Prof. zw. dr hab. inż. Tadeusz Uhl
Akademia Górniczo – Hutnicza

KOMPUTEROWE WSPOMAGANIE PROJEKTOWANIA MECHATRONICZNEGO

W pracy przedstawiono podstawowe definicje i określenia związane z mechatroniką. Szczególna uwagę zwrócono na projektowanie mechatroniczne. Przedstawiono również możliwości komputerowego wspomagania projektowania mechatroniczneg.

COMPUTER ASSISTED MECHATRONIC DESIGN

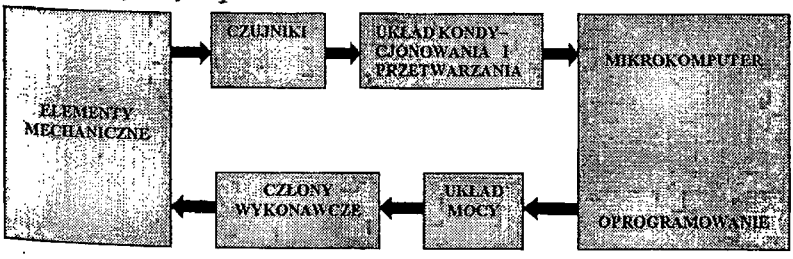
In the paper basic definitions of mechatronic design are shown. The cosideration are focussed on mechatronic design- concepts and methods. The tools of computer assited mechatronics are presented.

1. WPROWADZENIE

Większość współcześnie powstających produktów to produkty mechatroniczne. Produkty te charakteryzują się następującymi cechami:

- elastyczność rozumiana jako łatwość modyfikacji konstrukcji na etapie projektowania, produkcji lub eksploatacji np. modułowość,
- inteligencja czyli możliwość samodzielnego wypracowywania decyzji, uczenia się, lub komunikacji z otoczeniem,
- multifunkcyjność czyli możliwość realizacji różnych funkcji przez to samo urządzenie poprzez prostą jej transformację np. zmianę oprogramowania,
- niewidoczny dla operatora sposób realizacji procesu technologicznego w sposób w pełni zautomatyzowany co wymaga od urządzenia specjalnego interfejsu użytkownika w celu komunikacji z operatorem,
- zależność od rynku i możliwość technologicznych producenta.

Cechy te można osiągnąć dzięki charakterystycznej architekturze produktów schematycznie przedstawionej na rysunku 1.



Rys. 1. Architektura układu mechatronicznego.

Przykładami produktów mechatronicznych mogą być kserokopiarka, kamera video, silnik spalinowy, system ABS, robot, obrabiarka sterowna numerycznie i wiele innych. Podsumowując wyżej wymienione cechy w wyniku realizacji procesu projektowania mechatronicznego powinno powstać urządzenie pewne w działaniu, tanie, funkcjonalne, łatwe w obsłudze i niezawodne.

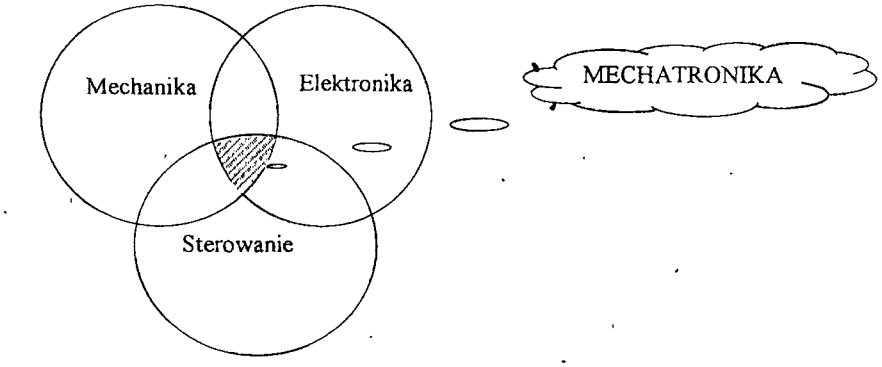
Jak widać z powyższego opisu typowej struktury układów mechatronicznych są to układy interdyscyplinarne, co oznacza że są złożone z współpracujących z sobą elementów o różnej naturze fizycznej. W układach mechatronicznych wyróżnia się następujące elementy:

- elementy mechaniczne,
- elementy elektryczne i elektroniczne
- elementy informatyczne w tym, sprzęt i oprogramowanie.

