II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

BEZAWARYJNE MECHANIZMY WSPOMAGANIA WSPÓŁCZESNYCH ROZWIĄZAŃ SPRZĘTOWYCH

Marek Sałamaj, Piotr Bubacz

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: M.Salamaj@iizp.uz.zgora.pl, P.Bubacz@iie.uz.zgora.pl

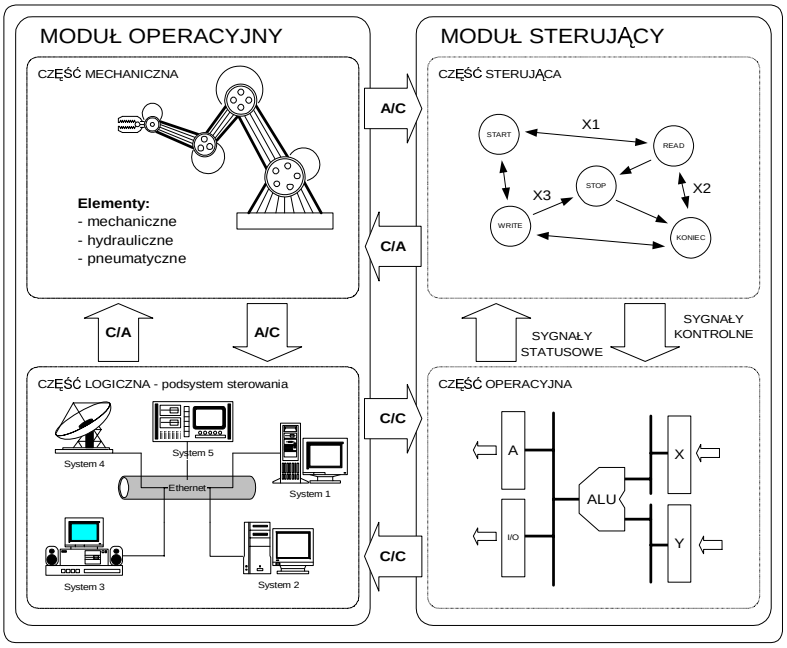
STRESZCZENIE

W referacie przedstawiono przegląd metod projektowania specjalistycznej aparatury sterującej, które rozwijano na przełomie ostatnich lat. Jest to spojrzenie na architekturę współczesnych rozwiązań sprzętowych stosowanych obecnie niemalże w każdej dziedzinie przemysłu. W artykule, przedstawiono również modułową budowę tych systemów, wraz z zależnościami pomiędzy nimi oraz właściwościami, które mają w tym przypadku istotny wpływ na bezpieczeństwo i bezawaryjność ich działania. Zwrócono również uwagę na aktualny stan rozwoju rozwiązań technicznych oraz zaproponowano nowe metody zwiększające ich precyzję, bezpieczeństwo, a przede wszystkim bezawaryjność ich działania w realizowanych systemach sterowania o „podwyższonym ryzyku działania".

1. WSPÓŁCZESNE ROZWIĄZANIA SPRZĘTOWE

W urządzeniach z dziedziny informatyki, elektroniki zauważyć można bardzo szybki wzrost możliwości projektowanego sprzętu. Jest on spowodowany rozwojem nowych technologii i narzędzi wspomagających projektowanie CAD (ang. *Computer Aimed Design*), które uzależnione jest od wzrostu zapotrzebowania na tego typu rozwiązania sprzętowe. W rozwiązaniach tych można wyszczególnić zaawansowany podział układu na mniejsze specjalizowane moduły: moduł sterujący i moduł operacyjny (rys. 1).

Tak przedstawiony, wewnętrzny podział projektowanego urządzenia spowodował, że każdy z modułów rozwijany był niezależnie. Doprowadziło to do tego, że każdy z modułów realizuje odmienne zadania w mikrosystemie. Elementy modułu operacyjnego, rys. 1, czynnie wpływają na wykonywaną pracę, natomiast elementy modułu sterującego, w zależności od pełnionej funkcji, określone zostały mianem „inteligentnej" jednostki sterującej. Jednostka ta wykonuje program przechowywany w pamięci, na podstawie którego realizowana jest synchronizacja i komunikacja w układzie.



