Robert ŁUKASZEWSKI, Patryk PANTER
POLITECHNIKA WARSZAWSKA, INSTYTUT RADIOELEKTRONIKI,
ul. Nowowiejska 15/19, 00-665 Warszawa

Biblioteka SubVI do programowania platformy CompactRIO z wbudowaną aplikacją FPGA

Dr inż. Robert ŁUKASZEWSKI

Od 1997 roku pracownik pracowni Komputerowej Techniki Pomiarowej, Instytutu Radioelektroniki Politechniki Warszawskiej. Prowadzone prace naukowo-badawcze oraz publikacje ściśle związane są z projektowaniem, modelowaniem i implementacją oprogramowania rozproszonych systemów pomiarowo-sterujących. Szczegółowy obszar zainteresowań obejmuje interfejsy pomiarowe, sieci komputerowe, projektowanie systemów, języki formalne, bezpieczeństwo systemów.

e-mail: rlukaszewski@ire.pw.edu.pl

Inż. Patryk PANTER

Od 2008 roku student Politechniki Warszawskiej na wydziale Elektroniki i Technik Informacyjnych, specjalizacja Radiokomunikacja i Techniki Multimedialne. Aktualne obszary zainteresowań obejmują zagadnienia związane z: systemami pomiarowo-sterującymi , systemami czasu rzeczywistego, sieciami komputerowymi, projektowaniem systemów w oparciu o układy FPGA oraz układy mikroprocesorowe, a także problematykę dotyczącą systemów rozproszonych.

e-mail: P.Panter@stud.elka.pw.edu.pl

Streszczenie

W artykule przedstawiono autorski projekt biblioteki przeznaczonej do programowania platformy cRIO współpracującą z wbudowaną aplikacją FPGA. W pierwszej części artykułu omówiono podstawowe własności technologii NI RIO, architekturę platformy cRIO oraz sposób projektowania oprogramowania systemu wykorzystującego tę platformę. Opisano stworzoną bibliotekę programową dla środowiska LabVIEW oraz przedstawiono sposób jej wykorzystania w projektach.

Słowa kluczowe: CompactRIO, FPGA, LabVIEW, systemy pomiarowo-sterujące, systemy czasu rzeczywistego.

SubVI software library for programming the CompactRIO platform with embedded FPGA application

Abstract

The paper presents an original software library for the CompactRIO platform with embedded FPGA application that is used in LabVIEW projects. There are discussed the basic properties of NI RIO technology, hardware architecture of CompactRIO platform and the way of designing systems based on this controller. The use of it gives new didactic capabilities due to a combination of several areas of engineering which are programming the classical microcontroller and FPGA systems, signal processing, development of distributed systems or real-time measuring and distribution systems. The paper also describes the software library for LabVIEW environment and the way of using it in user programs, including student projects in the classes of the CompactRIO platform. It also presents a user library interface and a detailed description of it. The basic benefits of its application and the system architecture which can be realized with use of it are given. The use of the described library allows focusing on the issues directly related to the topic of taught subjects. During the implementation of the tasks set out in the paper, students acquire practical skills to design and run real-time systems. They also get experience in the use of advanced inter-process and network communications and become aware of their limits.

Keywords: CompactRIO, FPGA, LabVIEW, measuring and control systems, real-time systems.

1. Wstęp

Integracja przyrządów pomiarowych z systemami wbudowanymi, strukturami programowalnymi FPGA (ang. Field Programmable Gate Array) oraz różnymi metodami transmisji danych. zaowocowała powstaniem nowej klasy urządzeń zdolnych do programowego definiowania funkcji pomiarowych. Urządzenia te zostały wyposażone w możliwości wymiany informacji z innymi węzłami rozproszonych systemów pomiarowo-sterujących (RSPS). W wielu przypadkach nowoczesne urządzenia pomiarowo-sterujące pracują pod kontrolą systemu operacyjnego czasu rzeczywistego i są zdolne do spełnienia twardych wymagań pracy w czasie rzeczywistym (RT – ang. Real Time).

W celu zaspokojenia rosnących potrzeb związanych z programowaniem systemów sterowania przemysłowego czołowe firmy z branży automatyki stworzyły nową klasę sterowników przemysłowych - PAC (ang. Programmable Automation Controller). Sterowniki PAC łączą w sobie wytrzymałość i odporność sterownika PLC (ang. Programmable Logic Controller) z funkcjonalnością komputera osobistego w ramach jednej otwartej i elastycznej architektury programowej. Sterowniki te zapewniają szerokie możliwości programowe, takie jak zaawansowane sterowanie, łączność, rejestrowanie danych i przetwarzanie sygnałów.

