Biuletyn WAT
Vol. LVII, Nr 4, 2008

Algorytmy i metody dwuprocesorowego sterowania precyzyjnym licznikiem czasu

TADEUSZ SONDEJ, MACIEJ GOŁASZEWSKI

Wojskowa Akademia Techniczna, Wydział Elektroniki, Instytut Telekomunikacji,
00-908 Warszawa, ul. S. Kaliskiego 2

Streszczenie. W artykule przedstawiono projekt oprogramowania systemu wieloprocesorowego, składającego się z dwóch procesorów programowych Nios II firmy *Altera* i precyzyjnego licznika czasu o rozdzielczości około 80 ps. Pierwszy procesor odpowiedzialny jest za komunikację systemu przez interfejs Ethernet z aplikacją uruchamianą na komputerze PC. Drugi procesor steruje licznikiem czasu oraz zajmuje się obliczeniami statystycznymi w czasie wykonywania próby pomiarowej. W artykule przedstawiono również opis projektu sprzętowego oraz problem komunikacji pomiędzy procesorami w systemie wieloprocesorowym.
Słowa kluczowe: układy cyfrowe, system wieloprocesorowy, układy SoC, FPGA, precyzyjny licznik czasu
Symbole UKD: 531.76

1. Wprowadzenie

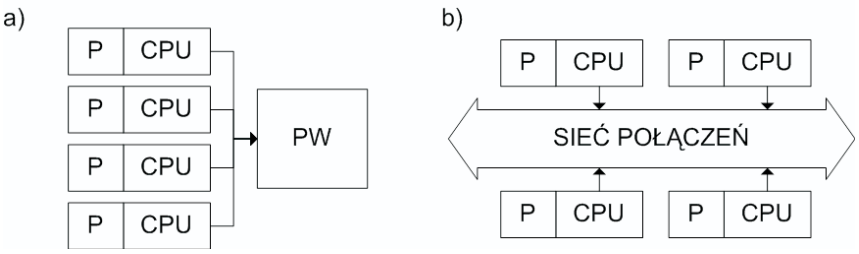
W ostatnich latach nastąpił dynamiczny rozwój systemów wieloprocesorowych. Związane jest to między innymi z barierą technologiczną, utrudniającą zwiększanie częstości taktowania rdzenia CPU, rosnącymi wymaganiami na moc obliczeniową. Ponadto do rozwoju systemów wieloprocesorowych przyczyniły się układy programowalne, które obecnie umożliwiają implementację w jednym układzie scalonym kompletnych systemów cyfrowych SoC (ang. *System-on-Chip*) z jednym lub wieloma procesorami. Procesory te są implementowane jako sprzętowe lub programowe (tzn. opisane w języku opisu sprzętu). Chociaż wydajność procesorów sprzętowych przeważnie jest większa od programowych [1], to jednak procesory programowe oferują znacznie większą elastyczność [2] i łatwość projektowania systemów

wieloprocesorowych [3, 4]. Zasadniczą zaletą systemów wieloprocesorowych jest możliwość równoległego wykonywania wielu zadań [5]. Jednak w takich systemach istotnym problemem staje się opracowanie oprogramowania. Należy z góry określić zadania dla każdego z procesorów i sposób komunikacji między nimi. Ponadto problemem staje się również korzystanie ze wspólnej pamięci.

W niniejszym artykule omówiono wykonany układ SoC, zawierający dwa procesory programowe Nios II i precyzyjny licznik czasu oraz przedstawiono projekt wykonanego oprogramowania. Opracowane aplikacje dla procesorów Nios II działają niezależnie, ale ze sobą współpracują. Jeden z procesorów zajmuje się sterowaniem licznikiem czasu i przetwarzaniem danych w czasie rzeczywistym, natomiast drugi realizuje komunikację z siecią Internet.

2. Sposoby komunikacji pomiędzy procesorami w systemach wieloprocesorowych

Zrównoleglenie operacji obliczeniowych wymaga podziału zadań pomiędzy rdzeniami i rozdzielenia danych, na których poszczególne rdzenie będą operowały. Komunikacja pomiędzy rdzeniami w systemach wieloprocesorowych może odbywać się dwojako. Wyróżniamy systemy ze współdzieloną pamięcią (SM — ang. *Shared Memory*), w których do komunikacji służy wspólny segment pamięci oraz architekturę z wymianą wiadomości (MP — ang. *Message Passing*), w których wymiana danych odbywa się bezpośrednio pomiędzy rdzeniami [6, 7]. Na rysunku 2.1 przedstawiono modele wymiany informacji pomiędzy rdzeniami procesorów (CPU).



Rys. 2.1. Modele komunikacyjne w systemach wieloprocesorowych

W modelu, w którym rdzenie dzielą wspólną przestrzeń adresową, do wymiany informacji służy segment pamięci współdzielonej (PW) — rys. 2.1a. Natomiast w modelu wymiany wiadomości rdzenie procesorów wyposażone są we własną lokalną pamięć (P) i komunikują się za pośrednictwem sieci połączeń (rys. 2.1b). Sieci połączeń systemów wieloprocesorowych mogą być statyczne (połączenie

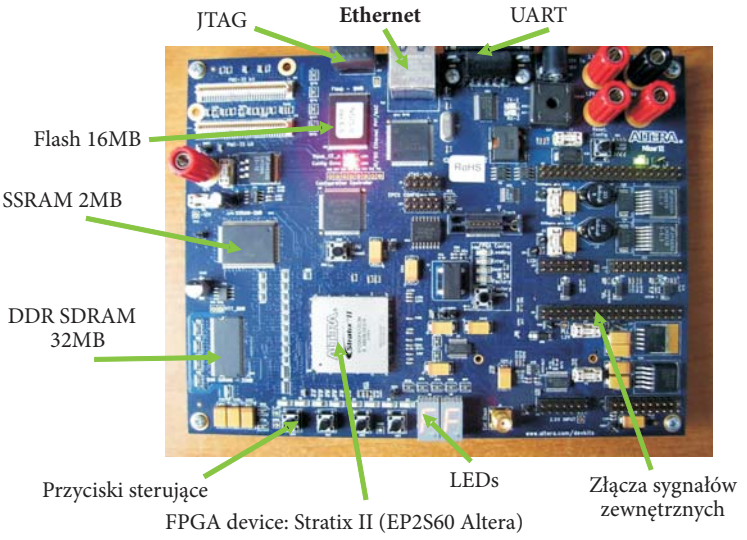
stałe) lub dynamiczne (połączenie zestawiane dla dwóch rdzeni w trakcie działania systemu). Połączenia dynamiczne realizowane są na zasadzie magistral lub też sieci przełączanych, np. typu crossbar [6, 8]. W połączeniach statycznych istnieją dowolne warianty konfiguracji połączeń w zależności od potrzeby połączeń konkretnych rdzeni — połączenia typu gwiazda lub każdy z każdym.

