AUTOMATYKA • 2009 • Tom 13 • Zeszyt 3

Aleksander Jasiński, Marek Gorgoń*

Uruchomienie i badanie właściwości systemu Petalinux w środowisku układu reprogramowalnego FPGA

1. Wprowadzenie

Poszerzanie zakresu zastosowań systemów wbudowanych sprawia, że konieczne staje się poszukiwanie metod zwiększania ich wydajności oraz stopnia zintegrowania. Efektem wzrostu ilości przetwarzanych danych jest niewystarczająca efektywność klasycznych rozwiązań o architekturze wykorzystującej mikroprocesory ogólnego zastosowania. W obliczu tendencji do miniaturyzacji i wzrostu energooszczędności sprzętu elektronicznego rozmiar oraz zapotrzebowanie na energię tego typu systemów bywa również nie do zaakceptowania. Obecnie alternatywą dla układów mikroprocesorowych są rozwiązania bazujące na technologii ASIC. Wysoka efektywność osiągana dzięki przetwarzaniu sprzętowemu w tego typu układach często nie jest wystarczająco silnym argumentem w zestawieniu z brakiem elastyczności oraz czasem i kosztami ich wytworzenia. Jak widać, ewolucja systemów wbudowanych doprowadziła do punktu, w którym koniecznym staje się opracowanie układów wykorzystujących nową technologię umożliwiającą ich dalszy rozwój [5, 6].

Jednym z możliwych rozwiązań opisanego problemu jest zastosowanie systemów wbudowanych z architekturą sprzętową generowaną w układzie FPGA (*Field Programmable Gate Array*). Rozwiązania tego typu charakteryzują się wysokim stopniem integracji, tak istotnym w odniesieniu do zastosowań wbudowanych. Wynika to z faktu, iż znacząca część architektury systemu znajduje się w obrębie układu reprogramowalnego, ograniczając tym samym jego rozmiar, pobór mocy oraz czas związany z komunikacją. Duża elastyczność gwarantująca szeroki zakres zastosowań systemu osiągana jest dzięki wykorzystaniu softprocesora o funkcjonalności porównywalnej z procesorami ogólnego zastosowania. Efektywność rozwiązania determinowana jest poprzez wykorzystanie do zadań o wysokiej złożoności obliczeniowej algorytmów przetwarzania sprzętowego implementowanych w układzie FPGA [2, 3]. Wydaje się zatem, że zaproponowana architektura systemu wbudowanego pozwala na istotne usprawnienie jego działania oraz rozwiązanie problemów opisanych w pierwszym akapicie [4].

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

W pracy opisano działania mające na celu stworzenie systemu komputerowego wykorzystującego softprocesor Microblaze [9, 11] oraz system operacyjny Petalinux. Prace stanowią powtórzenie eksperymentów prowadzonych przez zespół profesora Johna Williamsa na Uniwersytecie Queensland [8]. Celem pracy było samodzielne uruchomienie oraz zbadanie możliwości systemu pod kątem wykorzystania w zastosowaniach wbudowanych. Dzięki stworzonym aplikacjom, zaproponowane rozwiązanie wybiega poza funkcjonalność systemu bazowego. Podczas realizacji pracy wykorzystano układ XC3S500E zainstalowany na karcie Spartan-3E Starter Kit [10, 12] oraz system operacyjny Petalinux w wersji v0.30-rc1.

2. Układ FPGA jako baza systemu komputerowego

Dynamiczny rozwój technologii w branży elektronicznej sprawił, że układy FPGA przestały być postrzegane jedynie jako narzędzia do prototypowania architektur sprzętowych implementowanych następnie w strukturach ASIC. Obecnie, gdy pojemności układów FPGA sięgają milionów bramek logicznych, systemy komputerowe wykorzystujące je stają się poważną alternatywą na rynku superkomputerów i systemów wbudowanych [1].