Elementy mechaniczne stanowią najczęściej człony wykonawcze oraz elementy przeniesienia napędu. Układy elektryczne są układami napędowymi oraz zasilającymi. Elektronika jest elementem czujników, układów sterowania oraz układów zasilania. Sterowanie we współczesnych produktach jest realizowane w sposób algorytmiczny poprzez odpowiednie oprogramowanie uruchamiane na procesorze sterownika. W wielu konstrukcjach wszystkie jej elementy mają charakter modułów co oznacza, ze rozbudowując np. cześć wykonawcza systemu można również rozbudować o odpowiednie moduły część związaną ze sterowaniem i oprogramowanie. Bardzo często w tym zakresie wykorzystuje się rozwiązania sieciowe np. w samochodach sieć typu CAN. Z tego też względu w konstrukcjach mechatronicznych istotnym elementem są standardy interfejsów umożliwiających realizację połączeń modułów zarówno mechanicznych, jak elektronicznych i oprogramowania. Konstrukcja współczesnych produktów mechatronicznych w większości przypadków opiera się na integracji elementów o różnej naturze fizycznej. Stopień integracji w produktach mechatronicznych oraz jakość współpracy pomiędzy elementami systemu są miara ich zaawansowania technologicznego. Producenci wielu wyrobów doszli do wniosku, że integrowanie części mechanicznej z elektroniką i oprogramowaniem w ramach jednego produktu ma ogromny wpływ na koszt jego wytworzenia, jakość realizacji procesu technologicznego, jak również niezawodność działania.

Projektowanie produktów mechatronicznych wymaga specjalnego podejścia do realizacji projektów nazwanego w literaturze podejściem mechatroniczym, którego metodologia i techniki są pewnym działem mechatroniki.

Definicji mechatroniki jest wiele, obecnie przejęła się w Europie definicja podana przez Komisję Europejską, Dyrektoriat XII, ma ona brzmienie: „przez mechatronikę rozumie się synergiczna kombinację mechaniki, elektroniki oraz algorytmicznego sterowania".

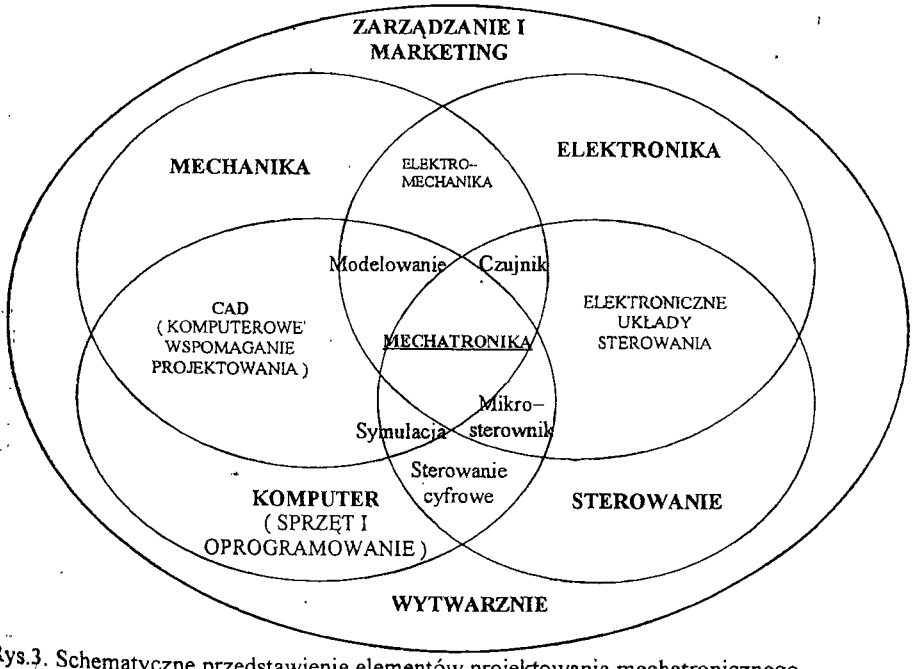


Rys.2. Schematyczne przedstawienie definicji mechatroniki

Synergia w tej definicji jest rozumiana jako uzyskiwanie z połączenia części o różnej naturze fizycznej pewnej wartości dodanej. Schematycznie definicje tą można przedstawić jak na rysunku 2. We wszystkich definicjach bardzo istotnym ich elementem jest związek pomiędzy produktem a rynkiem. Określenie produktu dobrego jest zdefiniowane poprzez akceptacje produktu przez rynek.

