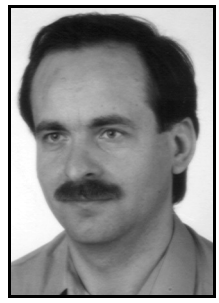
Stanisław DENIZIAK[1], **Robert TOMASZEWSKI**[2]
[1] POLITECHNIKA KRAKOWSKA
[2] POLITECHNIKA ŚWIĘTOKRZYSKA

Środowisko prototypowania systemów wbudowanych o architekturze NoC

Dr hab. inż. Stanisław DENIZIAK

Ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej, obronił pracę doktorską w 1994r, a habilitacyjną w 2006r. Jest profesorem nadzwyczajnym w Katedrze Informatyki Technicznej Politechniki Krakowskiej oraz profesorem wizytującym w Katedrze Informatyki Politechniki Świętokrzyskiej. Jego zainteresowania naukowe to metody szybkiego prototypowania systemów informatycznych, projektowanie systemów wbudowanych, testowanie i diagnostyka systemów cyfrowych.

e-mail: pedenizi@cyf-kr.edu.pl



Mgr inż. Robert TOMASZEWSKI

Ukończył studia na Wydziale Elektrotechniki, Automatyki i Informatyki Politechniki Świętokrzyskiej. Jest asystentem w Katedrze Informatyki WEAiI na Politechnice Świętokrzyskiej w Kielcach. Jego zainteresowania naukowe to sieci komputerowe, protokoły komunikacyjne, architektury Network-on-Chip.

e-mail: r.tomaszewski@tu.kielce.pl

Streszczenie

Praca prezentuje metodologię automatycznego odwzorowywania specyfikacji funkcjonalnej rozproszonego systemu wbudowanego, przedstawionej w języku SystemC, w zadaną architekturę typu NoC (ang. Network on Chip), w celu uzyskania prototypu implementowanego w FPGA. Protokół komunikacyjny sieci NoC oraz tabele rutingu generowane są na podstawie analizy komunikacji międzymodułowej. Procesy SystemC są konwertowane na programy w języku C++, a specyfikacja topologii NoC generowana jest w języku VHDL. Zalety przedstawionej metody obrazuje przykład wbudowanego serwera protokołu HTTP.

Słowa kluczowe: architektura NoC, modele SystemC, prototypowanie, synteza FPGA.

Prototyping environment for embedded systems of NoC architecture

Abstract

This work presents a methodology for mapping of a SystemC specification onto a given Network-on-Chip (NoC) architecture for the purpose of FPGA prototyping. A communication protocol and routing tables are generated automatically using inter-module communication analysis. For each processor in the target architecture, assigned SystemC processes are converted into C++ programs, where all communication method calls are replaced with sending/receiving messages to/from the network interface (NI) process. For each module implemented in hardware a VHDL code of the NI is generated. NIs convert transmitted data into/from network packets. The main advantage of our approach is the possibility to prototype and to evaluate many NoC architectures for a given system, without the necessity of modification of the source system specification. Presented embedded HTTP server example substantiates the benefits of the methodology.

Keywords: FPGA synthesis, NoC architectures, prototyping, SystemC models.

1. Wstęp

Systemy NoC są nowym rodzajem architektur wieloprocesorowych dla systemów typu MPSoC (Multi Processor System on Chip) [1]. Zastosowano w nich nowy styl komunikacji międzyprocesorowej – zamiast współdzielonej szyny informacje wymieniane są, podobnie jak w sieciach komputerowych, za pomocą przesyłanych ruterami pakietów. Pozwala to na znacznie lepszą skalowalność takich rozwiązań. To dopracowywane stale podejście wymaga wsparcia w postaci narzędzi do projektowania, modelowania, weryfikacji i prototypowania. Jedną z najdokładniejszych metod oceny parametrów systemu cyfrowego jest budowa jego prototypu FPGA z wykorzystaniem [2]. Złożoność obecnych układów FPGA umożliwia również implementowanie kompletnych systemów multiprocesorowych typu NoC [3].