Rys. 1. Struktura współczesnych rozwiązań sprzętowych

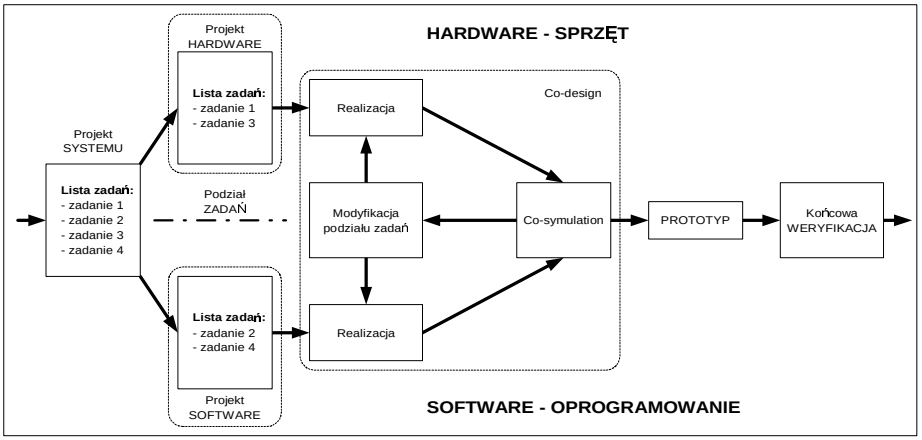
Jak można zauważyć, dynamiczny rozwój obu modułów, na przełomie ostatnich kilkudziesięciu lat sprawił, że zastosowane w nich elementy są efektywniejsze, bardziej złożone od początkowych rozwiązań. Dzięki temu, obecne moduły operacyjne realizowane są z dużo lepszych materiałów, są one bardziej wytrzymałe, lżejsze, odporniejsze na różnego rodzaju warunki zewnętrzne, a przede wszystkim umożliwiające wykonanie zadań z dużo większą precyzją.

Natomiast, w całkowicie innym kierunku rozwijany był moduł sterujący. W jednostce tej zwrócono szczególną uwagę na opracowanie coraz szybszych, bardziej pojemniejszych układów cyfrowych ASIC (ang. *Application Specific Integrated Circuits*), począwszy od małej, poprzez średnią, wielką, a kończąc na bardzo wielkiej skali integracji. W takim celu stworzono bardzo wiele użytecznych narzędzi CAD stosowanych do syntezy, implementacji, wykorzystywanych w modelowaniu, symulacjach oraz w analizach algorytmów. Algorytmów, których w owym czasie powstawało bardzo wiele, których przeznaczeniem było zwiększenie wydajności, szybkości realizowanych rozwiązań sprzętowych, a tym samym rozwiązań zwiększających moc obliczeniową.

W rezultacie wyżej opisanych działań opracowano efektywne metody i techniki realizacji sprzętu, w których nacisk kładziono głównie na ich bezawaryjność, dokładność, a przede wszystkim bezpieczeństwo działania.

2. BEZPIECZEŃSTWO

Rozpatrywany w poprzednim rozdziale podział (pod względem bezpieczeństwa) sprawia, że każdy z modułów, można interpretować niezależnie. Wynika to z faktu, że bezpieczeństwo modułu operacyjnego jest związane z zastosowaniem właściwych materiałów do jego realizacji. Materiałów, które powinny być odporne na czynniki zewnętrzne, jak nacisk, naprężenie, korozja czy też wilgoć. Również opracowanie i realizacja osłon ochronnych, określających obszar działania elementów wykonawczych ma w tym przypadku decydujący wpływ na zwiększenie bezpieczeństwa pracy w ich otoczeniu.



Rys. 2. Sprzętowa i programowa reprezentacja układów cyfrowych

W przypadku modułu sterującego, problem osiągnięcia bezpieczeństwa stał się bardziej skomplikowany. Uwarunkowane jest to tym, że jednocześnie steruje i kontroluje on pracą całego modułu operacyjnego (mikrosystemu), a więc stał się on dla niego jednostką nadrzędną.