Możliwość zaimplementowania warstwy sprzętowej i programowej w obrębie jednego urządzenia gwarantuje krótki czas ich wzajemnej komunikacji oraz wysoką energooszczędność [7]. Równie istotnym aspektem jest skalowalność rozwiązań wykorzystujących układy reprogramowalne. Ulegające systematycznemu skróceniu *time to market* oraz czas życia produktu sprawiają, że możliwość zmiany funkcjonalności systemu bez konieczności zastępowania jego jednostki sterującej może decydować o sukcesie komercyjnym na rynku.

Ważnym etapem rozwoju układów wbudowanych jest zastosowanie do ich kontroli systemów operacyjnych. Podejście to sprawia, że zmianie ulega postrzeganie tego typu urządzeń jako systemów czysto kontrolnych. Wykorzystanie systemu operacyjnego jest niewątpliwie krokiem w stronę tworzenia „rozwiązań inteligentnych", co w przyszłości prowadzić powinno do dalszej ekspansji systemów wbudowanych na rynku informatycznym.

Cechy układów wbudowanych bazujących na architekturach reprogramowalnych i systemach operacyjnych stały się motywacją do przeprowadzenia prac nad stworzeniem tego typu rozwiązania. Za bazę systemu posłużyły układ XC3S500E oraz system operacyjny Petalinux, używane z powodzeniem przez zespół badawczy z Uniwersytetu Queensland. Ze względu na stopień złożoności i skale trudności napotkanych przy realizacji projektu, wykorzystanie sprawdzonego rozwiązania okazało się właściwą decyzją.

2.1. Warstwa sprzętowa systemu

Zastosowanie systemu operacyjnego w tworzonym rozwiązaniu wymaga wygenerowania architektury sprzętowej umożliwiającej jego instalację. Minimalna funkcjonalność konieczna do poprawnego działania systemu Petalinux obejmuje:

- procesor,
- timer,

– kontroler przerwań,
– urządzenia wejścia/wyjścia,
– kontroler pamięci.

Bez wymienionych powyżej funkcji nie jest możliwe poprawne działanie systemu, dlatego z punktu widzenia projektu mają one kluczowe znaczenie [8].

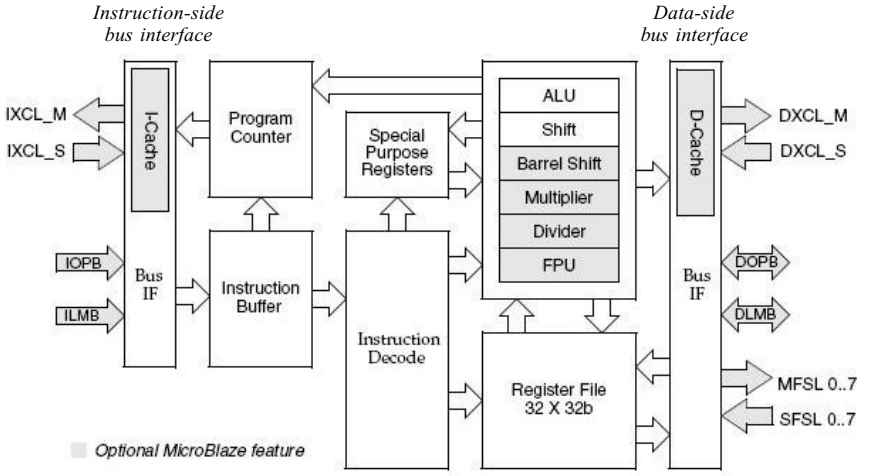
Jednostką centralną tworzonego systemu jest softprocesor Microblaze w wersji 6.0. Procesor ma architekturę typu RISC (*Reduced Instruction Set Computer*) i jest zoptymalizowany do pracy w układach reprogramowalnych. Tabela 1 przedstawia główne wielkości charakteryzujące procesor Microblaze wygenerowany w układzie Spartan 3E [9].

Tabela 1
Charakterystyka procesora Microblaze w układzie Spartan 3E

Rozmiar	Dhrystone 2.1*	Maksymalna częstotliwość zegara	Wydajność**
1843 LUTs	115 DMIPS	100 MHz	1.15 DMIPS/MHz

* Dhrystone – wynik badania przeprowadzonego za pomocą testu wydajności
** Wydajność – stosunek wyniku testu Dhrystone i maksymalnej częstotliwości zegara

Do głównych zalet Microblaze'a zaliczyć należy możliwość szerokiej konfiguracji zapewniającej optymalne wykorzystanie zasobów logicznych układu oraz wsparcie dla integracji z jego peryferiami [9]. Cechy te są szczególnie istotne w kontekście dość ograniczonych zasobów wykorzystanego układu FPGA. Na rysunku 1 przedstawiona jest architektura procesora Microblaze.