Proces prototypowania składa się z dwóch zasadniczych etapów: na pierwszym tworzona jest specyfikacja funkcjonalna systemu (tzw. prototyp wirtualny), na drugim zaś – implementacja fizycznego prototypu. Uznanym narzędziem do modelowania na poziomie systemowym jest język SystemC [4]. Nadaje się on doskonale do opisu systemów wbudowanych i jest chętnie wykorzystywany do modelowania architektur NoC.

Istnieje kilka metodologii do projektowania architektur NoC. Różnią się one sposobem specyfikacji wymagań dla systemu docelowego oraz zakresem wyboru docelowych topologii NoC. Algorytm odwzorowania grafu zadań systemu w optymalną sieć NoC o topologii siatki (mesh) zaprezentowano w pracy [3]. Metodę automatycznej implementacji algorytmów DSP, również opisanych w formie grafów, w sieci NoC podano w referacie [5]. Rozwijane od dłuższego czasu systemy do projektowania architektur NoC takie jak Aethereal [6], NetChip (xPipes [7]) oraz Pirate [8] używają własnych formatów specyfikacji wymagań wejściowych, natomiast opis w SystemC generowany jest jako etap pośredni tuż przed syntezą. Środowisko projektowe Nostrum [9] dopuszcza jedynie topologie siatki (mesh) dla sieci NoC.

Żadna z wyżej wymienionych metod nie pozwala na automatyczne odwzorowanie modelu systemu opisanego w SystemC na dowolną sieć NoC. W pracy zaprezentowano metodologię szybkiego prototypowania systemów NoC. Na podstawie opisu funkcjonalnego podanego w SystemC oraz opisu docelowej topologii NoC automatycznie generowane są programy C++, tabele rutingu oraz syntezowalny kod VHDL systemu. Środowisko prototypowania zawiera biblioteki komponentów programowych i sprzętowych (routery, protokoły, itp.), a także narzędzia do analizy komunikacji międzymodułowej, generacji kodu oraz wyboru ścieżek przesyłania pakietów między procesorami. Celem pracy jest zminimalizowanie wysiłku projektanta szukającego najlepszej architektury NoC dla danej aplikacji. Mając do dyspozycji jeden opis systemu może on przetestować różne konfiguracje NoC, wybierając najlepsze. Środowisko ma charakter otwarty, co pozwala na jego łatwe rozszerzanie i uzupełnianie o nowe komponenty, np. kolejne konstrukcje routerów, dodatkowe strategie rutingu, itp.

2. Metodologia szybkiego prototypowania

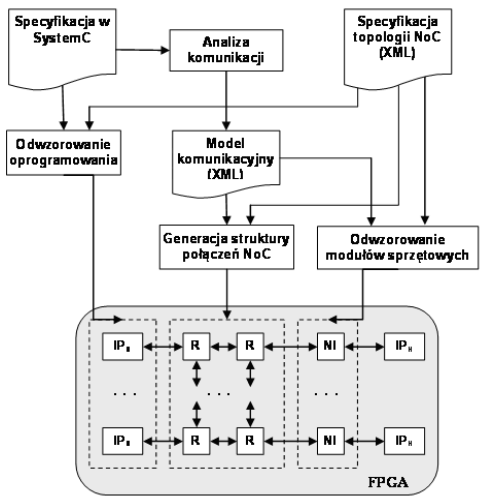
Specyfikacja systemu w SystemC ma strukturę modułów komunikujących się przy pomocy kanałów. Każdy moduł musi zawierać przynajmniej jeden proces aktywowany listą wrażliwości. W proponowanym podejściu najistotniejsze są kanały komunikacyjne, które implementują co najmniej jeden interfejs. W niniejszej pracy rozpatrzono następujące typy kanałów predefiniowanych: *sc_signal*, *sc_buffer* i *sc_fifo*. Najprostszy kanał – *sc_signal* – realizuje transmisje bez blokowania procesu i bez buforowania. Każda zmiana sygnału w kanale generuje zdarzenie. Kanał ten może łączyć co najwyżej jeden port wyjściowy (jedno- lub dwukierunkowy) z dowolną liczbą wejściowych. Kanał

sc_buffer różni się od *sc_signal* tylko tym, że generuje zdarzenie po każdej operacji zapisu (nawet nie zmieniającej stanu sygnału). Kanał *sc_fifo* umożliwia buforowane transmisje z/bez blokowania i łączy jedynie dwa porty. Operacje nieblokujące nie zmieniają stanu bufora FIFO, jeśli odczyt/zapis jest niemożliwy w momencie wywołania operacji.