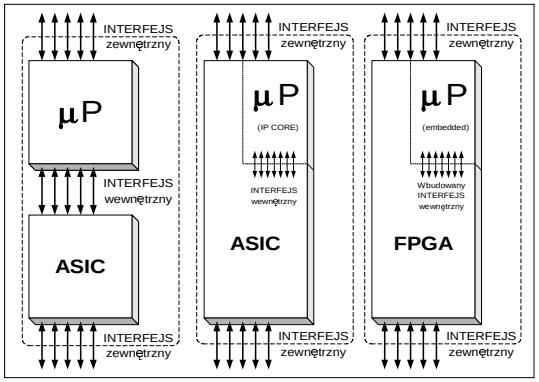
W celu zwiększenia bezpieczeństwa modułu sterującego, początkowo, opracowano nowe metody, osobno dla projektowania sprzętu i oprogramowania, by ostatecznie połączyć je w jedną całość [1], rys. 2. Spowodowało to, że projektując dany mikrosystem, rozpatrywano go w całości, od początku, na poziomie sprzętu i oprogramowania. W ten sposób, realizowane mikrosystemy są układami odpowiadającymi potrzebom projektowym pod względem szybkości, wydajności, bezawaryjności, a przede wszystkim bezpieczeństwa działania.

Jak już wspomniano, bezpieczeństwo modułu sterującego rozpatrywać można na poziomie oprogramowania i/lub sprzętu.

Bezpieczeństwo oprogramowania wiąże się z opracowaniem całkowicie nowych algorytmów działania, składni językowych, całych środowisk, pozwalających tworzyć rozbudowane systemy, począwszy od języków ASSEMBLER, PASCAL, C, ..., itp. Samo zastosowanie

bardziej wyrafinowanego oprogramowania spowodowało, że realizowane mikrosystemy projektuje się przy wykorzystaniu specjalistycznych narzędzi wspomagających projektowanie CAD. Ich wykorzystanie spowodowało, że realizowane projekty mogą być w nich tworzone, testowane, analizowane i weryfikowane w zależności od zaistniałego problemu by ostatecznie umieścić je w układzie programowalnym.

Zwiększenie bezpieczeństwa na poziomie sprzętu, związane było z koniecznością przeprowadzenia transformacji całego realizowanego mikrosystemu. Mikrosystemu składającego się z wielu osobnych układów cyfrowych tworzących jedną całość, rys. 3, na płytce drukowanej (ang. *SoB – System On Board*) do jednego układu programowalnego (ang. *SoC – System On Chip* lub *SoPC – System On Programmable Chip*) [2]. Takie rozwiązanie pozwoliło na pełną symulację i analizę mikrosystemu w całości. Co było niemożliwe w rozwiązaniach typu *SoB*, częściowo możliwe w *SoC*, a już rozwinięte w układzie *SoPC*. Stało się to możliwe dzięki szybkiemu rozwojowi układów programowalnych, których klasyfikację przedstawiono w kolejnym rozdziale.



Rys. 3. Złożoność specjalizowanych modułów sterujących (od lewej SoB, SoC i SoPC)

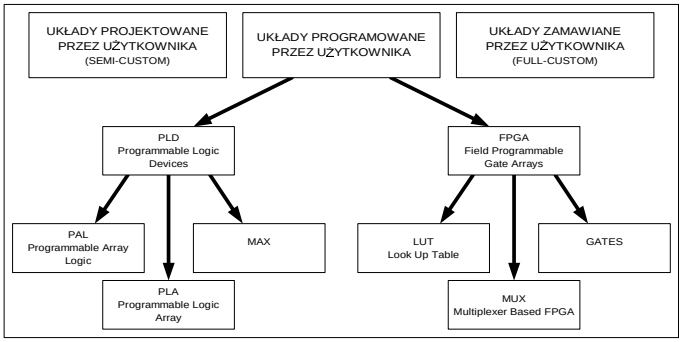
3. UKŁADY PROGRAMOWALNE

Wcześniejsze rozwiązania mikrosystemów sterowania realizowano na poziomie wielkiej, uzupełnionej elementami małej i średniej skali integracji. Jednak rozwój układów bardzo wielkiej skali integracji sprawił, że moc obliczeniowa, realizowanych układów, znacznie wzrosła w stosunku do wcześniejszych rozwiązań. W rezultacie, niezbędnym elementem każdego nowoczesnego urządzenia (mikrosystemu cyfrowego) stały się specjalizowane układy ASIC (ang. *Application Specific Integrated Circuits*) [3, 4]. Głównymi zaletami jest łatwość ich projektowania oraz programowania w początkowej fazie tworzenia mikrosystemu.