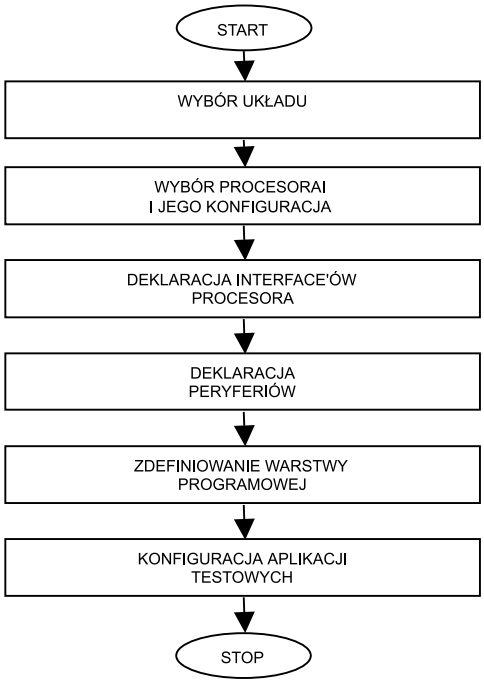
Rys. 1. Architektura procesora Microblaze [9]

W ramach prac został wygenerowany procesor Microblaze taktowany zegarem o częstotliwości 50 MHz, jest to maksymalna częstotliwość możliwa do osiągnięcia w wykorzystywanym układzie. Peryferia układu podłączone zostały do procesora za pomocą trzech magistral systemowych, taktowanych zegarem o takiej samej częstotliwości. W celu podwyższenia efektywności pracy do projektu dodano pamięci tymczasowe *cache* dla obszaru instrukcji oraz danych. Funkcjonalność ta realizowana jest za pomocą pamięci BRAM (*Block RAM*) i szybkich magistral LMB (*Local Memory Bus*). Procesor Microblaze wraz z pamięcią BRAM oraz magistralami LMB stanowią centralną część systemu.

Szkielet architektury systemu stanowi magistrala OPB (*On-Chip Peripheral Bus*) zapewniająca komunikację procesora z peryferiami. Poprzez linię danych magistrali OPB do procesora podłączone są następujące peryferia:

– kontroler Ethernet,
– kontrolery pamięci Flash i pamięci RAM,
– kontroler portu szeregowego,
– kontrolery diod, przełączniki i przyciski na karcie,
– kontroler przerwań.

Uruchomienie systemu Petalinux w układzie reprogramowalnym wymaga ponadto dołączenia do projektu warstwy sprzętowej modułu debugującego, modułu resetującego oraz magistral łączących je z procesorem. Na rysunku 2 przedstawiony został schemat etapów tworzenia warstwy sprzętowej systemu.



Rys. 2. Etapy tworzenia architektury sprzętowej systemu

Opisana architektura określa minimalną funkcjonalność niezbędną do instalacji systemu Petalinux w środowisku rekonfigurowalnym [8]. Należy zwrócić uwagę na fakt, że system w opisanym kształcie może zostać wygenerowany niemal w całości przez narzędzia dostarczane przez firmę Xilinx. Do niestandardowych czynności zaliczyć trzeba konfigurację opisanych w poprzednim akapicie modułów, oraz ewentualną rozbudowę o kontrolery peryferii. W ramach prowadzonych prac zaimplementowany został sterownik portu USB, jednak z uwagi na brak wolnych zasobów logicznych nie znalazł się w finalnej wersji projektu. Fakt ten może świadczyć o roli, jaką odgrywa optymalizacja rozmiaru tworzonej architektury. Możliwość decydowania o funkcjonalności oraz o rozmiarze procesora w przypadku wykorzystania układu FPGA o tak ograniczonej pojemności okazują się kluczowe.