W dalszej części artykułu przedstawiono autorską bibliotekę programową SubVI dla środowiska LabVIEW do programowania platformy NI cRIO [1] z wbudowaną aplikacją FPGA. Celem powstania tej biblioteki było ułatwienie projektowania i realizacji RSPS czasu rzeczywistego przy wykorzystaniu platformy cRIO.

2. Platforma CompactRIO

Technologia RIO (ang. Reconfigurable Input/Output) [1] firmy National Instruments (NI) umożliwia tworzenie rozproszonych, rekonfigurowanych systemów sterowania i akwizycji danych, umożliwia definiowanie własnych sprzętowych obwodów pomiarowych przy wykorzystaniu układów FPGA do np. szybkiej analizy sygnałów i podejmowania decyzji już na poziomie układu we/wy. Dzięki zintegrowanemu układowi FPGA, sterownikowi zarządzanemu przez RTOS oraz szerokiej gamie modułów wejściowych i wyjściowych (we/wy) dokładne poznanie technologii RIO gwarantuje nabycie umiejętności koniecznych do projektowania RSPS z wykorzystaniem większości obecnie dostępnych na rynku nowoczesnych platform sprzętowych. Z drugiej strony technologia RIO umożliwia programowanie graficzne w środowisku LabVIEW (w tym programowanie układu FPGA).

Platforma NI cRIO jest zwartym systemem skonstruowanym z myślą o zastosowaniach przemysłowych oraz w systemach akwizycji danych, w których niezbędna jest wysoka wydajność, niezawodność, dyscyplina czasowa oraz elastyczność. Największe możliwości projektowe daje konfiguracja ze sterownikiem RTOS dołączanym do obudowy cRIO. W przemysłowej obudowie zintegrowany jest układ FPGA podłączony bezpośrednio do modułów pomiarowych w postaci modułów we/wy serii „C" [2]. Każdy z dostępnych sterowników posiada wbudowany procesor przemysłowy oraz zainstalowany RTOS Wind River VxWorks [3]. Aplikacje opracowane na etapie projektowania osadzane są w układach FPGA przez środowisko LabVIEW [4]. Graficzne diagramy tłumaczone są do postaci tekstowej kodu VHDL (ang. Very high speed integrated circuits Hardware Description Language). Następnie wywoływane są narzędzia kompilacji Xiling ISE [5].

Projektowanie oprogramowania platformy cRIO [2] składa się zazwyczaj z trzech zadań: oprogramowania układu FPGA, stworzenia aplikacji RTOS oraz aplikacji operatorskiej działającej pod kontrolą systemu Microsoft Windows.

Programowanie cRIO wymaga m. in. stworzenia aplikacji osadzonej w układzie FPGA do sterowania wejściami i wyjściami w modułach we/wy. Jest to proces długotrwały i wymagający specjalistycznej i szczegółowej wiedzy.

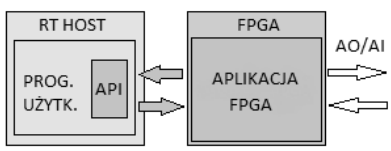
3. Architektura oprogramowania

Zrealizowane oprogramowanie jest komponentem programowym, uruchamianym w całości na platformie cRIO. Zbudowane jest z dwóch współpracujących ze sobą elementów.

Pierwszym z nich jest aplikacja osadzana w układzie FPGA. Realizuje ona wszystkie funkcje związane z generacją oraz akwizycją sygnałów, zaś dla jej użytkownika stanowi swego rodzaju „czarną skrzynkę".

Drugim elementem jest biblioteka funkcji SubVI, które składają się na aplikacyjny interfejs programowy - API (ang. Application Programming Interface). API to pośredniczy w wymianie informacji między użytkownikiem a wspomnianą wcześniej wbudowaną aplikacją FPGA. Interfejs ten dostępny jest z poziomu programu kontrolera RT i umożliwia „ukrycie" przed programistą szczegółowych mechanizmów komunikacyjnych.