Klasyfikację ogólnie dostępnych układów ASIC, przedstawiono na rys. 4 i w Tabeli 1.

Tabela 1. Charakterystyka układów ASIC

Rodzaj	Charakterystyka (* K – Klient, ** P – Producent)
Układy zamawiane przez użytkownika (full-custom)	Założenia (K*) -> projekt, prototyp, testy, produkcja (P**) Proces projektowania i uruchomienia produkcji – długi i kosztowny, układy bardzo szybkie i największe
Układy projektowane przez użytkownika (semi-custom)	Założenia, projekt, prototyp, testowanie (K*) -> produkcja (P**) Proces produkcji – bardzo kosztowny, lecz proces projektowania i produkcji znacznie krótszy, układy o prostych konstrukcjach i strukturach
Układy programowane przez użytkownika	Prefabrykaty (P**) -> założenia, projekt, prototyp, testowanie (K*) Proces projektowania i programowania – bardzo krótki, koszt produkcji – dużo niższy, układy wolniejsze, mniej pojemne pod względem złożoności realizowanych funkcji (prototypy, modele laboratoryjne, serie próbne)



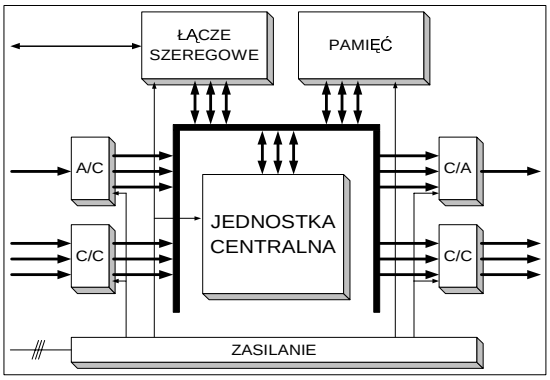
Rys. 4. Klasyfikacja układów ASIC

Jak można zauważyć na rys. 4, układy programowalne przez użytkownika, w przedstawionej klasyfikacji są grupą reprezentującą liczny zbiór układów cyfrowych, które w zależności od właściwości i parametrów technicznych najczęściej stosowane są w wielu modułach sterujących.

4. MIKROSTEROWNIKI

Przedstawiony we wcześniejszych rozdziałach moduł sterujący, ze względu na złożoność i realizowaną funkcję w systemie, określono jako mikrosterownik logiczny [5]. Strukturę tego układu przedstawiono na rys. 5.

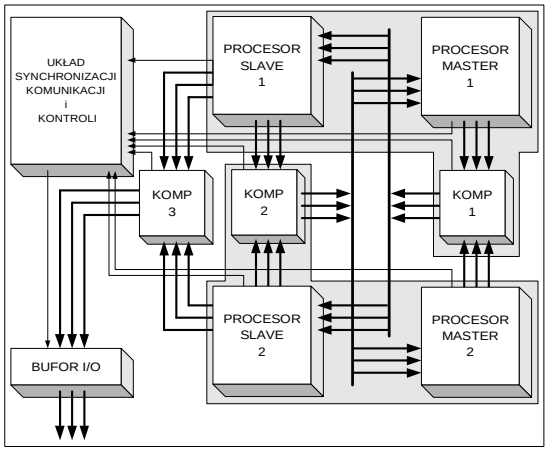
Przedstawiona na rys. 5, architektura programowalnych mikrosterowników logicznych jest architekturą charakterystyczną dla większości cyfrowych układów sterowania, gdyż wyszczególnić w niej można bloki funkcyjne występujące we wszystkich innych rozwiązaniach układowych [5]. Takie rozwiązanie sprawia, że na rynku, dostępny jest szeroki wybór odmian sterowników, różniących się tylko jednostką centralną, którą można zrealizować w jednym układzie programowalnym. To od niej zależy zakres i szybkość wykonywanych operacji, wielkość obszaru obsługiwanej pamięci, sposoby komunikacji szeregowej, dostępność dodatkowych modułów, czy liczba wejść i wyjść całego układu.