Typowymi cechami projektowania mechatronicznego jest interdyscyplinarność, integracja, zorientowanie na rynek oraz jakość produktu dostosowana do wymagań użytkownika. Interdyscyplinarny charakter projektowania mechatronicznego wyraża się koniecznością uwzględnienia w procesie realizacji konstrukcji interdyscyplinarnej natury projektowanych wyrobów, natomiast integracja procesu projektowania umożliwia traktowanie elementów o różnej naturze fizycznej z jednakową wagą.

Integracja w projektowaniu mechatronicznym ma szczególne znaczenie ze względu na realizację wielu projektów przez zespoły interdyscyplinarne złożone ze specjalistów różnych specjalności. Zorientowanie współczesnych produktów na rynek wymaga udziału w procesie projektowania specjalistów z zakresu marketingu i ekonomii, których wytyczne wynikające z potrzeb konsumentów muszą być uwzględnione we wszystkich fazach projektowania, a w szczególności w fazie formułowania założeń i wyboru koncepcji. Podejście mechatroniczne znacznie różni się od podejścia klasycznego, w którym poszczególne elementy projektowanej konstrukcji tworzone są oddzielnie. Schematycznie elementy procesu projektowania mechatronicznego przedstawiono na rysunku 3.



Rys.3. Schematyczne przedstawienie elementów projektowania mechatronicznego.

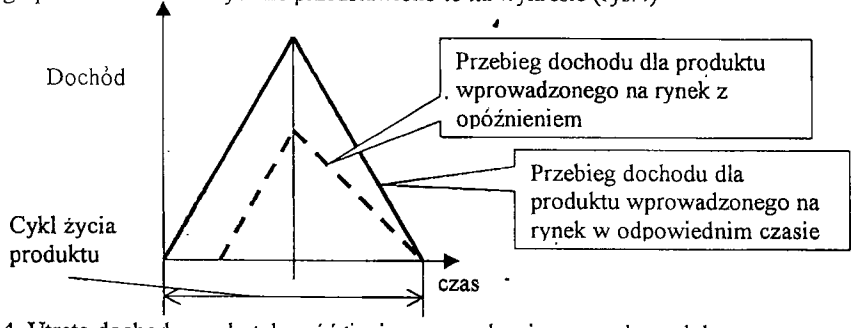
W większości przypadków projektowania mechatronicznego, proces ten jest realizowany poprzez zespół współpracujących projektantów, technologów, inżynierów jakości i specjalistów z zakresu marketingu. W zakresie projektowania wymagana jest od zespołu wiedza z zakresu projektowania elementów mechanicznych, elektronicznych i

oprogramowania. Jednym z najbardziej istotnych elementów projektowania jest przepływ informacji w zespole realizującym projekt.
Bardzo istotne jest aby inżynierowie wchodzący w skład zespołu realizującego projekt posiadali wiedzę interdyscyplinarną, która umożliwi łatwe przekazywanie informacji pomiędzy członkami zespołu. Szczególnie ważne jest to dla osoby odpowiedzialnej za zarządzanie projektem.
Innym aspektem projektowania mechatronicznego jest uwzględnienie w czasie projektu kosztów życia produktu (LCC), umożliwia to optymalizację tych kosztów na każdym etapie projektowania. Przez życie produktu rozumie się tutaj następujące etapy: wytwarzanie elementów składowych produktu (komponentów), zbieranie komponentów (logistyka), wytwarzanie produktu, serwis i pomoc w eksploatacji, likwidacja, recykling. Koszt na każdym z tych etapów powinien być uwzględniony przy optymalizacji produktu.