Rozwój procesorów programowych oraz możliwości układów FPGA spowodowały powstanie rozwiązań do projektowania systemów wieloprocesorowych w oparciu o gotowe moduły IP. Firma *Altera* ułatwia proces budowy systemów wieloprocesorowych przez udostępnienie dwóch rozwiązań sprzętowych. Pierwszym z nich jest układ semafora — mutex, który służy do zapewnienia wyłączności wykorzystania współdzielonego zasobu, takiego jak pamięć czy układ peryferyjny. Drugim modułem jest pamięć FIFO — mailbox, która służy do jednokierunkowego przesyłania 32-bitowych wiadomości pomiędzy rdzeniami. Rozwiązaniem firmy *Xilinix* jest dedykowane jednokierunkowe połączenie rdzeni procesorów typu punkt-punkt FSL (ang. *Fast Simple Link*). Rozwiązanie to umożliwia rozróżnienie poszczególnych rdzeni przez specjalny rejestr wersji procesora. Szersze omówienie połączenia FSL jak również możliwych wariantów połączenia rdzeni procesorów w układach programowalnych znajduje się w [8, 9, 10].

3. Projekt sprzętowy systemu dwuprocesorowego z licznikiem czasu

Omawiany system dwuprocesorowy przeznaczony jest do sterowania układem precyzyjnego konwertera czas–liczba, przetwarzania danych pomiarowych i wykonywania zdalnych pomiarów przez sieć Internet. Założeniem projektu jest również przeprowadzenie możliwie jak największej liczby obliczeń w układzie programowalnym. Jest to szczególnie istotne przy wykonywaniu serii pomiarowej, gdzie do oszacowania wyniku pomiaru należy posłużyć się odpowiednią statystyką. W najprostszym przypadku estymatorem dla próby jest średnia arytmetyczna otrzymanych pomiarów. Zastosowanie bardziej zaawansowanych estymatorów, które potrafią na przykład odrzucić część zakłóconych pomiarów, wymaga większych mocy obliczeniowych. Z tego powodu moc obliczeniową jednego procesora postanowiono wykorzystać tylko do wykonywania obliczeń statystycznych.

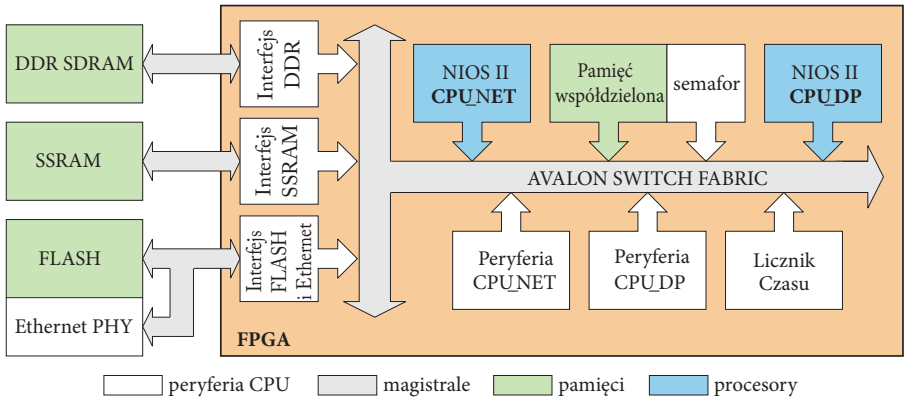
Głównymi elementami zaprojektowanego systemu są procesory programowe Nios II oraz precyzyjny licznik czasu. W pierwszym podrozdziale opisano system mikroprocesorowy, natomiast drugi podrozdział zawiera opis precyzyjnego licznika czasu. Prezentowany system został zaimplementowany w układzie Stratix II firmy *Altera* dostępnym na płytce rozwojowej *Nios II Developement Kit Stratix II Edition* (rys. 3.1).



Rys. 3.1. Widok płyty rozwojowej na której zaimplementowano system dwuprocesorowy z precyzyjnym licznikiem czasu

3.1. Projekt sprzętowy systemu mikroprocesorowego

W celu skrócenia czasu przetwarzania danych pomiarowych zdecydowano się na rozdzielenie funkcji pomiarowo-obliczeniowych od funkcji komunikacyjnych. W tym celu opracowano system dwuprocesorowy, w którym jeden procesor (CPU_NET) odpowiedzialny jest za komunikację systemu z zewnętrzną aplikacją sterującą, a drugi steruje procesem dokonywania pomiaru i przeprowadza obliczenia statystyczne (CPU_DP). Na rysunku 3.2 przedstawiono schemat zaprojektowanego systemu dwuprocesorowego.



Rys. 3.2. Schemta blokowy systemu dwuprocesorowego

Do obsługi konwertera wybrano procesory programowe Nios II, gdyż procesory te wykorzystują specyficzne właściwości układów FPGA firmy *Altera*. Pozwala to na uzyskanie większej wydajności niż procesorów programowych, które mogą być implementowane na wielu platformach. Firma *Altera* dostarcza również zintegrowane środowisko projektowe, które ułatwia proces projektowania wbudowanych systemów cyfrowych. Procesory Nios II charakteryzują się 32-bitową architekturą typu RISC o harwardzkiej architekturze pamięci. Z dostępnych trzech wersji rdzenia wybrano wersję standardową, charakteryzującą się korzystnym stosunkiem zajmowanych zasobów do wydajności.