Rys. 5. Struktura programowalnych mikrosterowników logicznych

Bardzo często mikrosterowniki logiczne wykorzystywane są w zarządzaniu systemami krytycznymi, a więc systemami od których wymaga się bezwzględnie poprawnego działania. To ze względu na pełnione funkcje w mikrosystemie, konieczne jest określenie warunków pracy dla stanu bezpiecznego [6]. Stan bezpieczny to stan, w którym na skutek awarii możliwe jest ustawienie wejść sterowanego obiektu w stan bezpieczny. Oznacza to, że od niesprawnego modułu sterującego wymaga się przewidywalnego zachowania – odpowiedniego stanu wyjść [6, 7, 8], którym najczęściej jest stan wyłączenia sterownika lub stan najmniejszej, doprowadzonej mocy. W przypadku, gdy taki warunek istnieje, a mikrosterownik może przejść do takiego stanu to mamy do czynienia z bezpiecznym mikrosterownikiem logicznym. Mikrosterownikiem zaprojektowanym i wykonanym bezbłędnie jest układ przedstawiony w [6, 9, 10], który ma możliwość wykrycia własnej awarii.

Przykładem bezpiecznego mikrosterownika logicznego jest układ, którego architekturę przedstawiono na rys. 6.

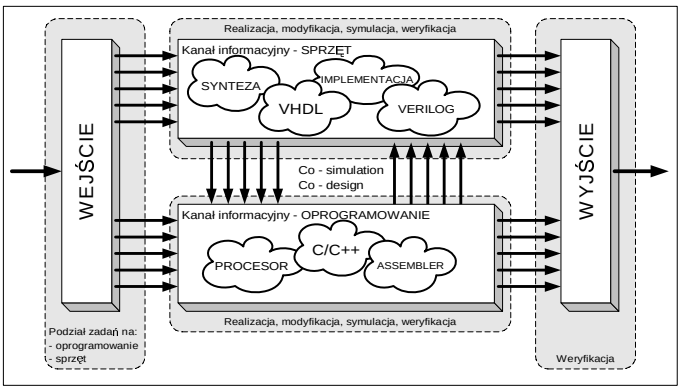


Rys. 6. Architektura mikrosterownika – Halanga, Śnieżka

W celu wykrycia własnej awarii, podczas pracy mikrosterownika, zastosowane zostały dwa kanały informacyjne, które współbieżnie przetwarzają te same informacje. Zrealizowano je tak, by dowolna różnica wyników pomiędzy nimi została wykryta i potraktowana jako błąd systemu, który spowoduje, że cały układ sterujący przejdzie w stan bezpieczny [2, 6, 11]. Reakcję na tego typu błędy systemowe rozwiązano przez zastosowanie bezpiecznych komparatorów logicznych, występujących w obu kanałach. Zastosowanie komparatorów stało się możliwe tylko w przypadku podziału wykonywanych czynności na część: sterującą i operacyjną.

Część sterującą stanowi, para procesorów MASTER, które nadzorują właściwą pracą całego mikrosystemu, generujące sygnały sterujące sterowanym obiektem. Natomiast część operacyjną zrealizowano przez zastosowanie pary procesorów SLAVE, które na życzenie procesorów MASTER, wykonują niezbędne operacje wpływające na bezpieczną pracę układu.

Realizacja kanałów informacyjnych w mikrosterowniku ma na celu wykrywanie nieprzewidywalnych błędów, jakie mogą pojawić się w pracy mikrosystemu. Czyli błędów, do których należą błędy konstrukcyjne, układowe, błędy oprogramowania oraz wszystkie inne anomalia nie przewidziane oraz nie uwzględnione w procesie symulacji i analizy, a mające istotny wpływ na bezpieczeństwo działania układu. Jest to dobrym rozwiązaniem, jednak w przypadku, gdy w każdym z kanałów, równocześnie wystąpi ten sam błąd to nie zostanie on wykryty. Wówczas układ mikrosterownika nie przejdzie do stanu bezpiecznego [6], dlatego aby zabezpieczyć się przed taką ewentualnością, każdy z kanałów, rys. 6, musi być realizowany innymi metodami, rys. 7.