Odwzorowanie opisu systemu w SystemC na prototyp FPGA dla zadanej topologii NoC przebiega następująco:

- specyfikacja SystemC jest analizowana pod kątem komunikacji międzymodułowej – w rezultacie powstaje model komunikacyjny zapisany w formacie XML,
- opisy funkcjonalne modułów przypisanych do procesorów (IP_S) konwertowane są na C++,
- dla modułów realizowanych sprzętowo (IP_H) generowane są interfejsy sieciowe (NI),
- kanały komunikacyjne SystemC odwzorowywane są w konfigurację NoC.

W wyniku powyższych kroków powstaje specyfikacja RTL VHDL całego systemu. Do syntezy wykorzystano układy FPGA firmy Altera z procesorami Nios II, system Quartus II oraz Nios II IDE [10]. Na rys.1 przedstawiono schemat procesu prototypowania. Opis docelowej topologii NoC, podany w języku XML, może być tworzony ręcznie lub automatycznie.

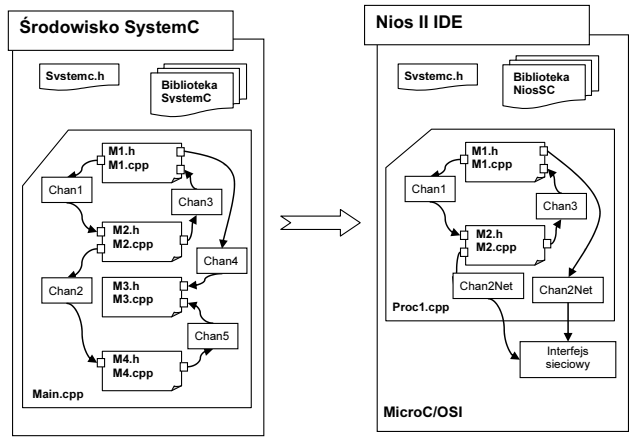


Rys. 1. Schemat metodologii prototypowania systemów NoC
Fig. 1. Rapid prototyping flow

2.1. Odwzorowanie modułów SystemC w programy

Oprogramowanie dla modułów IP_S jest generowane poprzez zamianę jądra symulacyjnego SystemC na wywołania systemu operacyjnego czasu rzeczywistego (RTOS) [11]. Kanały SystemC są zamieniane na odpowiadające im kanały (signal2ni, buffer2ni, fifo2ni) realizujące transmisje pomiędzy procesami lub modułami implementującymi NI. Adresy (ID) modułów oraz portów są przechowywane w procesie/module NI, który pośredniczy w komunikacji między procesorem a portem lokalnym rutera za pomocą niskopoziomowych operacji we/wy obsługiwanych przez RTOS.

Rys. 2 pokazuje metodę odwzorowania SystemC w programy. Moduły M1 i M2 przypisano do realizacji w procesorze Nios II, zatem ich specyfikacja w SystemC jest bez modyfikacji kopiowana do środowiska programowania Nios. Następnie generowana jest procedura *main()* uruchamiająca procesy i komunikację modułu. Do tego etapu wykorzystano w pełni równoległy MicroC/OSII RTOS.