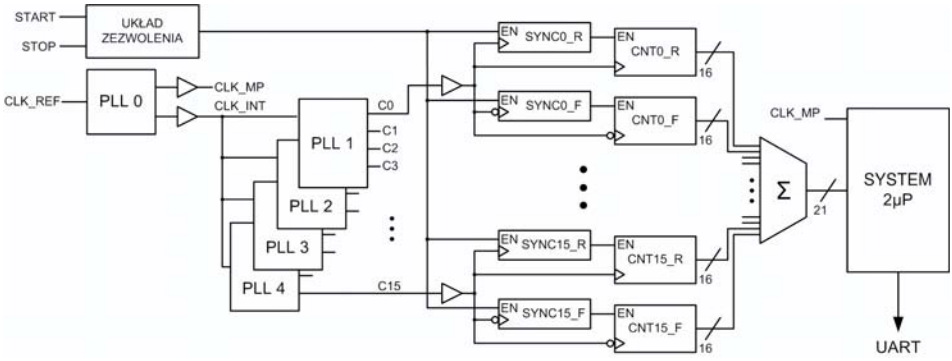
Wykorzystane z biblioteki standardowej procesora Nios II układy peryferyjne oraz interfejsy pamięci zostały połączone magistralą systemową Avalon Switch Fabric. Ze względu na wielkość dostępnej pamięci na płytce rozwojowej z układem Stratix II zdecydowano się na rozdzielenie pamięci programu oraz danych obu procesorów. Procesor komunikacyjny (CPU_NET) wykorzystywał pamięć DDR SDRAM o pojemności 32 MB, a procesor obliczeniowy (CPU_DP), ze względu na krótszy czas dostępu, wykorzystywał 2 MB pamięci SSRAM. Dodatkowo w dostępnej pamięci Flash o pojemności 16 MB przechowywane były bootloadery kodu dla obu procesorów. Pamięć Flash wykorzystywana jest również jako pamięć konfiguracji układu FPGA. Przy włączeniu zasilania następuje procedura konfiguracji układu programowalnego, po której bootloadery kodu obu procesorów przenoszą zapisany kod programu do odpowiedniej pamięci RAM.

3.2. Precyzyjny licznik czasu

Precyzyjny licznik czasu działa na podstawie metody licznikowej z wykorzystaniem zegara wielofazowego. Metoda licznikowa polega na zliczaniu okresów sygnału zegarowego o znanej częstotliwości w czasie trwania mierzonego odcinka czasu. Do zalet tej metody należy prostota implementacji układu licznika oraz stosunkowo krótki czas konwersji. Wykorzystanie pojedynczego układu licznika powoduje jednak, że otrzymujemy względnie niskie rozdzielczości. Na przykład dla zegara 200 MHz otrzymujemy rozdzielczość 5 ns. W celu podwyższenia rozdzielczości konwertera możliwe jest podwyższenie częstotliwości lub też zastosowanie sygnału wielofazowego.

Użyty w projekcie układ programowalny Stratix II EP2S60F672 ma wbudowane bloki funkcjonalne do kontroli oraz przetwarzania sygnału zegarowego — układy PLL (ang. *Phase Locked Loop*) [12]. Układy te mogą wytwarzać sygnały wewnątrz układu FPGA o częstotliwości od 1,5 MHz do 550 MHz, przesuwać fazę sygnałów z minimalnym krokiem 125 ps i z dokładnością ±15 ps.

Zaprojektowany konwerter czas–liczba wykorzystuje trzydzieści dwa liczniki zliczające okresy zegara szesnastofazowego. Schemat układu konwertera czas–liczba w układzie FPGA przedstawiono na rysunku 3.3.



Rys. 3.3. Schemat blokowy konwertera czas–liczba

Pierwsza pętla (PLL0) wytwarza sygnał zegarowy CLK_INT o częstotliwości 400 MHz dla konwertera oraz sygnał zegarowy dla dwuprocesorowego systemu przetwarzania danych CLK_MP. Sygnał zegarowy CLK_INT podawany jest następnie na 4 pętle PLL (PLL1, PLL2, PLL3, PLL4), które działają jako przesuwniki fazy. Zespół pętli PLL generuje 16 sygnałów zegarowych (C0..C15), przesuniętych względem siebie o 156,25 ps. Każdy z tych sygnałów podawany jest na dwa układy asynchronicznego szeregowego licznika 16-bitowego, z których jeden reaguje na zbocze narastające (CNTx_R), a drugi na zbocze opadające (CNTx_F) sygnału zegarowego. W celu zmniejszenia prawdopodobieństwa wystąpienia stanów niestabilnych w licznikach, sygnały zezwolenia na pomiar START oraz STOP synchronizowane są z kolejnymi fazami zegara na wejściu liczników (SYNCx_R — SYNCx_F). Po zakończeniu pomiaru wyniki z 32 liczników są sumowane i przekazywane do systemu dwuprocesorowego. Rozdzielczość tak zaprojektowanego układu wynosi 78,125 ps, a maksymalny zakres pomiarowy to 164 μs. Zakres ten może być łatwo poszerzony przez zwiększenie pojemności liczników. Licznik czasu został dołączony do magistrali systemowej Avalon Switch Fabric jako układ podrzędny — Avalon Slave.

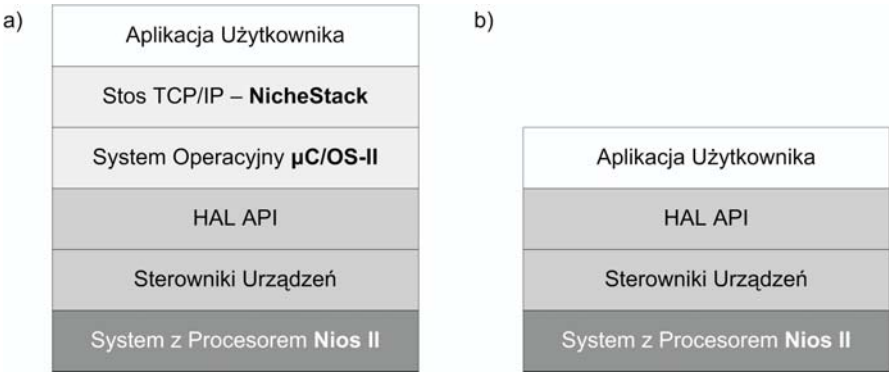
4. Oprogramowanie systemu dwuprocesorowego z licznikiem czasu

W systemach wieloprocesorowych istotnym problemem jest podział zadań pomiędzy procesory. Zadania te nie mogą być zbytnio „rozdrobione", bo to powoduje, że wypadkowa wydajność takiego systemu niewiele się zwiększa w porównaniu z systemem jednoprocesorowym. W omawianym systemie dwuprocesorowym do głównych zadań oprogramowania należą: sterowanie precyzyjnym licznikiem czasu, przetwarzanie danych uzyskanych z układu licznika — obliczanie wartości

statystycznych w różnych statystykach w czasie trwania pomiaru, obsługa układów peryferyjnych płytki bazowej oraz obsługa komunikacji TCP/IP z aplikacją sterującą. Aby nie przeciążać jednego procesora oraz zminimalizować czas martwy pomiaru, postanowiono przydzielić zadania pomiarowe do jednego procesora. Wszystkie inne zadania zostały przydzielone do procesora drugiego.