W uproszczonej architekturze stworzonego oprogramowania można wyróżnić bloki stanowiące wspomniane API oraz program dla FPGA (rys. 1). Strzałki znajdujące się między nimi symbolizują ukrytą przed programistą komunikację tych dwóch podzespołów. Aplikacja użytkownika korzysta z funkcji biblioteki SubVI.



Rys. 1. Uproszczona architektura oprogramowania
Fig. 1. Simplified architecture of the software

Zrealizowana, wbudowana aplikacja FPGA jest kompletnym programem LabVIEW umożliwiającym realizację pomiarów w deterministycznym czasie. Oznacza to, że osoba budująca projekt w jego oparciu może skupić się na programowaniu kontrolera RT co nie wymaga tak długiego czasu realizacji jak program dla FPGA. Z pewnością zdaje sobie z tego sprawę producent środowiska LabVIEW, ponieważ umieścił on w swoim produkcie specjalny tryb uruchamiania aplikacji na platformie cRIO o nazwie *Scan mode*. W trybie tym układ FPGA nie bierze jawnego udziału w komunikacji pomiędzy programem dla kontrolera RT a modułami we/wy. Upraszcza to znacznie realizację projektów, lecz tryb ten ma wyraźne ograniczenia dotyczące szybkości przesyłu danych i nie nadaje się do realizacji bardziej zaawansowanych projektów. W związku z tym powstał pomysł stworzenia własnego oprogramowania będącego odpowiednikiem trybu *Scan mode*, który umożliwiłby realizację takich projektów.

Ogólna zasada działania zrealizowanego oprogramowania polega na wytworzeniu sygnałów sterujących wyjściami modułu AO (ang Analog Output), a następnie akwizycji sygnałów na wejściach modułu AI (ang. Analog Input). Jednocześnie może się odbywać generacja i akwizycja sygnałów w maksymalnie czterech torach pomiarowych. W programie użytkownika do funkcji API muszą zostać przekazane wszystkie niezbędne parametry pomiarowe, takie jak liczba wykorzystywanych torów pomiarowych oraz częstotliwość sygnałów dla każdego z nich. Dane te są przekazywane do układu FPGA za pomocą standardowych wejść i są tam doprowadzane do poszczególnych modułów wewnętrznych. Jeżeli dane te są poprawne, rozpoczyna się generacja od jednego do czterech niezależnych sygnałów sinusoidalnych na wyjściach analogowych modułu AO (np. NI-9263). Następnie wbudowana aplikacja FPGA oczekuje na sygnał rozpoczęcia pomiaru, który powinien nadejść z programu użytkownika poprzez API. Jego stan pozytywny oznacza rozpoczęcie pracy modułów akwizycji danych we wszystkich aktywnych torach pomiarowych. Dane te pochodzą z wejść modułów AI (np. NI-9205) i są umieszczane w wewnętrznych kolejkach FIFO (ang. First In, First Out) układu FPGA. Stąd pobierane są przez moduł komunikacji, który łączy dane w grupy i umieszcza w zewnętrznym FIFO DMA (ang. Direct Memory Access) [6], do którego możliwy jest dostęp z poziomu kontrolera RT. API cyklicznie pobiera z FIFO DMA próbki, dzieli je na tablice i zwraca do programu użytkownika. Każda taka tablica zawiera próbki z jednego wirtualnego toru pomiarowego.

Zadaniem **aplikacji FPGA** jest generacja i akwizycja sygnałów zgodnie z zadanymi parametrami oraz komunikacja z API. Realizowane jest to za pomocą trzech modułów składających się z równoległych pętli, po jednej dla każdego modułu, których działanie jest od siebie niezależne.

Zadaniem **modułu generacji** jest wysterowanie wyjść modułu AO w taki sposób, aby uzyskać przebieg sinusoidalny o zadanej częstotliwości. Moduł ten w pierwszej kolejności generuje ciąg próbek odpowiadający przebiegowi sygnału sinusoidalnego o zadanej przez użytkownika częstotliwości. Ze względu na działanie maksymalnie czterech torów pomiarowych, generowane jest sekwencyjnie po jednej próbce dla każdego z tych torów. Każda z tych próbek jest przekazywana do modułu AO i wystawiana na jego styk skojarzony z odpowiednim torem. W ten sposób dla aktywnych czterech torów co czwarta generowana próbka będzie odpowiadała każdemu z nich.