Jak można wnioskować z przedstawianych wyżej rozważań do realizacji projektowania mechatronicznego konieczna jest wiedza inżynierska bardzo szeroka obejmująca dyscypliny wchodzące w skład mechatroniki, umiejętność pracy w zespole oraz komunikowanie się pomiędzy specjalistami różnych dziedzin, oraz posługiwanie się nowoczesnymi narzędziami pracy inżyniera. Jednymi z podstawowych narzędzi służącymi do realizacji projektów mechatronicznych integrującymi ten proces są metody komputerowego wspomagania projektowania. Podstawa tych metod jest przede wszystkim modelowanie układów mechatronicznych, analiza ich własności oraz synteza produktu.

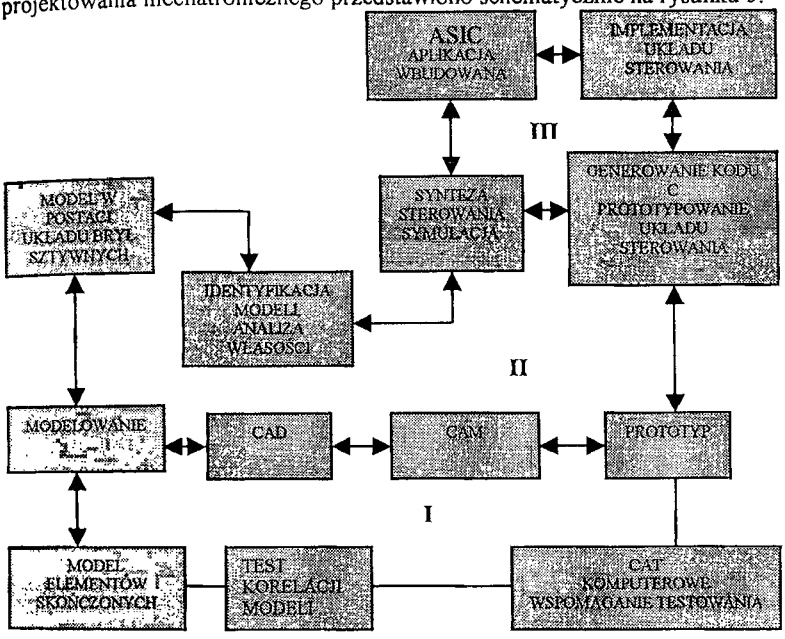
2. KOMPUTEROWE WSPOMAGANIE W PROJKETOWANIU MECHATRONICZNYM

Do realizacji wszystkich etapów projektowania mechatronicznego stosuje się narzędzia komputerowego wspomagania. Podstawowymi etapami są: opracowanie specyfikacji produktu, analiza specyfikacji, opracowanie i wybór koncepcji, opracowaniu projektów detali, wykonaniu prototypu, badanie prototypu, opracowaniu procesu produkcyjnego. Zadaniem oprogramowania do realizacji projektu mechatronicznego jest przede wszystkim integracja różnych dziedzin, poprzez możliwość ujednolicenia procesu modelowania oraz uwzględnienie wzajemnego oddziaływania elementów o różnej naturze fizycznej. W szczególności to ostatnie działanie jest istotne z punktu widzenia mechatroniki. Może ono być realizowane poprzez zamodelowanie układu logicznego produktu z wykorzystaniem narzędzi komputerowego wspomagania modelowania typu UML, AML, XML. Jednym z podstawowych idei zastosowania projektowania mechatronicznego jest skrócenie czasu powstawania produktu i uruchamiania jego produkcji. Uzasadnione jest to możliwością uzyskania dużych zysków poprzez wyprzedzenie konkurencji we wprowadzeniu na rynek danego produktu. Schematycznie przedstawiono to na wykresie (rys.4)



Rys.4. Utrata dochodu na skutek opóźnienia wprowadzenia na rynek produktu

Opracowany przez zespół kierowany przez autora system komputerowego wspomagania projektowania mechatronicznego przedstawiono schematycznie na rysunku 5.