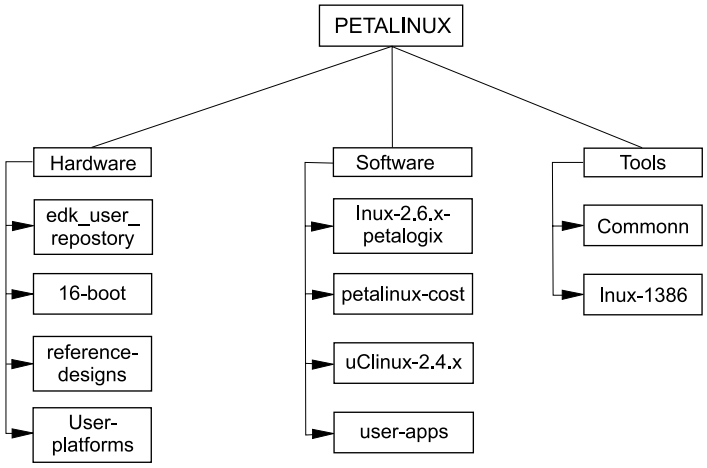
Pomimo opisanych ograniczeń, układ XC3S500E firmy Xilinx wraz ze środowiskiem programistycznym Xilinx EDK (*Embedded Development Kit*) stanowią doskonałą bazę do badań nad stworzeniem w pełni funkcjonalnych systemów wbudowanych, wykorzystujących układy FPGA.

2.2. Warstwa programowa systemu

Za bazę do stworzenia warstwy programowej posłużył system operacyjny Petalinux. Jest to oprogramowanie typu otwartego bazujące na systemie μClinux dedykowanym do zastosowań wbudowanych, a w szczególności do układów pozbawionych jednostek MMU (*Memory Management Unit*). Petalinux wykorzystuje jądra linuxowe serii 2.0, 2.4 i 2.6. Ograniczenie wymagań sprzętowych systemu osiągnięte zostało dzięki zredukowaniu funkcjonalności w stosunku do wersji oryginalnych. Polegało ono na zmniejszeniu ilości dostępnych funkcji oraz zastąpieniu standardowych bibliotek przez ich „lżejsze" wersje. Komendy systemu Petalinux pochodzą z pakietu BusyBox, przeznaczonego do zastosowań w systemach o ograniczonej ilości pamięci operacyjnej oraz szybkości procesora. Podczas kompilacji jądra Petalinuxa istnieje możliwość wyboru jedynie potrzebnych funkcji. Efektem jest ograniczony rozmiar obrazu systemu ładowany do pamięci Flash układu.

Do systemu dołączony jest kompilator języka C oraz biblioteka μClibc zawierająca wiele funkcji języka C zoptymalizowanych do zastosowania w systemach wbudowanych. Pomimo niewielkiego rozmiaru biblioteki μClibc większość aplikacji obsługiwanych przez glibC działa poprawnie z wykorzystaniem funkcji pochodzących z μClibc. Migracja projektu ogranicza się najczęściej jedynie do przekompilowania kodu źródłowego.

Istnieje możliwość poszerzenia funkcjonalności tworzonego systemu o własne aplikacje. Ponieważ obraz systemu przechowywany jest w pamięci trwałej układu, do której dostęp możliwy jest jedynie przez programy rozruchowe, definicja aplikacji użytkownika musi nastąpić już na poziomie kompilacji jądra systemu. Sam proces dodawania nowych funkcjonalności jest w pełni zautomatyzowany przez skrypty dostarczone wraz z oprogramowaniem. Po jego zakończeniu nowe aplikacje są widoczne z poziomu użytkownika jak wbudowane komendy systemu [13, 14]. Rysunek 3 przedstawia drzewo katalogów systemu Petalinux.