Zastosowany moduł AO NI-9263 posiada tylko jeden przetwornik C/A co skutkuje podziałem maksymalnej częstotliwości próbkowania na liczbę aktywnych torów pomiarowych. Częstotliwość ta dla jednego toru wynosi 400, 200, 133 i 100 [kHz] dla odpowiednio 1, 2, 3 i 4 jednocześnie użytych torów pomiarowych.

Moduł akwizycji w każdym cyklu pobiera próbki sygnałów dla styków modułu AI NI-9205, które są wejściami analogowymi poszczególnych torów pomiarowych. To, który pin modułu AO skojarzyć z którym pinem modułu AI jest określone przez użytkownika poprzez parametry wejściowe API opisane w rozdziale 4.

Ze względu na konieczność sekwencyjnego pobierania pojedynczych próbek dla kolejnych torów pomiarowych oraz istnienie pojedynczego przetwornika A/C w module NI-9205 również w tym przypadku występuje podział maksymalnej częstotliwości próbkowania na liczbę aktywnych torów pomiarowych. Jest to 250, 125, 83,3 i 62,5 [kHz] dla odpowiednio 1, 2, 3 i 4 torów pomiarowych. Każda pobrana próbka jest umieszczana w jednym z 4 wew. FIFO układu FPGA. Każda taka kolejka odpowiada jednemu torowi pomiarowemu.

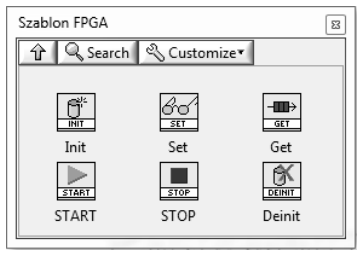
Moduł komunikacji na bieżąco sprawdza czy w każdym aktywnym wew. FIFO znajduje się co najmniej jedna próbka. W momencie, gdy sytuacja taka nastąpi, moduł komunikacji wyciąga po jednym elemencie z każdego FIFO i po kolei umieszcza je w FIFO DMA. W ten sposób w kolejce tej zawsze powinna się znajdować liczba próbek będąca całkowitą wielokrotnością liczby aktywnych torów pomiarowych. Jest to istotne przy późniejszym odbieraniu tych próbek z kolejki.

Do komunikacji z API wykorzystano tylko jedną kolejkę FIFO DMA ze względu na ograniczenia w zakresie komunikacji z kontrolerem RT używanej w projektach platformy cRIO NI-9073.

Zaprojektowane **API** jest zbiorem 6 funkcji LabVIEW w postaci SubVI, które opakowują polecenia związane z inicjalizacją i obsługą programu dla układu FPGA oraz komunikacją z nim. Główną zaletą tego rozwiązania jest fakt, że pozwala to ukryć złożoną logikę aplikacji FPGA oraz dość nieczytelną jego obsługę. Wszystkie funkcje API zostały przygotowane w sposób zgodny z konwencjami stosowanymi w innych bibliotekach LabVIEW.

4. API

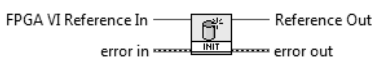
W celu ułatwienia korzystania ze zrealizowanej biblioteki SubVI stworzono paletę zawierającą wszystkie funkcje API (rys. 2). Paleta o nazwie „Szablon FPGA" dostępna jest w aplikacji kontrolera RT z okna menadżera funkcji środowiska LabVIEW.



Rys. 2. Paleta funkcji biblioteki SubVI
Fig. 2. Palette of SubVI library functions

Funkcja **Init** (rys. 3) jest odpowiedzialna za inicjalizację aplikacji FPGA. Powinna ona zostać wykonana na samym początku działania programu kontrolera RT. Żadna inna funkcja z palety „Szablon FPGA" nie będzie działać poprawnie, jeżeli wbudowana aplikacja FPGA nie zostanie zainicjalizowana i nie zostanie do niej doprowadzona referencja podprogramu Init.

Wejście błędów, które wystąpiły przed wykonaniem funkcji Init oznaczono *Error in*. Do wejścia *FPGA VI Reference In* należy doprowadzić numer referencyjny uruchomionej aplikacji FPGA. Wyjście *Reference Out* zwraca numer referencyjny tej aplikacji, który należy doprowadzić do wejść referencyjnych pozostałych funkcji z palety „Szablon FPGA". Natomiast wyjście *Error out* zwraca informację o błędach.