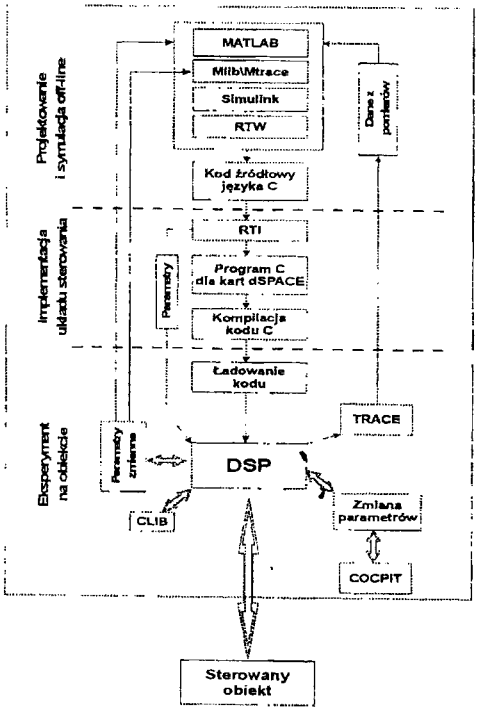


Rys. 5. Schemat systemu komputerowego wspomagania projektowania mechatronicznego.

W Katedrze Robotyki i Dynamiki Maszyn AGH w Laboratorium Mechatroniki zrealizowano ten system w oparciu o stacje robocze, komputery klasy PC połączone siecią typu Ethernet. Przedstawioną na schemacie architekturę oprogramowania można podzielić na trzy pętle. W pierwszej pętli dokonuje się analizę i syntezę konstrukcji ze względu na jej własności mechaniczne, natomiast w drugiej realizuje się proces analizy, syntezy, prototypowania układu sterowania. Natomiast trzecia pętla stanowi etap implementacji. Pętle jeden i dwa mają wspólne części, które stanowi ogniwo integracji różnych dziedzin wchodzących w zakres projektu mechatronicznego. Realizacja pętli trzeciej zależy od sposobu implementacji. Na różnych etapach realizacji interdyscyplinarnych projektów mechatronicznych stopień integracji dziedzin jest różny: na etapie tworzenia i analizy koncepcji, budowy i badania prototypu bardzo duży natomiast na etapie projektowania detali konieczność integracji jest znacznie mniejsza. Większość realizowanych projektów mechatronicznych rozpoczyna się od sporządzenia jej szkicu za pomocą systemu CAD. Na podstawie opracowanej geometrii (modelu geometrycznego) oraz przyjętych własności materiałów buduje się modele projektowanego obiektu (prototyp wirtualny). W wyniku symulacji tych modeli dokonuje się oceny ich własności oraz spełnienia sformułowanych założeń projektowych. Jest to pierwszy etap realizacji projektu mechatronicznego od wyników, którego zależy wybór koncepcji do dalszej realizacji. W drugim etapie za pomocą systemu CAD projektuje się proces technologiczny wykonania oraz montażu prototypu. Po wykonaniu prototypu, przystępuje się do jego testowania. Obecnie stosowane w praktyce systemy badania prototypów są najczęściej wielokanałowymi systemami pomiarowymi umożliwiającymi realizację różnego rodzaju czynnych eksperymentów na fizycznym prototypie budowanego układu. Ze względu na złożoność tych systemów do przeprowadzenia

eksperymentu oraz opracowania jego wyników konieczne jest zastosowanie systemów komputerowego wspomagania eksperymentu oraz przetwarzania jego wyników (CAT ang. Computer Assisted Testing).