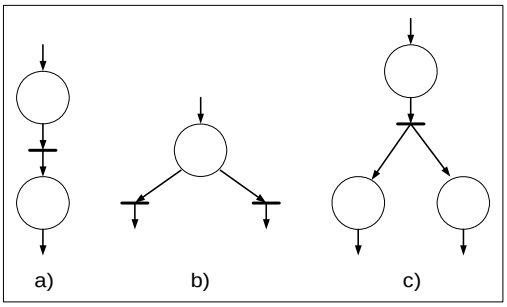
Rys. 7. Mikrosterownik – reprezentacja programowa i sprzętowa

Zastosowanie powyższego rozwiązania powoduje, że każdy kanał realizowany jest odmienną metodą (programową i sprzętową), w których realizowany algorytm działania jest identyczny w obu przypadkach [2]. Natomiast wykrycie różnicy pomiędzy dwoma kanałami: sprzętowym i programowym powoduje wprowadzenie całego układu w stan bezpieczny.

Rozpatrując problem niezawodności działania mikrosterowników logicznych, można wywnioskować, że aby je zwiększyć należy zastosować w nich technologię „lokalnie synchronicznych, globalnie asynchronicznych" układów cyfrowych. Technologia ta związana jest z zastosowaniem odpowiednich sieci działań w realizowanych układach cyfrowych. O doborze sieci w tej technologii, decydują przede wszystkim ich właściwości, które pomagają uzyskać układ (mikrosterownik) znacznie dokładniejszy, bardziej bezpieczniejszy od aktualnych rozwiązań.

Zastosowanie technologii „lokalnie synchronicznych, globalnie asynchronicznych" układów cyfrowych w bezpiecznym mikrosterowniku logicznym wiąże się z zastosowaniem sieci Petriego (lokalnie) oraz sieci „Uścisku dłoni" (ang. *Handshake*) (globalnie). Wykorzystanie synchronicznej sieci Petriego lokalnie, wiąże się z zastosowaniem jej w układach procesorowych (MASTER, SLAVE), gdyż doskonale nadaje się ona do modelowania synchronizacji i komunikacji procesów współbieżnych [12]. Miejsca takiej sieci interpretowane są jako warunki (stany) programu, natomiast tranzycje jako akcje, czyli instrukcje bądź funkcje. Właśnie dlatego sieci Petriego, idealnie odwzorowuje algorytmy realizowanych programów, a ruch jej znaczników modeluje postępujące wykonanie procesów.

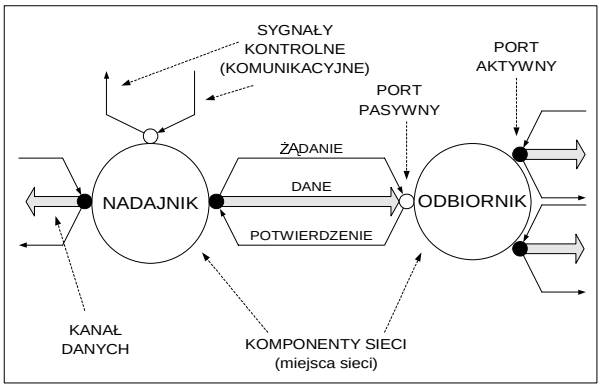
Sieć Petriego jest modelem przepływu sterowania w systemach współbieżnych, której elementy pozwalają na proste modelowanie konstrukcji sterujących, rys. 8:



Rys. 8. Sieć Petriego: a) sekwencja, b) – alternatywa, c) - rozszczepienie

O zaletach stosowania sieci Petriego, przemawia również fakt, że w dziedzinie układów cyfrowych, dostępnych jest wiele narzędzi, wspomagających projektowanie. Przez ich zastosowanie, w analizach i symulacjach sieci Petriego możliwe jest przebadanie wielu istotnych właściwości, takich jak: bezpieczeństwo, ograniczoność, zachowawczość, zakleszczenia jak i żywotność badanej sieci.

Analizując sieć o zasięgu globalnym, czyli sieć, która łączy wszystkie elementy składowe badanego mikrosterownika, najlepszym rozwiązaniem jest zastosowanie asynchronicznej sieci „Uścisku Dłoni" (ang. *Handshake*). Fragment takiej sieci przedstawiony został na rys. 9.