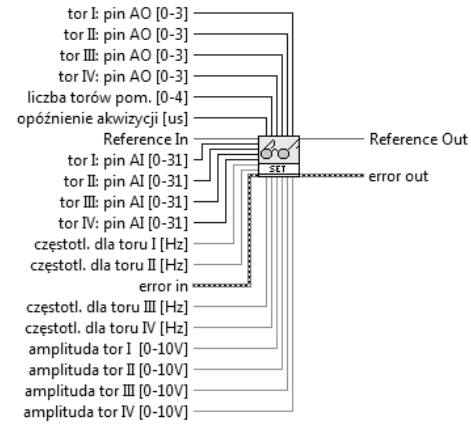
Rys. 3. Funkcja Init
Fig. 3. Init function

Funkcja **Start** (rys. 4) jest odpowiedzialna za rozpoczęcie pomiaru. Będzie on trwał do czasu pojawienia się funkcji Stop. Jej wywołanie powoduje zatrzaśnięcie większości wartości wprowadzanych do aplikacji FPGA za pomocą funkcji Set. Wejścia, których to nie dotyczy to: częstotliwość i amplituda generacji oraz opóźnienie akwizycji.

Rys. 4. Funkcja Start
Fig. 4. Start function

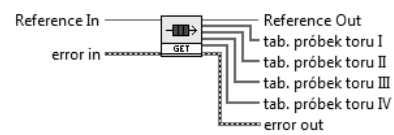
Funkcja **Set** (rys. 5) jest odpowiedzialna za dostarczenie do aplikacji FPGA danych konfiguracyjnych i powinna zostać wykonana przed rozpoczęciem procesu pomiarowego.

Wejście *Tor i: pin AO* funkcji Set określa numer pinu modułu analogowych wyjść NI-9263, na którym ma się odbywać generacja sygnału dla i-tego toru pomiarowego. Parametr *Liczba torów pom.* określa liczbę torów pomiarowych, które będą aktywne. Dane konfiguracyjne dla torów nieaktywnych będą ignorowane przez aplikację FPGA. Zmienna na wejściu *Opóźnienie akwizycji* określa czas jaki upłynie między dwiema operacjami pobrania próbki sygnału przez moduł AI NI-9205. W przypadku gdy czas ten jest krótszy niż okres próbkowania (np. 4us dla jednego aktywnego toru pomiarowego), to nie ma on wpływu na okres akwizycji. Numer referencyjny aplikacji FPGA (patrz funkjca Init) należy dołączyć do wejścia *Reference in*. Przez odpowiednie wysterowanie wejść *Tor i: pin AI* można określić numer pinu modułu AI NI-9205, na którym ma się odbywać akwizycja sygnału dla i-tego toru pomiarowego. Dla każdego z tych torów należy podać częstotliwość i amplitudę sygnału generowanego na analogowym wyjściu przez wejścia funkcji Set, odpowiednio: *Częstotl. dla toru i* oraz *Amplituda dla toru i*.



Rys. 5. Funkcja Set
Fig. 5. Set function

Funkcja **Get** (rys. 6) jest odpowiedzialna za zwracanie danych zmierzonych przez aplikację FPGA. Funkcja ta musi być umieszczona w oddzielnej pętli typu Timed Loop o okresie cyklu równym 3ms. Wyjście *Tab. próbek toru i* zwraca tablicę zawierająca 1200 próbek sygnału. Jeżeli ich liczba jest mniejsza, to wszystkie pozostają w FIFO i funkcja Get w danej iteracji zwróci puste tablice.



Rys. 6. Funkcja Get
Fig. 6. Get function

Dodatkowo w API umieszczono funkcję Deinit służącą do zamykania aplikacji FPGA oraz funkcję Stop powodującą wysłanie poprzez zwykłą zmienną sygnału zakończenia pomiaru.

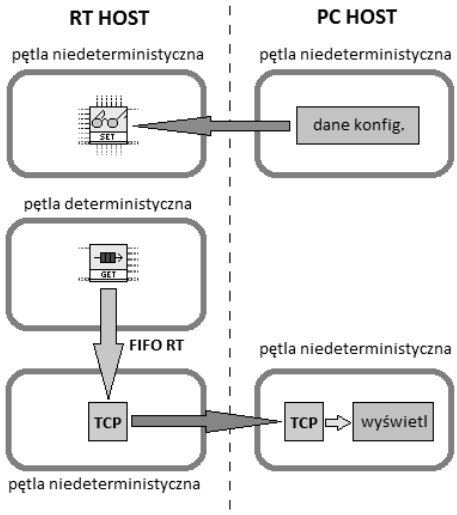
5. Korzystanie z biblioteki SubVI

Zrealizowane oprogramowanie ukrywa przed programistą mechanizmy swojego działania. Przy korzystaniu z jego API należy jednak kierować się pewnymi regułami, które pozwolą uniknąć nieoczekiwanej lub błędnej pracy systemu. Ograniczenia aplikacji FPGA powodują m.in., iż w jednym programie użytkownika może istnieć jedna instancja tej aplikacji.