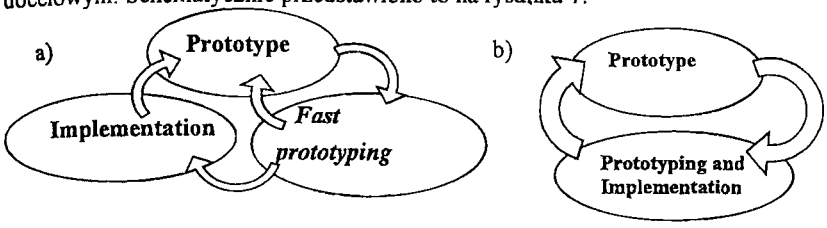
Wyniki testów prototypu w pierwszym etapie projektu służą do identyfikacji przyjętych modeli oraz ich weryfikacji. W zależności od rodzaju układu (egzemplarz jednostkowy, produkt masowy) w różny sposób realizuje się jego układ sterowania. Z tego też względu do implementacji układu sterowania (projektowania) stosuje się różne narzędzia programowe. Ze względu na konieczność współpracy oprogramowania do realizacji projektów różnego typu konstrukcji (mechanicznych, elektrycznych, elektronicznych) oprogramowanie to musi posiadać standardowe interfejsy umożliwiające komunikację pomiędzy poszczególnymi programami. Bardzo istotnym elementem realizacji tak przedstawionego procesu projektowania wspomaganego zastosowaniem narzędzi w postaci oprogramowania jest etap prototypowania, który można podzielić na: prototypowanie wirtualne i prototypowanie fizyczne. Pierwsze realizowane jest z wykorzystaniem symulacji, drugie natomiast wymaga współpracy oprogramowania z prototypowanym obiektem. Stosuje się tutaj dwie techniki, technikę szybkiego prototypowania oraz technikę 'hardware in the loop simulation'. Technika szybkiego prototypowania polega na symulacji w czasie rzeczywistym sterownika. Do tego celu stosuje się specjalizowane układy najczęściej parte o procesory sygnałowe. Jednym z przodujących rozwiązań z tego zakresu jest rozwiązanie firmy dSPACE schematycznie przedstawione na rysunku 6. Technika 'hardware in the loop simulation'



Rys.6. Prototypowanie układów sterowania poprzez symulację w czasie rzeczywistym

polega na zastosowaniu rzeczywistego sterownika, natomiast część wykonawcza projektowanego urządzenia jest realizowana poprzez symulacje. W rzeczywistości oznacza to ,że sygnały z czujników na rzeczywistym obiekcie są otrzymane droga symulacji modelu w

czasie rzeczywistym, jak również model reaguje na sterowania pochodzące ze sterownika w sposób identyczny jak obiekt [4]
W chwili obecnej, ze względu na zwiększenie możliwości obliczeniowych stosownych do implementacji sterowników przemysłowych stosuje się technikę prototypowania na układzie docelowym. Schematycznie przedstawiono to na rysunku 7.



Rys.7 Prototypowanie układów sterowania; a) z zastosowaniem specjalizowanych układów, b) na sterowniku docelowym.

Zastosowanie prototypowania na sterowniku docelowym jest możliwe w przypadku, gdy sterownik posiada możliwość programowania w języku C. Tego typu implementacja jest stosowana gdy jako sterownik stosuje się układ mikroprocesorowy lub komputer przemysłowy. Nieco inne podejście opracowano w Katedrze Robotyki i Dynamiki Maszyn do implementacji układów sterowania stosowanych w produkcji masowej w oparciu o elementy elektroniczne typu ASIC/FPGA [6]