4.1. Oprogramowanie procesora komunikacyjnego

W celu oprogramowania interfejsu TCP/IP wybrano implementację stosu TCP/IP NicheStack firmy *InterNiche Technologies Inc*. [13]. Oprogramowanie to jest dostarczane wraz ze środowiskiem projektowym Nios II IDE dla procesorów programowych Nios II. Implementacja NicheStack dla procesorów Nios II wymaga do swojego działania uruchomienia systemu operacyjnego czasu rzeczywistego RTOS (ang. *Real Time Operating System*) µC/OS-II, gdyż odwołuje się on do specyficznych funkcji tego systemu operacyjnego [14]. Rozwiązanie to umożliwia obsługę połączenia TCP/IP przez programowanie gniazd (ang. *sockets*), czyli podobnie jak ma to miejsce w przypadku programowania w systemach operacyjnych dla komputerów PC np. Linux. Oprogramowanie dla procesorów Nios II zostało napisane uwzględniając tak zwany model warstwowy (rys. 4.1).



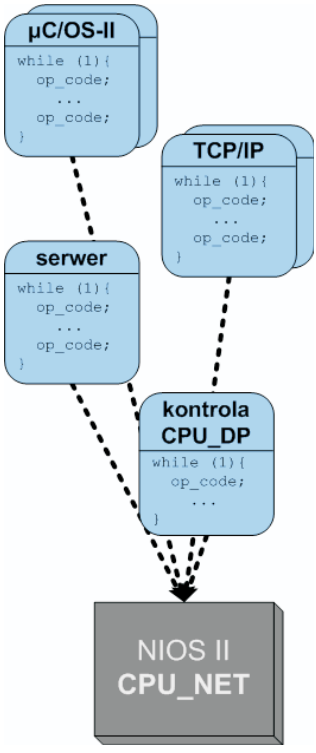
Rys. 4.1. Model warstwowy oprogramowania procesorów Nios II: a) dla procesora komunikacyjnego; b) dla procesora obliczeniowego

Na najniższym poziomie znajduje się zintegrowany system cyfrowy zbudowany na podstawie magistrali systemowej Avalon z procesorami programowymi Nios II. Dla każdego bloku IP, będącego układem peryferyjnym procesora, muszą zostać napisane sterowniki, które opisują interfejs dostępu do nich w systemie. Znajduje się tu z reguły opis w języku C w postaci makr rejestrów danego bloku. Aby ułatwić programiście dostęp do urządzeń zaprojektowana została warstwa abstrakcji sprzętu (HAL — ang. *Hardware Abstraction Layer*). Zawiera ona zbiór funkcji i makr,

które ukrywają przed programistą szczegóły sprzętowej implementacji dostępu do danego zasobu. System operacyjny µC/OS-II jest warstwą, która korzysta z odwołań do interfejsu programowego — API (ang. *Application Programming Interface*) sterowników z warstwy HAL. Najwyższymi warstwami w tym modelu są warstwy stosu protokołów TCP/IP oraz aplikacje użytkownika. Aplikacje użytkownika mogą odwoływać się do dowolnej niższej warstwy. Model warstwowy aplikacji zaprojektowanej tak jak moduł NicheStack i system operacyjny µC/OS-II przedstawiony został na rysunku 4.1a.

System RTOS µC/OS-II pozwala na projektowanie aplikacji wielowątkowych. Podstawowym zadaniem systemu operacyjnego jest zarządzanie zadaniami — wątkami oraz przydzielaniem czasu procesora dla poszczególnych zadań. Oferuje także takie usługi systemu operacyjnego jak komunikację pomiędzy wątkami oraz ich synchronizację. System operacyjny µC/OS-II zajmuje się również zarządzaniem pamięcią oraz obsługą czasu systemowego.

Na procesorze CPU_NET po procedurze startowej systemu uruchomionych jest 6 wątków. Wśród nich są wątki systemu µC/OS-II: wątek zbierający statystyki procentowego wykorzystania czasu procesora *OS_TaskStat()* oraz wątek bez-



Rys. 4.2. Wątki systemu µC/OS-II uruchomione na procesorze CPU_NET

czynności *OS_TaskIdle()*. Wątek bezczynności wywoływany jest wtedy, gdy inne zadanie systemu nie jest w stanie gotowości do wykonania, np. gdy oczekuje na dane z urządzenia wejścia/wyjścia tak, aby procesor cały czas pracował. Na rysunku 4.2 przedstawiono graficzną prezentację wątków uruchomionych na procesorze CPU_NET.

Następnymi są dwa wątki obsługi stosu protokołów TCP/IP NicheStack *tk_nettick()* oraz *tk_netmain()*. Wątek *tk_netmain()* jest głównym wątkiem obsługi stosu NicheStack. Jest on w stanie uśpienia do czasu, gdy nowy pakiet IP jest gotowy do przetworzenia. Pakiety odbierane są w funkcji obsługi przerwania, która przekazuje je do kolejki odbiorczej, a następnie wybudza wątek główny. Drugi wątek *tk_nettick()*, który jest okresowo wybudzany w celu obsługi liczników stosu TCP/IP. Pozostałe dwa to wątek realizujący funkcje serwera TCP/IP — *ServerTask()* oraz wątek komunikacji z procesorem CPU_DP i obsługi grupy pomiarów *TCMeasureMngTask()*. Wątek serwera odpowiedzialny jest za przyjmowanie lub odrzucanie połączeń przychodzących z sieci. W projekcie przewidziano komunikację z tylko jedną aplikacją sterującą, jednakże program może być z łatwością przepisany tak, aby umożliwić komunikację systemu z wieloma hostami jednocześnie.

Wątek *ServerTask()* odbiera od aplikacji sterującej odpowiednio sformatowane komunikaty. Następnie w zależności od otrzymanego polecenia zawartego w komunikacie wykonywana jest odpowiednia czynność — na przykład rozpoczęcie serii pomiarowej o zadanej liczności próby zostaje zlecone wątkowi *TCMeasureMngTask()*.

Komunikacja pomiędzy wątkami odbywa się poprzez kolejkę komunikatów oraz grupę flag. W kolejce komunikatów wysyłane są komendy sterujące, przetworzone przez wątek serwera do wątku *TCMeasureMngTask()*. Wątek *TCMeasureMngTask()* ustawia flagi informujące wątek serwera TCP/IP o stanie pomiaru. Jest on odpowiedzialny za kontrolę i sterowanie procesora obliczeniowego. Obsługuje komunikację z procesorem CPU_DP przez pamięć współdzieloną, przechowuje ustawienia ostatniego pomiaru na wypadek utraty połączenia TCP/IP z aplikacją sterującą.