Należy zwrócić uwagę, że po wykonaniu funkcji Init może upłynąć pewien czas, zanim układ FPGA zdąży się zainicjować. Aby upewnić się, że aplikacja FPGA pracuje należy skorzystać z odpowiedniego pola sygnalizacyjnego dostępnego w funkcji Set. Nosi ona nazwę „FPGA Ready?" i umożliwia poinformowanie użytkownika projektowanego systemu o tym, czy ładowanie aplikacji FPGA jeszcze trwa.

Innym istotnym faktem jest to, że dane, które są zwracane przez funkcję Get, nie muszą być przez użytkownika odbierane. Jeżeli jego program zignoruje tablice z danymi pomiarowymi to jedynym niepożądanym zjawiskiem będzie ich utrata.

Prawidłowe wykorzystanie opisywanego oprogramowania wymaga implementacji odpowiedniej architektury samej aplikacji użytkownika (rys. 7). Architektura ta musi być też zgodna z zaleceniami [2] producenta platformy cRIO i środowiska LabVIEW. Logikę każdego oprogramowania opartego na zrealizowanej bibliotece SubVI i wbudowanej aplikacji FPGA można zawrzeć w pięciu pętlach programowych oraz odpowiedniej komunikacji międzyprocesowej.



Rys. 7. Przepływ danych między pętlami w programie użytkownika
Fig. 7. Data flow between loops in the user application

Większość pętli w zalecanej architekturze programu użytkownika to pętle niedeterministyczne, czyli nieobciążane sztywnymi wymaganiami czasowymi. Wyjątkiem jest tutaj pętla zawierająca funkcję Get. W jej przypadku konieczne jest wykonanie każdej iteracji w z góry określonym czasie, ponieważ w przypadku niespełnienia tego warunku może nastąpić utrata danych.

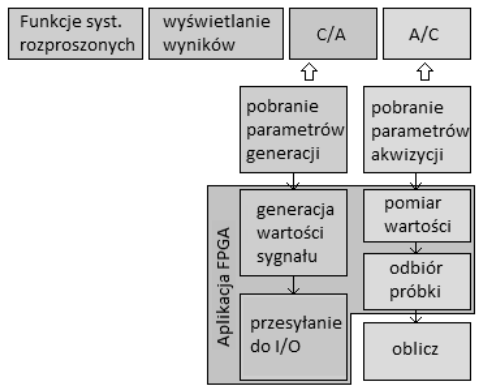
Analizę zalecanej architektury należy rozpocząć od osadzonej w aplikacji dla komputera nadzorczego (PC HOST) pętli pobierającej, np. od operatora systemu, dane konfiguracyjne, w tym parametry generacji i akwizycji. Następnie dane konfiguracyjne są przesyłane do kontrolera RT. Komunikacja może odbywać się przy użyciu zmiennych sieciowych [7] lub innych mechanizmów sieciowej wymiany danych dostępnych w LabVIEW.

Dane konfiguracyjne są następnie dostarczane do szablonu FPGA za pośrednictwem funkcji Set należącej do API szablonu. Szczególnym rodzajem informacji konfiguracyjnej jest sygnał wyzwolenia pomiaru. Dostarczany on jest do tej samej pętli, lecz jego wartość pozytywna powoduje wywołanie funkcji Start, negatywna natomiast wywołuje funkcję Stop.

Opisane powyżej pętle biorą udział w komunikacji przebiegającej od aplikacji PC HOST do kontrolera RT. Przepływ danych w przeciwnym kierunku rozpoczyna się od pętli deterministycznej, zawierającej funkcję Get. Odbiera ona tablice z danymi, które następnie przekazywane są do pętli komunikacji. W celu zachowania determinizmu pętli zawierającej funkcję Get, konieczne jest wykorzystanie kolejki FIFO RT jako pośrednika w komunikacji. Ze względu na brak sztywnych wymagań czasowych pętli komunikacji, do przekazania danych w kierunku aplikacji dla PC można np. wykorzystać niedeterministyczny protokół TCP. Po odebraniu danych ze strumienia TCP i po ich odpowiednim przetworzeniu mogą zostać wyświetlone na ekranie PC HOST lub zmagazynowane przy użyciu wbudowanych mechanizmów LabVIEW.