Rys. 3. Schemat drzewa katalogów systemu Petalinux

Reasumując, można stwierdzić, że system Petalinux wydaje się optymalnym rozwiązaniem dla tworzonego rozwiązania. Jego modułowość, niewielki rozmiar generowanego jądra oraz możliwości rozbudowy sprawiają, że z powodzeniem może zostać wykorzystany w szerokiej klasie zastosowań. System Petalinux jest rozwiązaniem popularnym wśród osób tworzących systemy wbudowane wykorzystujące układy FPGA. Fakt ten ma niemałe znaczenie, ponieważ możliwość ewentualnego wsparcia oraz ilość dostępnej dokumentacji mają niejednokrotnie kluczowe znaczenie podczas realizacji projektu.

3. Uruchomienie systemu

Przeprowadzony proces uruchomienia systemu Petalinux w środowisku układu reprogramowalnego podzielić można na trzy główne etapy:

1. Konfiguracja otoczenia systemu.
2. Rozruch wersji bazowej systemu.
3. Rozbudowa systemu.

Podczas pierwszej fazy kluczowym zadaniem jest optymalny dobór używanych systemów operacyjnych oraz właściwe zestawienie środowiska programistycznego. Praca nad konfiguracją systemu Petalinux wymusza użycie do pracy systemu typu Linux, jednak brak kompatybilności oprogramowania Xilinx z większością tego typu systemów sprawia, że wybór staje się bardzo ograniczony. Jedynym w pełni zgodnym ze środowiskiem programistycznym firmy Xilinx systemem jest Red Hat Enterprise Linux. Wybór padł zatem na jego darmowy klon – system CentOS 3 (*Community Enterprise Operating System*) zbudowany w oparciu o jądro serii 2.4 umożliwiające bezproblemową współpracę ze sterownikami do karty Spartan 3E Starter Kit.

Kolejnym etapem tworzenia środowiska jest instalacja oprogramowania Xilinx. Podczas prowadzonych prac używano pakietu ISE i EDK w wersji 9.1 z dodatkami serwisowymi 3 (ISE) i 2 (EDK). Brak zgodności pomiędzy kolejnymi wersjami oprogramowania Xilinx powoduje, że wybór odpowiedniego środowiska może okazać się niezwykle istotny dla wykorzystania istniejących projektów. Próba wykorzystania narzędzia służącego do migracji projektu do nowszej wersji zwykle kończy się zniszczeniem plików źródłowych.

Faza konfiguracji środowiska kończona jest przez instalację programu Kermit wykorzystywanego do realizacji transmisji szeregowej z układem. Do głównych cech programu należą jego uniwersalność w dziedzinie wykorzystywanych mediów transportowych oraz możliwość tworzenia skryptów automatyzujących jego pracę. Przykład skryptu używanego podczas pracy z systemem Petalinux zamieszczony został na rysunku 4.

```
set line /dev/ttyS0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set key \127 \8
set key \8 \127
set window 5
```

Rys. 4. Skrypt automatyzujący komunikację z układem Spartan 3E Starter Kit

Pierwszy etap pomimo braku obiektywnych trudności okazał się stosunkowo czasochłonny, głównie ze względu na brak doświadczenia oraz niewielką ilość dostępnej dokumentacji. Złożoność projektu sprawia, że ewentualne błędy popełnione na tym etapie mogą okazać się kluczowe dla realizacji jego kolejnych faz.

Rozruch wersji bazowej systemu to proces nie nastręczający znacznych trudności pod warunkiem właściwej realizacji poprzedniej fazy projektu oraz poprawności używanych źródeł. Na tym etapie widoczna staje się przewaga systemu Petalinux nad poprzednikiem (μClinux). Jakość dokumentacji oraz oprogramowanie automatyzujące proces instalacji systemu w układzie w znacznym stopniu wyręczają użytkownika. Twórcy systemu nie ustrzegli się niedociągnięć i w skryptach służących do kopiowania obrazu systemu do pamięci trwałej urządzenia pojawiły się błędy. Stworzenie własnych skryptów rozwiązuje w pełni ten problem i system może zostać zainstalowany bez przeszkód. Efektem prac drugiego etapu jest działający system Petalinux w podstawowej funkcjonalności. Jest to podstawa dla dalszej rozbudowy i badań. System wygenerowany w efekcie tych działań