4.2. Oprogramowanie procesora obliczeniowego

Oprogramowanie dla procesora obliczeniowego CPU_DP nie wykorzystuje do swego działania systemu operacyjnego. Jest to więc aplikacja jednowątkowa. W przypadku tego procesora istotnym parametrem jest czas wykonywania obliczeń oraz odstęp pomiędzy kolejnymi pomiarami. W przypadku aplikacji, która zakłada wykonywanie jednego zadania w danym momencie, użycie systemu operacyjnego wprowadziłoby dodatkowe opóźnienia ze względu na obsługę systemu operacyjnego. Rozdzielenie zadań na dwa procesory pozwoliło przydzielić całą moc obliczeniową procesora Nios II CPU_DP do obliczeń statystycznych, wykonywanych w trakcie realizacji próby pomiarowej oraz po jej zakończeniu.

Podstawowymi parametrami wyznaczanymi podczas próby pomiarowej jest wartość minimalna i maksymalna. Wykonywane są również działania pomocnicze do obliczenia wartości średniej. Po zakończeniu serii pomiarów system oblicza średnią, a następnie wartość odchylenia standardowego. Procesor CPU_DP podczas wykonywania serii pomiarów zapisuje wyniki pomiarów odcinków czasu oraz statystyk w pamięci współdzielonej. Nie występuje więc potrzeba kopiowania wyników pomiarów z pamięci RAM procesora do pamięci współdzielonej w celu wymiany danych. Pamięć współdzielona jest pamięcią wbudowaną w układzie FPGA, jest więc dużo szybsza niż pamięci zewnętrzne.

Jak już wspomniano wcześniej, procesor CPU_DP odpowiedzialny jest za kontrolę układu precyzyjnego konwertera czas–liczba. Moduł ten podłączony jest do magistrali systemowej Avalon. Jest on urządzeniem typu slave dla procesora CPU_DP. W celu kontroli tego układu napisano odpowiednie sterowniki dla procesora Nios II oraz API. Na listingu 4.1 przedstawiono zawartość pliku nagłówkowego „wat_avalon_timer_counter_regs.h", który opisuje dostęp do rejestrów konwertera czas–liczba.

List. 5.1. Plik sterownika opisujący rejestry konwertera czas–liczba

```
#ifndef __WAT_AVALON_TIMER_COUNTER_REGS_H__
#define __WAT_AVALON_TIMER_COUNTER_REGS_H__
#include <io.h>
/* Basic address, read and write macros. */
/* Registers 0,1,3 are reserved. Reading from this registers result
undefined value */
#define IOADDR_WAT_AVALON_TIMER_COUNTER_CTRL(base) __IO_CALC_AD-
DRESS_NATIVE(base, 2)
#define IOWR_WAT_AVALON_TIMER_COUNTER_CTRL(base, data) IOWR(base,
2, data)
#define IOADDR_WAT_AVALON_TIMER_COUNTER_RESULT(base) __IO_CALC_AD-
DRESS_NATIVE(base, 4)
#define IORD_WAT_AVALON_TIMER_COUNTER_RESULT(base) IORD(base, 4)
#defineIOADDR_WAT_AVALON_TIMER_COUNTER_STATUS(base) __IO_CALC_AD-
DRESS_NATIVE(base, 7)
#define IORD_WAT_AVALON_TIMER_COUNTER_STATUS(base) IORD(base, 7)
/* Masks */
#define WAT_AVALON_TIMER_COUNTER_CTRL_GO_MSK (0x1)
#define WAT_AVALON_TIMER_COUNTER_STATUS_BSY_MSK (0x1)
#define WAT_AVALON_TIMER_COUNTER_RESULT_MSK (0x001FFFFF)
#endif /* __WAT_AVALON_TIMER_COUNTER_REGS_H__ */
```

Plik ten zawiera definicję makr opisujących odczyt i zapis pod dany rejestr licznika makrami z pliku „io.h", definiującymi operacje zapisu i odczytu w systemie Avalons — odpowiednio IORD() i IOWR(). Makra te uwalniają programistę od potrzeby pamiętania przesunięć w adresowaniu konkretnego rejestru, gdyż przy użyciu tych funkcji należy podać jedynie adres bazowy licznika. Adres bazowy licznika przechowywany jest w pliku nagłówkowym „system.h", generowanym przez środowisko programistyczne Nios II IDE.

Z pliku opisującego rejestry układu konwertera korzystają zaprojektowane funkcje API konwertera opisane w pliku „timer_counter.c". Definicje funkcji oraz pomocnicze stałe zawarte są w skojarzonym pliku nagłówkowym „timer_counter.h".

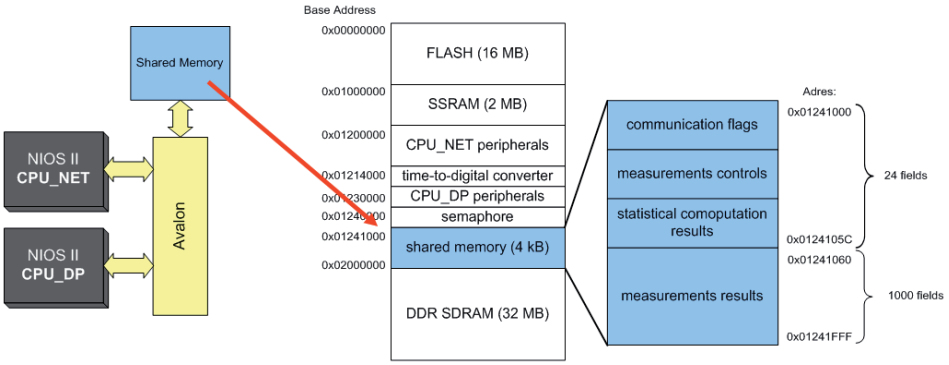
Funkcje API udostępniają programiście wykonywanie pomiarów odcinka czasu przez konwerter czas–liczba obecny w systemie. Podstawowymi funkcjami obsługi konwertera są funkcje włączenia pomiaru — *tc_start()* oraz funkcja odczytu wyniku pomiaru — *tc_get_result()*. Funkcja *tc_get_result()* zwraca zsumowaną zawartość liczników okresów, będącą jednocześnie wynikiem pomiaru w jednostkach rozdzielczości konwertera. Funkcjami pomocniczymi są funkcja sprawdzania statusu licznika — *tc_get_status()* oraz funkcja blokująca — *tc_while_busy()*, która służy do wstrzymywania programu do czasu zakończenia pomiaru odcinka czasu. Dodatkowa funkcja do wyżej opisanych służy do przeprowadzenia pełnego pomiaru, czyli uruchomienia modułu konwertera, odczekania na koniec pomiaru oraz odczytania i zwrócenia wyniku pomiaru. Wszystkie powyższe funkcje przyjmują jako parametr adres bazowy modułu konwertera czas–liczba w systemie Avalon w postaci 32-bitowej liczby bez znaku.

