z PC poprzez JTAG
Wykonujemy komendę:
File→ Program FPGA



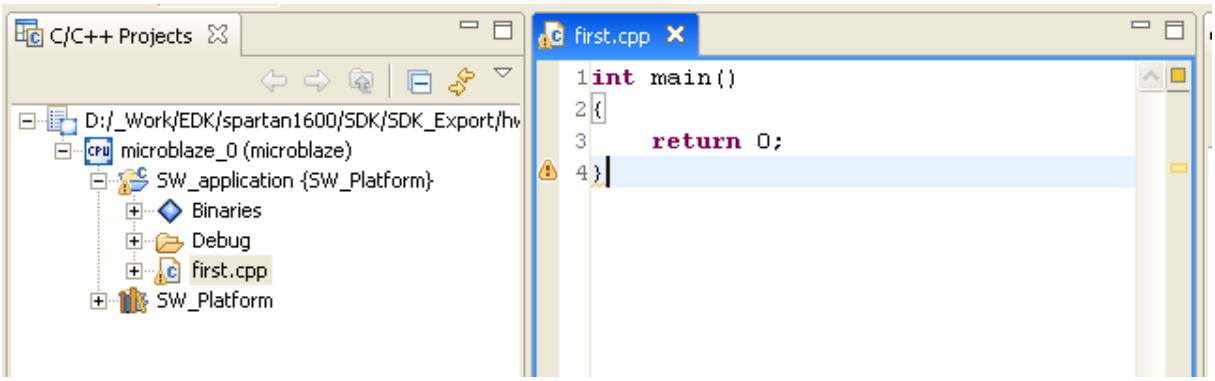
Automatycznie wybrany jest plik konfiguracyjny naszej platformy hardwarowej i plik konfiguracji pamięci dla niej. W pliku **.bmm** zawarte są wszystkie informacje potrzebne do prawidłowego umieszczenia skompilowanego kodu w pamięciach Block RAM FPGA.
We wgranej konfiguracji procesor wykonywać będzie program "branch-to-itself": bootloop. Można ten program zmienić wybierając inny skompilowany kod czyli inny plik **.elf**. Przy pierwszym uruchomieniu nie zmieniamy nic. Naciskamy „Save and Program"

d) Pisanie i kompilacja kodu

Do utworzonego projektu dodajemy nowy plik źródłowy
File→ New → Source File



Plik pojawi się w oknie projektu



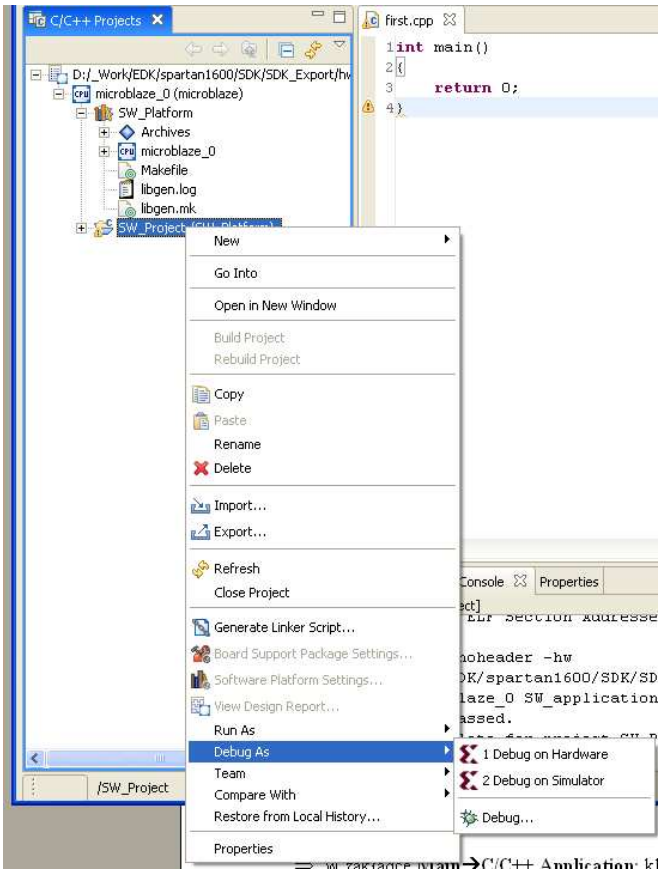
Teraz możemy przystąpić do pisania naszej aplikacji.
W oknie edytora wpisujemy poniższy kod i zapisujemy projekt.

```
1  #include "xgpio.h"
2  #include "xparameters.h"
3
4  main()
5  {
6     Xuint8 a,b;
7
8     while(1)
9     {
10       a=-1;b=-1;
11
12       while(a>9){
13          xil_printf("\nPodaj a=");
14          a=XUartLite_RecvByte(XPAR_RS232_BASEADDR);
15          a-=0x30;
16       }
17
18       while(b>9){
19          xil_printf("\nPodaj b=");
20          b=XUartLite_RecvByte(XPAR_RS232_BASEADDR);
21          b-=0x30;
22       }
23
24
25       xil_printf("\nSuma=%d", a+b);
26    }
27
28    return 0;
29 }
```