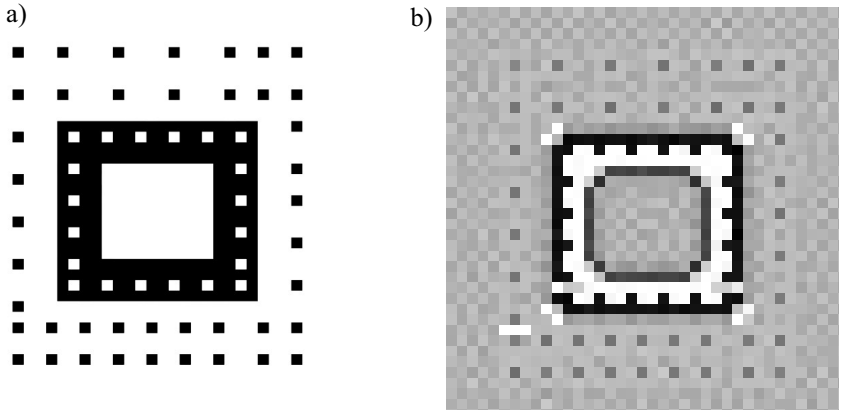
charakteryzuje się stosunkowo niewielką funkcjonalnością. Jego standardowe funkcje można by określić jako demonstracyjne. Rozbudowa systemu polega w głównej mierze na dołączaniu kolejnych komend z pakietów BusyBox oraz dziedziczonych z µClinuxa. Wydaje się, że jest to największa słabość testowanego rozwiązania. Z uwagi na liczne błędy w kodzie źródłowym systemu, zaimplementowanie wielu funkcji jest niemożliwe. Brak diagnostyki sprawia, że wykrycie wadliwych miejsc w kodzie okazuje się niezwykle kłopotliwe. Przyczyną takiego stanu rzeczy jest niewątpliwie otwarty charakter używanego oprogramowania. Pomimo opisanych trudności udało się wygenerować jądro gwarantujące systemowi względnie szeroką funkcjonalność. Urządzenie dysponujące tak szerokim zakresem komend z powodzeniem może pracować jako warstwa sterująca systemu wbudowanego.

4. Eksperymenty i testy

Prowadzone prace miały na celu zbadanie przydatności systemu do realizacji programowego przetwarzania obrazów oraz kontroli peryferii zainstalowanych w układzie.

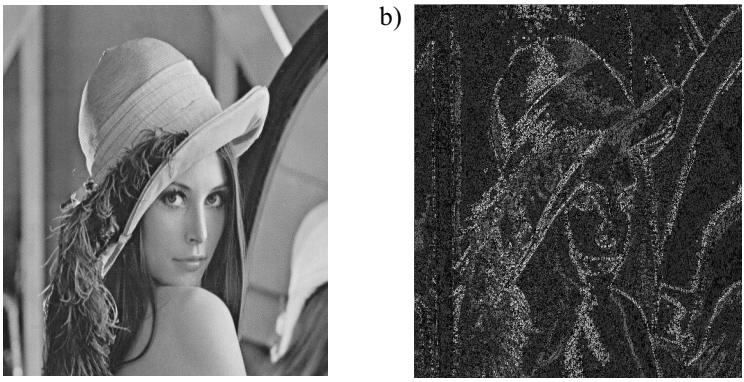
4.1. Aplikacja do przetwarzania obrazów

Stworzony program służy do obróbki obrazów zapisanych w formacie *bmp*, w których każdy piksel opisany jest przez osiem bitów. Ponieważ większość realizowanych operacji odbywa się na obrazach czarno-białych, jakość obrazu o 256 stopniach szarości jest wystarczająca. Zadaniem programu jest realizacja przykładowych operacji przetwarzania obrazów za pomocą konwolucyjnych filtrów liniowych i nieliniowych. Na rysunkach 5 i 6 zilustrowano operacje uśredniania i wykrywania krawędzi.