Rys. 2. Generowanie programów wbudowanych
Fig. 2. Embedded software generation

2.2. Sprzętowa realizacja modułów SystemC

Zakłada się, że moduły sprzętowe (IP_H) są dostępne w postaci komponentów IP (ang. Intelectual Property core). Tworzony jest zatem jedynie kod VHDL umieszczający te moduły w projekcie. Każdy IP_H komunikuje się z innymi poprzez dedykowany proces/moduł NI oraz za pośrednictwem ruterów. Funkcjonalność NI może być różna w zależności od rodzaju portów modułu (dokładniejszy opis NI zawarto dalej), co skutkuje różnicami w wygenerowanym kodzie VHDL. Podobnie jak w ruterach LiPaR [12] transmisje są uzgadniane za pomocą sygnałów *Req*/*Ack*. Niestandardowe IP_H wymagają zastosowania adaptera dostosowującego interfejs do naszego NI. Jeżeli atrybuty *size* i *width* (przykłady plików *system*_**sc.xml** i *system*_**noc.xml** podano dalej) są różne to pojedyncza transmisja między IP a NI ma rozmiar *width* i składa się z *((size + (width - 1))* **div** *width)* bloków. W przeciwnym wypadku wykonywana jest pojedyncza transmisja o rozmiarze *size*.

2.3. Odwzorowanie kanałów w sieć komunikacyjną

Wykorzystywany w rozpatrywanym przykładzie ruter to LiPaR [12] ze zmodyfikowaną logiką trasowania – zamiast protokołu XY, odpowiedniego jedynie dla topologii regularnych, wybrano bardziej uniwersalny protokół oparty na najkrótszej ścieżce. Lokalny wybór wyjścia rutera do dalszego przesłania pakietu odbywa się na podstawie informacji zapisanych w pamięci danego rutera (w tzw. logice trasującej). Trasy są wyznaczane na bazie algorytmu przeszukiwania grafu wszerz (BFS [13]), po uprzednim odwzorowaniu docelowej, żądanej topologii sieci NoC na graf.

Komponent NI odpowiada za formowanie pakietów oraz ich wysyłanie/odbieranie (write/read). Do jego zadań należy również translacja adresów z formatu lokalnego na globalny. Format lokalny, używany na łączu między IP a NI, definiuje ID portu (opisany w pliku *system*_**sc.xml,** poprzedzony przez ID przynależnego modułu SystemC) – globalny wyznacza ID modułu IP. Translacja odbywa się na podstawie tabel zbudowanych na bazie informacji zawartych w plikach *system*_**sc.xml** i *system*_**noc.xml**. Każdy NI używa dwóch tabel o różnej zawartości: jednej do wysyłania i jednej do odbioru pakietów. Schemat tabel przedstawiono na rys. 3.

Dest IP	Local Addr	Dest Addr	Multi

Rys. 3. Schemat tabeli translacyjnej
Fig. 3. Translational table scheme

Pola *Local Addr, Dest Addr* i *Length of Data* tworzą nagłówek warstwy transportowej, traktowany razem z danymi jako ładunek dla pakietu warstwy sieciowej. Pole *Multi* zawiera adresy kolejnych pozycji w tabeli dla transmisji multicastowych (w SystemC jest to sytuacja, gdy jeden port OUT jest podłączony do kilku IN). Wartość 0 oznacza transmisję unicast. Każdy port lokalny w NI posiada swój adres oraz bufor mogący pomieścić pakiet o maksymalnym rozmiarze (atrybut *size* dla znacznika *port* w *system*_**sc.xml**).