Funkcje API zostały napisane, aby zapewnić uniwersalność i niezależność od konkretnej implementacji systemu. Odwoływanie się do modułu konwertera czas–liczba przez adres bazowy umożliwia zmianę jego umiejscowienia w przestrzeni adresowej. Zmiany implementacji funkcji obsługi konwertera w pliku „timer_counter.c" w związku z powyższym nie wpływają również na zachowanie się programów korzystających z funkcji API.

4.3. Komunikacja pomiędzy procesorami

Opracowany system dwuprocesorowy został zaprojektowany jako system ze współdzieloną pamięcią — SM. Oba procesory systemu mają dostęp do pamięci RAM z zasobów układu FPGA o pojemności 4 kB (rys. 4.3).

Aby wyeliminować jednoczesny dostęp do pamięci i zapewnić tym samym spójność danych w niej przechowywanych z pamięcią, skojarzony został układ sprzętowego semafora — moduł IP mutex firmy *Altera*. Przed dostępem do pamięci współdzielonej procesor musi wpierw sprawdzić czy semafor jest ustawiony, czyli zajęty przez drugi procesor. Jeżeli jest on nie ustawiony czyli wolny, jest on automatycznie zajmowany w trakcie operacji typu „test&set" [15]. Eliminuje się więc tym



Rys. 4.3. Organizacja pamięci współdzielonej

możliwość zajęcia semafora przez drugi procesor. Jeżeli oba procesory respektują te zalecenia, to tylko jeden z nich będzie mógł w danym momencie dokonywać operacji na pamięci. Podejście takie skutecznie usuwa niebezpieczeństwo nadpisania informacji przez jeden z procesorów, używanej przez procesor drugi.

Pamięć współdzielona opisana jest w programie procesorów jako tablica typu całkowitego — int, czyli posiada 1024 pola. Pierwsze 24 pola zarezerwowane zostały jako pola do przesyłania flag sterujących, na przykład rozpoczęcia pomiaru, oraz do przechowywania wyników statystycznych dla serii pomiarów. Pozostałe 1000 pól przechowuje wyniki otrzymane bezpośrednio z każdego kolejnego pomiaru z serii. Zaprojektowany system może wykonywać serie pomiarów o maksymalnej liczności 1000 próbek.

Komunikacja pomiędzy procesorami odbywa się przez ustawianie odpowiednich flag w przestrzeni adresowej pamięci współdzielonej. Procesor, po ustawieniu semafora i uzyskaniu wyłączności do pamięci współdzielonej, odczytuje flagę skojarzoną z procesorem, z którym się komunikuje. Jeżeli flaga jest ustawiona, czyli wartość danej komórki pamięci jest większa od zera, sprawdzany jest kod komendy i jeżeli jest to znana mu komenda wywołuje odpowiednią funkcję obsługi — dla przykładu rozpoczyna serię pomiarową. Następnie po wykonaniu danej funkcji zeruje flagę w pamięci współdzielonej oraz ewentualnie ustawia flagę z odpowiednią komendą dla drugiego procesora. Na listingu 4.2. przedstawiono główną pętlę programu procesora CPU_DP, która implementuje opisany powyżej protokół komunikacji z procesorem CPU_NET.

List. 4.2. Główna pętla programu procesora cpu_dp z protokołem komunikacyjnym

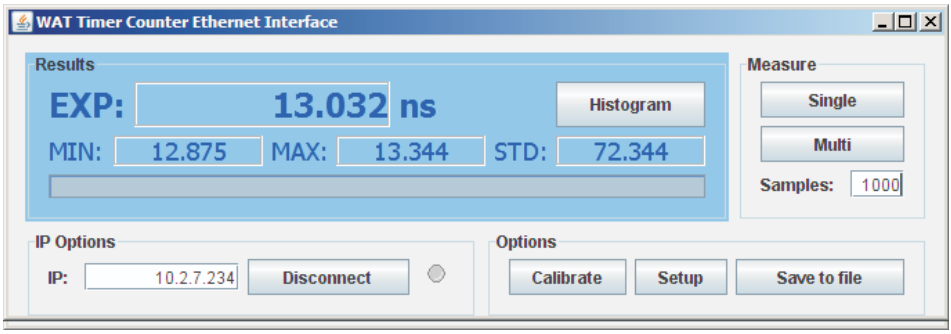
```
/* petla glowna */
while(1) {
 // sprobuj zdobyc semafor
 if(altera_avalon_mutex_trylock(shared_mem_mutex, 56) == 0) {
```

```
 /* semafor jest moj */
 // sprawdz flage cpu_net
 temp = shared_mem[WAT_MP_COMM_NET_FLAG];
 if (temp == WAT_MP_COMM_START) {
 /* flaga jest ustawiona na start pomiaru */
 // odczytaj liczbe probek do pobrania
 temp = shared_mem[WAT_MP_COMM_SAMPLES];
 // wyczyść zawartosc bufora probek
 for (i = 0; i < temp; i++) {
 shared_mem[WAT_MP_COMM_DATA_OFFSET + i] = 0;
 }
 // zwolnij semafor
 altera_avalon_mutex_unlock(shared_mem_mutex);
 // rozpocznij pomiar
 start_measure(temp, shared_mem);
 // Uwaga: semafor jest zablokowany w funkcji start_measure() !!
 // wyczysc liczbe probek i falge cpu_net
 shared_mem[WAT_MP_COMM_SAMPLES] = shared_mem[WAT_MP_COMM_NET_FLAG]
= 0x00;
 // ustaw flage cpu_dp na koniec pomiaru
 shared_mem[WAT_MP_COMM_DP_FLAG] = WAT_MP_COMM_DATA_READY;
 }
 // zwolnij semafor
 altera_avalon_mutex_unlock(shared_mem_mutex);
 }
 else { /* jezeli nie udalo sie zablokowac semafora to poczekaj
50us */
 usleep(50);
 }
} /* koniec petli while(1) */
```

4.4. Oprogramowanie komputerowe

Do sterowania systemem i przetwarzania danych z układu precyzyjnego konwertera czas–liczba poprzez sieć Internet opracowano aplikację pomiarową dla komputera PC w języku Java. Na rysunku 4.4 przedstawiono okno aplikacji w systemie operacyjnym Windows XP.