Rys. 5. Przykład działania filtru uśredniającego: a) obraz testowy;
b) obraz testowy po uśrednieniu

a) b)



Rys. 6. Przykład operacji wykrywania krawędzi przy zastosowaniu filtrów Sobela: a) obraz wejściowy; b) obraz wynikowy

Aplikacja została napisana w całości w języku C z wykorzystaniem najbardziej podstawowych funkcji. Podyktowane jest to ograniczeniami narzuconymi przez docelowe środowisko jej działania. Wykorzystanie zapisu jednego piksela na ośmiu bitach pozwoliło na znaczne uproszczenie procesu przetwarzania ograniczając je do manipulacji wartościami elementów tablicy typu char.

4.2. Aplikacja do kontroli peryferii

W ramach testów systemu stworzona została aplikacja służąca do realizacji komunikacji z peryferiami układu. Jej zadaniem jest odbieranie sygnałów z przycisków sterujących karty oraz umieszczanie odpowiednich wpisów w założonym w tym celu pliku tekstowym.

Działanie aplikacji zostało oparte o funkcje i interfejsy dostarczane przez biblioteki języka C, przeznaczone do obsługi urządzeń zewnętrznych w systemach Linuxowych. Zdarzenia obsługiwane przez aplikację opisane są za pomocą struktury *Event* (rys. 7).

```
struct event {
        struct timeval time;
        unsigned short type;
        unsigned short code;
        unsigned int value;
              };
```

Rys. 7. Struktura Event używana do obsługi peryferii

Na rysunku 8 znajduje się fragment funkcji przeznaczonej do odczytu wartości z przycisków sterujących. Przykład pokazuje sposób wykorzystania struktury Event.

```
if(event->type != EV_KEY)
            return;
   switch(event->code)
   {
        case KEY_LEFT:
            left = event->value;
            break;
       case KEY_RIGHT:
            right = event->value;
            break;
       case KEY_UP:
            up = event->value;
            break;
       case KEY_KPENTER:
            centre = event->value;
            break;
   };
```

Rys. 8. Funkcja wykorzystująca strukturę Event

Po kontroli poprawności typu zdarzenia sprawdzana jest jego wartość, która w dalszej części kodu aplikacji jest zapisywana do pliku tekstowego. Opisywana aplikacja jest jedynie przykładem możliwości, jakie oferuje połączenie peryferii układu Spartan 3E z systemem operacyjnym Petalinux i aplikacjami działającymi pod jego kontrolą. Jest to świetny przykład na zasadność idei tworzenia mocno zintegrowanych systemów łączących zalety rozwiązań sprzętowych i programowych.

Aplikacje stworzone w ramach pracy mają głównie charakter demonstracyjny. Ilość zasobów możliwych do wykorzystania po wygenerowaniu architektury sprzętowej okazała się niewystarczająca do dalszego poszerzania funkcjonalności systemu. Należy jednak sądzić, że zwiększenie pojemności układu FPGA zaowocowałoby większą ilością obsługiwanych peryferii, co w konsekwencji prowadziłoby do rozbudowy stworzonego systemu.

5. Doświadczenia i wnioski

Realizacja prac opisanych w artykule przyczyniła się do zgromadzenia szerokiej wiedzy z zakresu architektur sprzętowych, systemów operacyjnych oraz metod realizacji ich współdziałania. Niewątpliwie doświadczenia zdobyte podczas tworzenia od podstaw tego systemu będą niezwykle cenne pod kątem prowadzenia dalszych badań. Po zakończeniu projektu nasuwają się następujące wnioski:

- wybór środowiska w którym prowadzone są badania, ma kluczowy wpływ na ich rezultat;
- oprogramowanie programistyczne Xilinx jest wrażliwe na wersję systemu operacyjnego oraz nie zapewnia kompatybilności projektów z różnych wersji;

- układ XC3S500E pozwala na realizację jedynie podstawowych funkcji;
- środowisko programistyczne EDK umożliwia wygenerowanie z gotowych komponentów architektury sprzętowej systemu;
- rozbudowa systemu ograniczona jest jedynie możliwościami sprzętowymi zastosowanego układu;
- system Petalinux umożliwia stworzenie rozwiązania o wysokiej funkcjonalności i z pewnością nie jest ograniczeniem dla dalszych prac;
- otwarty charakter systemu Petalinux może być przyczyną wystąpienia błędów podczas pracy nad systemem;
- środowisko wspierające pracę z systemem Petalinux dostarczane wraz z nim nie pozbawione jest błędów, których wykrycie może okazać się uciążliwe;
- stworzone aplikacje działają w sposób efektywny pomimo ograniczonych możliwości sprzętowych systemu;
- zaproponowane rozwiązanie posiada funkcjonalność umożliwiającą wykorzystanie w systemie wbudowanym.