Projekt zostanie przekompilowany a w oknie konsoli pojawi się komunikat:
„`Build complete for project SW_Project`”

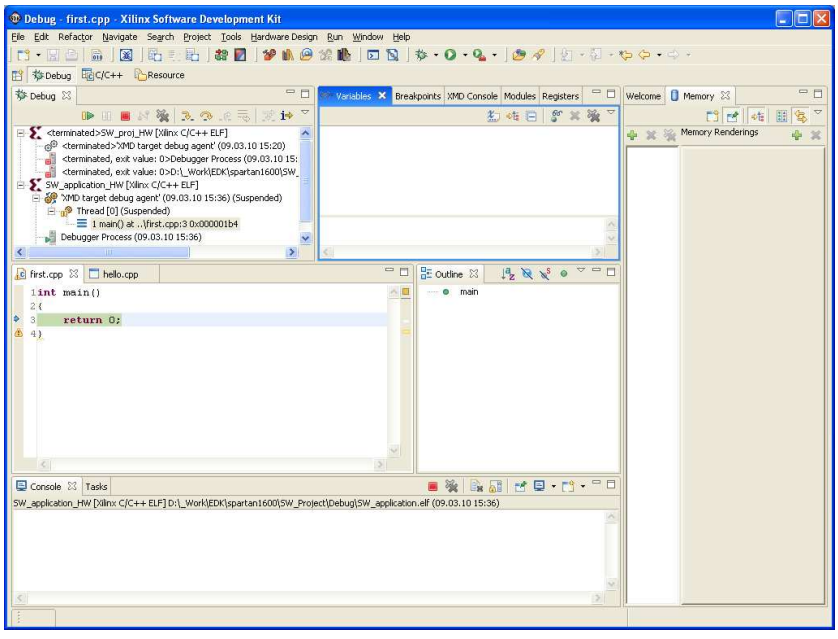
e) Uruchomienie debuggera

Z menu kontekstowego projektu wybieramy
DebugAs→ Debug on Hardware



f) Wgrywanie i debuggowanie programu

Przed uruchomieniem programu płytkę Spartan3E200 należy połączyć z terminalem np.: Hyperterminal Windows.
Aplikacja automatycznie załaduje nasz program do pamięci (plik *.elf) zaczynając od lokacji 0x00000000. Następnie przejdzie do okna debuggowania.



Debugger środowiska SDK umożliwia wszystkie czynności, jakie dostępne są w innych środowiskach programistycznych np. Visual C++. Możliwa jest praca krokowa., ustawianie pułapek (**breakpoint**), podgląd zmiennych, pamięci rejestrów procesora i wiele innych.

Nie należy zapominać że program wykonuje się w rzeczywistym procesorze na naszej platformie sprzętowej, a środowisko SDK komunikuje się z procesorem przy interfejsu JTAG

Program należy przekrokować obserwując zachowanie płytki.

5. Zaliczenie

Napisać i uruchomić program spełniający następujące funkcje:

- Wyświetlający na diodach led wartość wpisana z terminala
- Wypisujący na terminalu wartość ustawioną na przełącznikach DIP
- Reagujący zdefiniowanym komunikatem po wciśnięciu przycisku „push button” np.: „Wciśnięto przycisk A”.

Wszystkie wymienione funkcje mogą być realizowane równocześnie – w jednej pętli programu.