Realizacja przykładowego projektu polega na zaprojektowaniu dwu aplikacji zgodnie z architekturą cRIO. Pierwsza z nich, osadzona w sterowniku RT, ma na celu: wysłanie i odebranie danych do i z aplikacji FPGA, wstępne ich przetworzenie, przekazanie danych sterujących do niej oraz komunikacji z aplikacją nadzorczą za pomocą zmiennych sieciowych. Druga aplikacja działająca na komputerze nadzorczym służy do komunikacji z użytkownikiem, z bazą danych i z aplikacją czasu rzeczywistego.

Realizowane projekty wykorzystują opisywaną bibliotekę SubVI oraz wbudowaną aplikację FPGA. W znacznym stopniu upraszcza to ich realizację zdejmując z programisty konieczność realizacji niektórych funkcji poprzez ich zautomatyzowanie. Tyczy się to głównie algorytmów związanych z generacją i akwizycją danych. W typowych projektach istnieje kilka obszarów, które mogą zostać zrealizowane za pomocą opisywanego oprogramowania. Zgodnie z opisanymi wcześniej zaleceniami dla programu użytkownika obszar działania wbudowanej aplikacji FPGA (rys. 8) obejmuje znaczną część działań związanych z generacją i akwizycją sygnałów przez co pozwala projektantowi skupić się na oprogramowaniu kontrolera czasu rzeczywistego oraz warstwach prezentacji i sieciowej.



Rys. 8. Uproszczona architektura projektu użytkownika
Fig. 8. Simplified user project architecture

6. Podsumowanie

Platforma cRIO pozwala na projektowanie systemów charakteryzujących się wysoką wydajnością w porównaniu z rozwiązaniami czysto programowymi, porównywalną z rozwiązaniami czysto sprzętowymi. Dzięki otwartej architekturze, oprócz zalet technicznych, platforma cRIO ma duży potencjał dydaktyczny.

W artykule opisano autorskie oprogramowanie na platformę cRIO. W pierwszej części artykułu omówiono podstawowe własności technologii NI RIO, architekturę sprzętową platformy cRIO oraz sposób projektowania oprogramowania systemu wykorzystującego tę platformę. Opisano stworzoną bibliotekę programową dla środowiska LabVIEW wraz ze zrealizowaną aplikacją FPGA. oraz przedstawiono sposób jej wykorzystania w programach użytkownika. Wykorzystanie zaprojektowanego oprogramowania pozwala na skupienie uwagi projektanta na programowaniu kontrolera RT oraz komputera nadzorczego.

7. Literatura

[1] National Instruments, NI CompactRIO – Reconfigurable Control and Acquisition System: http://zone.ni.com/devzone/cda/tut/p/id/2856, 28.03.2012.
[2] Łukaszewski R., Bilski P., Mroczek K.: Wykorzystanie w dydaktyce rekonfigurowalnych przyrządów pomiarowo-sterujących i systemów wbudowanych, PAK 2011 nr 11, s. 1329-1333, 2011.
[3] Wind River, Build reliable and optimized embedded systems with the world's most widely adopted RTOS: http://www.windriver.com/products/vxworks/, 12.2012.
[4] National Instruments, LabVIEW FPGA Compilation Process: From Run Button to Bitfile: http://www.ni.com/white-paper/9381/en, 03.01.2012.
[5] Denton J. D.: Programming Logic Fundamentals Using Xilinx ISE and CPLDs, Pretince Hall, 203 pages. Introduction to PLDs using Xilinx ISE, 2004.
[6] National Instruments,Using DMA FIFO to Develop High-Speed Data Acquisition Applications for Reconfigurable I/O Devices, http://www.ni.com/white-paper/4534/en, 03.03.2012.
[7] National Instruments, Using the LabVIEW Shared Variable, http://www.ni.com/white-paper/4679/en, 04.03.2012.

otrzymano / received: 23.01.2013
przyjęto do druku / accepted: 01.03.2013 *artykuł recenzowany / revised paper*