Wybór języka programistycznego podyktowany był przenośnością aplikacji pomiędzy różnymi systemami operacyjnymi bez potrzeby rekompilacji kodu programu. Java jest nowoczesnym obiektowym językiem programistycznym,

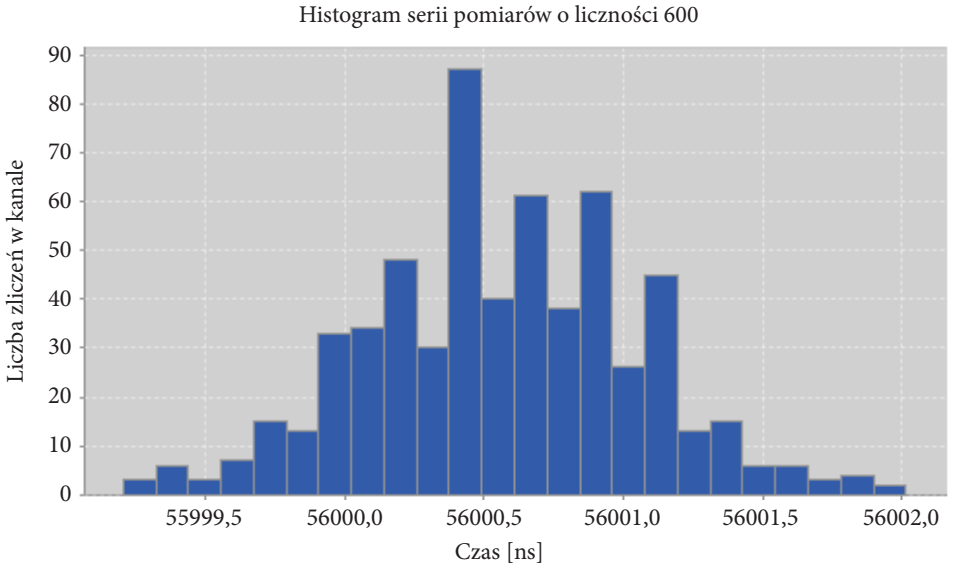


Rys. 4.4. Okno aplikacji pomiarowej w systemie operacyjnym Windows XP

który pozwala na uruchamianie raz skompilowanych programów w różnych środowiskach operacyjnych na tak zwanej maszynie wirtualnej. We wstępnej fazie projektowania napisano prostą aplikację z graficznym interfejsem użytkownika — GUI (ang. *Graphical User Interface*), która umożliwiła połączenie z systemem przez połączenie TCP/IP. Pozwoliło to na szybsze opracowanie i przetestowanie protokołu komunikacyjnego pomiędzy aplikacją docelową a systemem dwuprocesorowym.

W końcowej aplikacji, która powstała po zakończeniu prac nad protokołem komunikacyjnym, zastosowano prezentację wyników pomiaru w postaci histogramu. W tym celu posłużono się ogólnodostępnym zbiorem klas języka Java — JFreeChart. JFreeChart umożliwia, po krótkim zapoznaniu się z dokumentacją, używanie we własnych aplikacjach zaawansowanych wykresów. Przykładowy histogram otrzymany przy pomiarach odcinków czasu przedstawiono na rysunku 4.5.

Głównymi funkcjami opracowanej aplikacji jest sterowanie procesem pomiarowym za pośrednictwem sieci Internet. W tym celu możliwe jest podanie adresu IP płytki z systemem pomiarowym. Po nawiązaniu połączenia z serwerem uruchomionym na procesorze CPU_NET aplikacja przechodzi w stan gotowości, udostępniając użytkownikowi funkcje pomiarowe. Możliwe jest wykonywanie pojedynczych pomiarów oraz serii pomiarowej o liczności próby od 2 do 1000 próbek. W przypadku pojedynczego pomiaru aplikacja wyświetla otrzymany wynik. Natomiast dla serii pomiarów aplikacja wyświetla obliczony przez zdalny system pomiarowy wynik obliczeń statystycznych. W odpowiednich polach podawana jest wartość średnia z próby, odchylenie standardowe oraz wartości minimalne i maksymalne w serii. Odpowiedni przycisk uruchamia funkcję rysowania histogramu na podstawie próbek otrzymanych od systemu pomiarowego. Wyznaczaniem histogramu zajmują się odpowiednie funkcje z biblioteki JFreeChart.

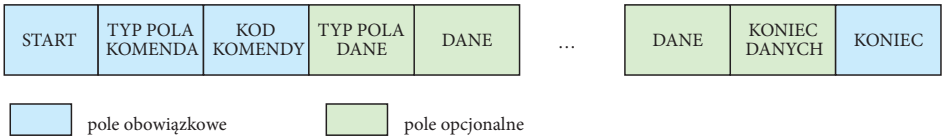


Rys. 4.5. Przykładowy histogram serii pomiarów

4.5. Protokół wymiany danych

Ponieważ jako warstwę transportową danych pomiarowych oraz sterujących zastosowano protokół TCP, protokół nie musi uwzględniać zabezpieczenia transmisji. Protokół TCP ma zaimplementowane mechanizmy zapewnienia wierności transmisji oraz kolejkowania przychodzących segmentów danych. Odczyt danych zarówno po stronie aplikacji dla komputera PC oraz systemu µC-OSII odbywa się na zasadzie czytania strumienia bajt po bajcie. Aby określić koniec przesyłanej grupy komunikatów, przyjęto, że znak końca linii ('\n' — kod ASCII 'Line Feed': 0x0A) będzie wymuszał odczytywanie bufora odbiorczego TCP/IP. Każdy komunikat zawiera informację o jednej komendzie. W dalszej części teksu terminem komunikatu określana będzie wiadomość ograniczona kolejnymi znakami końca komunikatu KONIEC, w której przesyłana będzie informacja o kodzie komendy do wykonania przez stronę odbiorczą wraz z opcjonalnymi dodatkowymi danymi. Na rysunku 4.6 zaprezentowano ogólną strukturę komunikatu przesyłanego pomiędzy aplikacją sterującą a systemem do precyzyjnego pomiaru czasu.

Pole KONIEC oznacza koniec komunikatu. Po tym polu może nastąpić pole końca linii lub pole START następnego komunikatu. Pole danych może przyjmować dowolną długość. Jednakże bufor odbiorczy jak i nadawczy w systemie mikroprocesorowym jest mocno ograniczony. Z tego powodu wysyłany komunikat lub grupa komunikatów ograniczonych znakiem końca linii powinny być mniejsze od 1500 bajtów.