6. Podsumowanie

Celem opisanych w artykule prac było stworzenie systemu wbudowanego z użyciem układu rekonfigurowalnego i system typu Linux. W trakcie prac stosowano układ Spartan 3E Starter Kit oraz systemy operacyjne μClinux oraz Petalinux. Efektem prac jest autonomiczny system wbudowany o funkcjonalności zadowalającej z punktu widzenia stawianych założeń. Cel pracy, jakim było powtórzenie eksperymentów nad tego typu systemem oraz próba jego rozwoju, został osiągnięty. Pomimo badawczego charakteru rozwiązania, osiągnięte efekty są dowodem tezy na zasadność wykorzystania do budowy systemów wbudowanych układów rekonfigurowalnych w połączeniu z systemami operacyjnymi. Jak widać, rozwiązanie tego typu cechuje się sporą elastycznością i funkcjonalnością, czyli cechami które powinny charakteryzować system wbudowany.

Dalszy rozwój zaproponowanego rozwiązania powinien koncentrować się na zapewnieniu sprawnej komunikacji pomiędzy warstwą programową i sprzętową systemu. Przesunięcie części logiki do układu FPGA zbliżyłoby omawiany system do idei komputera rekonfigurowalnego [2], wpływając w zdecydowany sposób na jego efektywność. Równie ważnym z punktu widzenia systemu wbudowanego parametrem jest czas generowania odpowiedzi. Działania mające na celu spełnianie przez sprzęt twardych ograniczeń czasowych z pewnością przyczyniłyby się do zwiększenia zakresu zastosowań systemu.

Przedstawione rozwiązanie pomimo niewątpliwych słabości może być dowodem na to, że systemy wbudowane bazujące na układach reprogramowalnych obdarzone są wysokim potencjałem rozwojowym.

Literatura

[1] Compton K., Hauck S., *An Introduction to Reconfigurable Computing*. Northwestsern University, Dept of ECE Technical Report, 1999.
[2] Compton K., Hauck S., *Reconfigurable Computing: A Survey for Systems and Software.* ACM Computing Surveys, vol. 34, No. 2, June 2002.
[3] Dellson A., Sandberg G., Mohl S., *Turning FPGAs into Supercomputer*. 2006.
[4] Hartenstein R., *A Decade of Reconfiguralbe Computing: a Visionary Retrospective*. CS Dept, University of Keiserslautern, Germany.
[5] Hauck S., *The Future of Reconfigurable System*. 5th Canadian Conference on Field Programmable Devices,Montreal, June 1998.
[6] Russek P., Wiatr K., *Perspektywy przyśpieszania obliczeń w instalacjach o wielkich mocach obliczeniowych za pomocą układów logiki rekonfigurowalnej*. Automatyka (półrocznik AGH), t. 9, z. 3, 2005.
[7] Todman T.J., Constantinides G.A., Wilton S.J.E., Mencer O., Luk W., Cheung P.Y.K., *Reconfigurable Computing: architectures and design methods*. IEEE Proc.-Comput. Digit. Tech., vol. 152 No. 2, March 2005.
[8] Wu J., Williams J., *Creating a simple uClinux ready Microblaze Design*. Ocotber 2005.
[9] Xilinx, *MicroBlaze Processor Reference Guide, Embedded Development Kit 9.1i*. 2006.
[10] Xilinx, *Spartan-3E FPGA Family: Complete Data Sheet*. May 2007.
[11] Xilinx, *MicroBlaze – Frequently Asked Questions.*
[12] Xilinx, *Xilinx Spartan-3E Family – Frequently Asked Questions.*
[13] http://developer.petalogix.com/.
[14] http://www.uclinux.org.