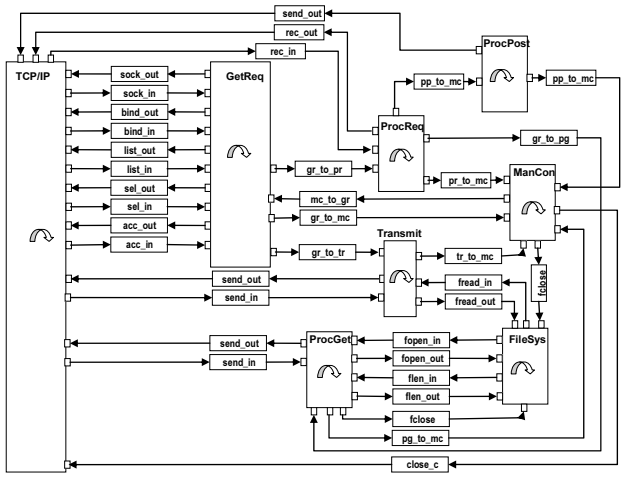
Zapis do portu oznacza wysłanie pakietu z nagłówkiem opisanym wcześniej do bufora z odpowiednim adresem, odczyt – odebranie z bufora. NI nadawcy, mając dany lokalny adres przeznaczenia, ustala adres globalny zdalnego IP i formuje numerowane ramki przesyłane następnie do ruterów. NI odbiorcy składa ramki po kolei, wg numerów, dekoduje nagłówek gotowego pakietu i szuka w swej tablicy translacyjnej adresu interfejsu między sobą a swoim modułem IP dla danego adresu lokalnego. Jeśli nagłówek zawierał znacznik transmisji z blokowaniem, odbiorca odsyła do nadawcy mały pakiet potwierdzający odczyt wiadomości. Jeśli NI otrzyma żądanie blokowanego odczytu (zapisu) a zaadresowany bufor FIFO jest pusty (pełny), wysyła do lokalnego procesu sygnał wymuszający czekanie.

Szczegóły odwzorowania kanałów SystemC:

- *sc_signal* – lokalny bufor NI jednoelementowy, operacja zapisu wymusza transmisję tylko wtedy, gdy poprzednie dane w buforze są różne od aktualnie zapisywanych,
- *sc_buffer* – jak wyżej, ale zapis zawsze wymusza transmisję danych,
- *sc_fifo* – bufory NI nadawcy i odbiorcy mają tyle elementów ile podano w specyfikacji (atrybut *buffer* znacznika *port* w *system*_**sc.xml**); dane istnieją w dwóch kopiach (u nadawcy i u odbiorcy), co ujednolica operacje blokujące i nieblokujące (z uwzględnieniem wcześniejszych uwag); tylko operacja zapisu generuje transmisję

3. Przykład

Dla zilustrowania powyższej metodologii został opisany w SystemC model wbudowanego serwera WWW, odwzorowany następnie w architekturę NoC. Serwer obsługuje polecenia GET i POST protokołu HTTP. Strukturę serwera przedstawia rys. 4. Tworzą go moduły *GetReq*, *ProcReq*, *ProcGet*, *ProcPost*, *Transmit* i *ManConn* oraz standardowe *TCP/IP* i *FileSys*. Wszystkie transmisje realizowane są jednoelementowymi kanałami *sc_fifo* (z wyjątkiem 6-elementowych kanałów *gr_to_pr*, *gr_to_tr* i *gr_to_mc*).



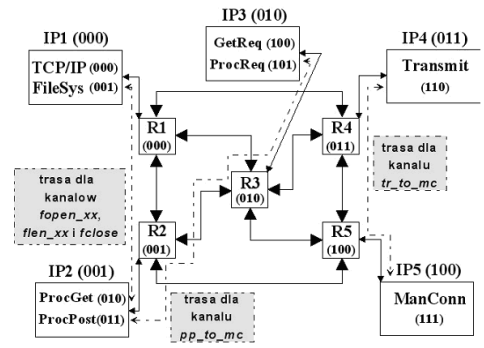
Rys. 4. Architektura serwera opisana w SystemC
Fig. 4. SystemC architecture scheme of the WWW server

Na podstawie opisu SystemC otrzymano plik *HTTP_sc.xml* częściowo pokazany na rys. 5. Docelową konfigurację NoC wraz z opisem w pliku *HTTP_noc.xml* ilustruje rys. 6. Przerywaną linią na rys. 6a zaznaczono najkrótsze ścieżki pomiędzy wybranymi modułami – rezultat algorytmu opartego na BFS. Porty ruterów zakodowano następująco: lokalny – 000, North – 001, East – 010, South – 011 i West – 100.

```
<arch id="HTTP_SC">
 <module id="010"> <!-- moduł ProcGet -->
  <process id="0">0</process>
  <!-- porty połączone kanałami fopen_xx -->
  <port id="0000" type="sc_fifo_in"
          process_conn="0" size="32">0</port>
  <port id="0001" type="sc_fifo_out"
          process_conn="0" size="2072">1</port>
  ...
 </module>
 <module id="001"> <!-- moduł FileSys -->
  <process id="0">0</process>
  <!-- porty połączone kanałami fopen_xx -->
  <port id="0000" type="sc_fifo_out"
          process_conn="0" size="32">0</port>
  <port id="0001" type="sc_fifo_in"
          process_conn="0" size="2072">1</port>
  ...
 </module>
 <!-- kanał fopen_in -->
 <channel id="000000" type="sc_fifo" buffer="1"
          conn="001_0000 010_0000">0</channel>
 <!-- kanał fopen_out -->
 <channel id="000001" type="sc_fifo" buffer="1"
          conn="001_0001 010_0001">1</channel>
</arch>
```