Rys. 4.6. Struktura ramki przesyłanych komunikatów

5. Podsumowanie

Omawiany system dwuprocesorowy został w praktyce zaimplementowany, oprogramowany i przetestowany. W tabeli 1 przedstawiono wykorzystanie zasobów układu programowalnego przez zaprojektowany system.

Tabela 1

Wykorzystanie zasobów układu FPGA Stratix II

Zasób	System 2uP	System 2uP + Licznik Czasu	Dostępne w Stratix II
ALUT	5 120	6 364	48 352
Rejestry	3 392	4 228	48 352
Bloki DSP	16	16	288
Pamięć (bity)	126 464	126 464	2 544 192
Pętle PLL	1	5	6

System dwuprocesorowy stanowi około 80% zasobów zajmowanych przez cały projekt. Dużo większa zajętość zasobów pełnego projektu w porównaniu z miejscem zajmowanym przez układ samego licznika czasu zrekompensowana jest przez możliwość przetwarzania wyników po stronie układu FPGA. Pozwala to na przesyłanie do aplikacji sterującej przetworzonych wyników pomiarów. Jednakże całkowite wykorzystanie układu programowalnego Stratix II jest stosunkowo niewielkie — około 19% zasobów logicznych i mniej niż 5% wbudowanej pamięci RAM. Możliwe jest więc przeniesienie systemu do mniejszego układu programowalnego, co powinno zmniejszyć koszty rozwiązania. Można również wykorzystać więcej wbudowanej pamięci, aby wyeliminować jedną lub obydwie pamięci zewnętrzne.

Projekt zrealizowanego systemu dwuprocesorowego do sterowania precyzyjnym licznikiem czasu pokazuje, że z powodzeniem można budować wieloprocesorowe zaawansowane zintegrowane systemy cyfrowe, bazując na jednym układzie FPGA. Jednak istotnym problemem takich systemów jest korzystanie z pamięci, komunikacja pomiędzy procesorami oraz podział zadań na poszczególne procesory.

Korzystanie z jednej wspólnej pamięci dla danych i programu jest możliwe, ale wydłuża się wówczas czas dostępu do pamięci, co spowalnia wykonywanie się programu. Lepszym rozwiązaniem jest przydzielenie oddzielnej pamięci dla każdego procesora i korzystanie tylko z pamięci współdzielonej do wzajemnej wymiany informacji. Takie rozwiązanie wymaga stosowania semafora sprzętowego. Sama obsługa semafora (lub semaforów) zajmuje też pewien czas, dlatego wymiana informacji pomiędzy procesorami powinna występować znacznie rzadziej niż czas potrzebny na obsługę semafora. W przeciwnym przypadku stosowanie systemu wieloprocesorowego może nawet zmniejszyć moc obliczeniową całego systemu. Istotnym czynnikiem jest również podział zadań dla procesorów. Zadania te powinny być w sposób czytelny rozdzielone, np. tak jak w zaproponowanym systemie dwuprocesorowym, gdzie jeden procesor zajmuje się sterowaniem i przetwarzaniem danych z precyzyjnego licznika czasu, a drugi realizuje zdalną komunikację on-line przez sieć Internet.

Artykuł wpłynął do redakcji 14.07.2008 r. Zweryfikowaną wersję po recenzji otrzymano we wrześniu 2008 r.

LITERATURA

[1] T. Sondej, L. Zagoździński, R. Pełka, *Porównanie wydajności sprzętowego i programowego procesora w układzie FPGA Xilinx Virtex-4*, Pomiary Automatyka Kontrola, nr 7bis, 2006, 20-22.

[2] P. Yiannacouras, J. Rose, J. G. Steffan, *The microarchitecture of FPGA-Based Soft Processors*, Conference on Compilers, Architecture and Synthesis for Embedded Systems, 2005, 202-212.

[3] A. A. Jerraya, W. Wolf, *Multiprocessor Systems-on-Chips*, Morgan Kaufman, 2005.

[4] M. Hübner, K. Paulsson, J. Becker, *Parallel and Flexible Multiprocessor System-On-Chip for Adaptive Automotive Applications based on Xilinx MicroBlaze Soft-Cores*, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005, 149a-149a.

[5] P. James-Roxby, P. Schumaher, C. Ross, *A Single Program Multiple Data Parallel Processing Platform for FPGAs*, Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2004.

[6] A. Grama, A. Gupta, G. Karypis, V. Kumar, *An Introduction to Parallel Computing*, Second Edition, Addison Wesley, 2003.

[7] P. Gai, G. Lipari, M. Di Natale, M. Duranti, A. Ferrari, *Support for multiprocessor synchronization and resource sharing in Soft Core processors*, IEEE, 2005.

[8] P. Huerta, J. Castillo, J. I. Marinex, V. Lopez, *A MicroBlaze based multiprocessor SoC*, WSEAS Transactions on Circuits and Systems, May 2005, 423-430.

[9] H. C. Freitas, D. M. Colombo, F. L. Kastensmidt, P. O. A. Navaux, *Evaluating Network-on-Chip for Homogeneous Embedded Multiprocessors in FPGAs*, IEEE, 2007.

[10] L. P. Sun, El M. Aboulhamid, J. P. David, *Network on chip using a reconfigurable platform*, 2004.

[11] T. Sondej, R. Pełka, A. Poniecki, *Optimized data processing in precision laser rangefinder with embedded microcontroller*, Metrology and Measurement Systems, vol. 10, 3, 2003, 271-286.

[12] Altera, Stratix II Device Handbook, vol. 1 & 2; 2008.
[13] InterNiche Technologies Inc., *NicheStack Technical Reference*, 2006.
[14] J. J. Labrosse, µC/OS-II, *The Real-Time Kernel*, 2nd Edition, R&D Technical Books, 2002.
[15] W. Stallings, *Systemy operacyjne: struktura i zasady budowy*, PWN, Warszawa, 2006.

T. SONDEJ, M. GOŁASZEWSKI

Algorithms and methods for dual processor control system with precision time counter

Abstract. This paper presents issues of designing and implementing software for multiprocessor systems. Practical example consists of two softcore processors Nios II from Altera. Developed system is designed for control and data processing of precision timer counter with 80-ps resolution. The first processor runs as a server, providing communication and supervision of the system via the Internet. The second processor controls timer counter and performs statistical computation. Shared memory from FPGA resources is used to interchange data between processors.
Keywords: digital systems, multiprocessor system, system-on-chip, FPGA, precision time counter
Universal Decimal Classification: 531.76