Rys. 9. Sieć typu Handshake

Asynchroniczna sieć „*Handshake*" jest siecią, która realizuje bezpieczne mechanizmy przekazywania informacji pomiędzy jej komponentami (NADAJNIK → ODBIORNIK), [13]. Jest ona siecią, która w odróżnieniu od synchronicznej sieci Petriego nie została opracowana do realizacji procesów współbieżnych. Jej największą zaletą jest zastosowanie sygnałów statusowych (potwierdzenia), sterujących (żądania) oraz kanałów informacyjnych, które tworzone są pomiędzy każdymi dwoma miejscami takiej sieci, rys. 9. To właśnie dzięki nim, możliwa stała się bezpieczna, bezbłędna i precyzyjna synchronizacja całego mikrosystemu.

5. ZAKOŃCZENIE

W artykule przedstawiono, aktualny stan oraz nowe kierunki rozwoju specjalistycznej aparatury sterującej i diagnostycznej, prezentują istotne aspekty jej tworzenia pod względem bezpieczeństwa działania. W ten sposób, pokazano przewagę i złożoność modułu sterującego nad operacyjnym, który odpowiedzialny jest za szereg działań związanych z bezpieczeństwem. Wynika to stąd, że realizuje on określony algorytm działania (program), który steruje nie tylko modułem sterującym (mikrosterownikiem), ale również modułem operacyjnym, a więc i całym mikorsystemem. Takie rozwiązanie zmusza projektanta do rozważania modułu sterującego pod kątem bezpieczeństwa, co zostało zaprezentowane w drugiej części artykułu.

Bardzo często, specjalistyczne urządzenia nadzorujące są wykorzystywane w zarządzaniu systemami krytycznymi, a więc systemami, od których wymaga się bezbłędnego i precyzyjnego działania. Dlatego, zaprezentowano kilka metod, które funkcjonują pod wpływem zastosowania podziału układu mikrosterownika na mniejsze bloki, moduły funkcyjne. W rezultacie tych rozgraniczeń otrzymuje się poprawnie działające urządzenia sterujące, mogące w przypadku awarii przejść do stanu bezpiecznego, a więc stanu, w którym ich praca nie spowoduje żadnych strat materialnych, nie doprowadzi do katastrofy ekologicznej bądź nie narazić życia człowieka.

LITERATURA

[1] A. Bukowiec, M. Węgrzyn: *Design of Logic Controllers For Safety Critical Systems Using FPGAs with Embedded Microprocessors*, *28th IFAC/IFIP Workshop on Real-Time Programming*, Stanbuł, Turcja, 2004, s. 23-28

[2] A. Bukowiec: *Realizacja kontrolerów o podwyższonym stopniu bezpieczeństwa w FPGA o architekturze z wbudowanymi procesorami*, Uniwersytet Zielonogórski 2003, Zielona Góra, Polska

[3] D. Zdrojewska: *Układy FPGA, ich synteza i minimalizacja*, Politechnika Szczecińska 2000, Szczecin, Polska

[4] S. Gorzelanny: *Minimalizacja funkcji logicznych w syntezie układów FPGA*, Politechnika Szczecińska 2000, Szczecin, Polska

[5] S. Brok, R. Muszyński, K. Urbański, K. Zawirski: *Sterowniki programowalne*, Wydawnictwo Politechniki Poznańskiej 2000, Poznań, Polska

[6] M. Śnieżek, W. A. Halang: *Bezpieczny programowalny sterownik logiczny*, Wydawnictwo Politechniki Rzeszowskiej 1998, Rzeszów, Polska

[7] GTI Industrial Automation: *An Introduction to MagLog 24 Inherently Fail-Safe Logic Technology*, Eliminating the Unexpected, Apeldoorn 1993

[8] P. Hildebraudt GmbH&Co.KG: Main Catalogue – The HIMA-Planer-System Brochure HK90.11, Brühl 1991

[9] T. F. Melham: *Higher Order Logic and Hardware Verification*, Cambridge University Press 1993, Cambridge

[10] G. Milne: *Formal Specification and Verification of Digital Systems*, McGRAWHILL Book Company 1994, London, England

[11] A. Bukowiec: *Modelowanie, symulacja i synteza sterowników bezpiecznych z wykorzystaniem języka VERILOG*, Politechnika Zielonogórska 2001, Zielona Góra, Polska

[12] K. Sacha: *Specyfikacja i projektowanie oprogramowania – sieci Petriego*, `http://www.ia.pw.edu.pl/~sacha/petri.html`

[13] S. Furber, J. Sparso: Principles of Asynchronous Circuit Design – A Systems Perspective, Kluwer Academic Publishers 2001