Rys. 5. Część pliku HTTP_sc.xml
Fig. 5. Part of HTTP_sc.xml file

a)



b)

```
<noc id="HTTP_NoC">
 <router id="000" l_port="000" N_port="011"
          E_port="010" S_port="001"
          type="lipar">0</router>
 <router id="001" l_port="001" N_port="000"
          E_port="010" S_port="100"
          type="lipar">1</router>
 <router id="010" l_port="010" N_port="000"
          E_port="011" S_port="100" W_port="001"
          type="lipar">2</router>
 <router id="011" l_port="011" N_port="000"
          S_port="100" W_port="010"
          type="lipar">3</router>
 <router id="100" l_port="100" N_port="011"
          S_port="001" E_port="010"
          type="lipar">4</router>
 <IP id="000" impl="SW" sc_mod="000 001">0</IP>
 <IP id="001" impl="SW" sc_mod="010 011">1</IP>
 <IP id="010" impl="SW" sc_mod="100 101">2</IP>
 <IP id="011" impl="SW" sc_mod="110">3</IP>
 <IP id="100" impl="HW" sc_mod="111">4
   <port id="cin0" width="8" />
   <port id="cin1" width="8" />
   <port id="cin2" width="8" />
   <port id="cin3" width="8" />
   <port id="cin4" width="8" />
   <port id="cout" width="8" />
   <port id="fclose" width="32" />
   <port id="ccloce" width="16" />
 </IP>
</noc>
```

Rys. 6. Schemat docelowej architektury NoC(a) i odpowiadający jej plik HTTP_noc.xml (b)
Fig. 6. Scheme of target NoC architecture(a) and corresponding HTTP_noc.xml (b)

4. Podsumowanie

Większość prac dotyczących systemów NoC skupia się na pojedynczych aspektach modelowania i optymalizacji. Zaprezentowana metodologia łączy dotychczasowe osiągnięcia w tej dziedzinie pozwalając projektantowi łatwo uzyskać działający prototyp FPGA i umożliwiając tym samym znalezienie najlepszej dla danego systemu architektury NoC na podstawie tej samej specyfikacji w SystemC. Mimo złożoności całego procesu pokazano, że jego automatyzacja jest możliwa. Główny nacisk został położony na aspekty komunikacyjne, najważniejsze z punktu widzenia rozwiązań NoC. Dalszy rozwój środowiska to rozbudowa różnych aspektów komunikacji, w tym wbudowanie innych strategii rutingu.

5. Literatura

[1] L.Benini, G. De Michelli, "Networks on Chips: A New SOC Paradigm", IEEE Computer, pp.70-78, 2002.

[2] D. Kissler, A. Kupriyanov, F. Hannig, D. Koch, J. Teich, "A Generic Framework for Rapid Prototyping of System-on-Chip Designs", Proc. of the CDES, pp.189-195, 2006.

[3] B. Sethuraman, R. Vemuri, "optiMap: A Tool for Automated Generation of NoC Architectures using Multi-Port Routers for FPGAs", Proc. of the DATE, pp.947-952, 2006.

[4] IEEE Standard SystemC Language Ref. Manual, IEEE, New York, 2006.

[5] X.Wu, T.Ragheb, A.Aziz, Y.Massoud, "Implementing DSP Algorithms with On-Chip Networks", Proc. of the NOCS, pp.307-316, 2007.

[6] K.Goossens et al., "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification", Proc. of the DATE, pp.1182-1187, 2005.

[7] A.Jalabert, S.Murali, L.Benini, G. de Michelli, "XpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip", Proc. of the DATE, pp.884-889, 2004.

[8] G. Palermo and C. Silvano, "PIRATE: A Framework for Power/Performance Exploration of Network-On-Chip Architectures", Lecture Notes in Computer Science, vol. 3254, pp.521-531, 2004.

[9] Z.Lu, I.Sander, A.Jantsch, "Refinement of a Perfectly Synchronous Communication Model onto Nostrum NoC Best-effort Communication", Proc. of the Forum on Design Languages, 2005.

[10] www.altera.com

[11] F.Herrera, H.Posadas, P.Sanches, E.Villar, "Systematic Embedded Software Generation form SystemC", Proc. of the DATE, pp.142-147, 2003.

[12] B.Sethuraman et al. "LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip", 15th Great Lakes Symposium on VLSI (GLSVLSI'05), pp.452-457, 2005.

[13] T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein, "Introduction to algorithms", The MIT Press, 2001.

Artykuł recenzowany

INFORMACJE

VII Konferencja Systemy Pomiarowe w Badaniach Naukowych i w Przemyśle SP'08

W dniach 18 – 20. czerwca 2008 r. odbyła się w Łagowie VII Konferencja Naukowa „Systemy Pomiarowe w Badaniach Naukowych i w Przemyśle SP'08". Konferencję zorganizował Instytut Metrologii Elektrycznej Uniwersytetu Zielonogórskiego. Podstawowym celem konferencji było zaprezentowanie wyników prac wykonywanych w ramach projektów badawczych, celowych oraz badań własnych realizowanych w ośrodkach akademickich i jednostkach badawczych, a dotyczących szeroko pojętej tematyki systemów pomiarowych. Tematyka VII konferencji SP obejmowała zagadnienia teorii, konstrukcji i badania komputerowych systemów pomiarowych, w tym zwłaszcza systemów z bezprzewodowym przesyłem informacji, zastosowań cyfrowego przetwarzania sygnałów w konstrukcji narzędzi pomiarowych, analizę metrologiczną przetworników i systemów pomiarowych.

Tradycją konferencji jest prezentacja podczas sesji otwarcia aktualnej i ważnej dla środowiska metrologów tematyki przez zaproszonych specjalistów. W tym roku referaty koncentrowały się na zastosowaniach zaawansowanych metod przetwarzania sygnałów w medycynie. Prof. dr hab. inż. Tomasz Zieliński z Akademii Górniczo-Hutniczej w Krakowie przedstawił referat pt.: „Segmentacja i dopasowywanie cyfrowych obrazów medycznych: przetwarzanie nagrań wideo-endoskopowych strun głosowych oraz danych tomograficznych zmian rakowych", natomiast prof. nzw. dr. hab. inż. Krzysztof Zaremba z Politechniki Warszawskiej wygłosił referat na temat: „Czy można zmierzyć myśli, czyli podstawy funkcjonalnego rezonansu magnetycznego". Obydwa referaty wywołały ożywioną i ciekawą dyskusję wśród uczestników konferencji.

25 prac nadesłanych przez autorów przedstawiono podczas sześciu sesji grupujących następujące zagadnienia: przetwarzanie sygnałów, bezprzewodowe systemy pomiarowe, przyrządy i układy pomiarowe, pomiary wielkości elektrycznych. Artykuły opracowane na podstawie referatów prezentowanych na konferencji są wydrukowane m.in. w czasopiśmie PAK.

Fot. 1. Uczestnicy konferencji Systemy Pomiarowe w Badaniach Naukowych i w Przemyśle SP'08

Organizatorzy mają nadzieję, że miejsce obrad, wspaniały klimat i malowniczy krajobraz okolic Łagowa pozostawi miłe wspomnienia i zachęci do udziału w kolejnej VIII Konferencji organizowanej w 2010 r.

Przewodniczący Komitetu Organizacyjnego Konferencji
dr hab. inż. Ryszard Rybski