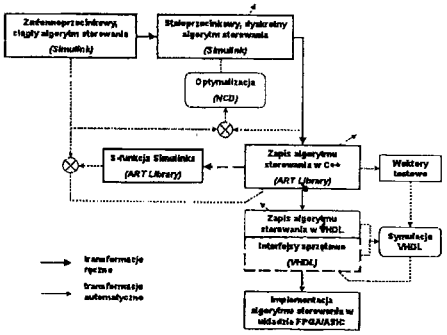
3. IMPLEMENTACJA UKŁADÓW STEROWANIA - PODEJŚCIE MECHATRONICZNE

Wybór platformy sprzętowej sterownika zależy generalnie od przewidywanej wielkości produkcji. Dla produkcji jednostkowej stosowane są przeważnie komputery przemysłowe, dla produkcji średnioseryjnej sterowniki autonomiczne, zaś sprzęt oparty na układach specjalizowanych (ASIC) stosowany jest w przypadku produkcji wielkoseryjnej. Wybór taki dokonywany jest ze względu na szereg własności, którymi dla tej ostatniej platformy są: wysoki stopień integracji, niezawodność, duże możliwości optymalizacji, niski jednostkowy koszt wytworzenia, ale równocześnie mała elastyczność i duży koszt implementacji. Cechy komputerów przemysłowych są dokładnie przeciwne, natomiast sterowniki autonomiczne posiadają właściwości pośrednie. W chwili obecnej rozwój układów elektronicznych poszedł w kierunku wytwarzania układów typu FPGA - programowanych układów typu ASIC. Stwarza to nowe możliwości implementacji układów sterowaniu w oparciu o schemat postępowania przedstawiony na rysunku 5.
Algorytm sterowania, który ma zostać zaimplementowany w układach FPGA/ASIC jest zwykle ciągły w czasie, wykorzystuje ciągłe wartości sygnałów i opisany jest za pomocą równań matematycznych, albo za pomocą schematu blokowego. Opis taki nie nadaje się do bezpośredniego zastosowania. Przede wszystkim potrzebna jest dyskretyzacja w czasie. Drugą znaczącą transformacją jest dyskretyzacja amplitudy (kwantyzacja). Zastosowanie wyłącznie arytmetyki stałoprzecinkowej pozwala na znaczną redukcję kosztu realizacji algorytmu sterowania. Programy używane do syntezy układów FPGA i ASIC akceptują ich opis w specjalnych językach (ang. HDL – Hardware Description Language), rzadko znanych inżynierom automatykom. Co więcej, kodowanie w językach HDL jest procesem czasochłonnym i podatnym na błędy. Z używaniem takiej notacji związane są również problemy z testowaniem i weryfikacją poprawności działania i osiągów algorytmu. Istniejące narzędzia pozwalają na dokładną symulację elektroniki na poziomie bramek, wspierają, w

pewnym stopniu, symulację na poziomie algorytmu, ale sprawdzenie jakości sterowania, tj. równoczesna symulacja sterownika i sterowanego obiektu jest jak dotąd niemożliwa. Z drugiej strony, programy powszechnie używane podczas projektowania algorytmów sterowania, takie jak Simulink, pozwalają w stosunkowo prosty sposób symulować urządzenie mechaniczne równocześnie ze sterowaniem, opisanym za pomocą schematu blokowego, ale nie umożliwiają symulacji algorytmu zapisanego przy użyciu języków HDL. Dlatego sprawdzenie zgodności opisu sterownika w języku HDL z pierwotną formą w postaci schematu blokowego wydaje się być jednym z najtrudniejszych problemów podczas implementacji sprzętowej. Z powyższych powodów opracowano systematyczne podejście do implementacji sterowania w układach FPGA/ASIC.

Głównymi celami podczas tworzenia tej procedury było ograniczenie ilości i zakresu dokonywanych ręcznie transformacji algorytmu, zastosowanie, tam gdzie to możliwe, komercyjnie dostępnego oprogramowania oraz próba wyraźnego wyodrębnienia kodu HDL nie należącego bezpośrednio do algorytmu sterowania. Pierwszy cel wynika z kosztów przekształceń wykonywanych ręcznie – są one czasochłonne i są źródłem częstych błędów. Drugi cel zmniejsza koszty tworzenia procedury i pozwala na natychmiastowe korzystanie z udogodnień nowych wersji oprogramowania komercyjnego bez dodatkowych nakładów na adaptację. Osiągnięcie ostatniego z wymienionych celów tworzy potencjalne możliwości realizacji szybkiego prototypowania na docelowym sprzęcie.

Procedura, pokazana schematycznie na rys. 8, składa się z kilku etapów. Niektóre z nich są realizowane ręcznie, a pozostałe automatycznie, przez odpowiednie oprogramowanie. Pierwsze przekształcenie do postaci stałoprzecinkowej, dyskretnej w czasie, jest realizowane w Simulinku, z wykorzystaniem standardowej biblioteki stałoprzecinkowej Fixed-Point Blockset. Pozwala to na zbadanie dynamiki sygnałów i efektów obliczeń wykonywanych z przyjętą precyzją i zakresem reprezentacji sygnałów oraz charakterystykami nasycenia lub przepełnienia. Transformacja ta musi być wykonana ręcznie, z tym że wartości parametrów mogą być dopasowane z zastosowaniem biblioteki Nonlinear Control Design Toolbox (NCD), starannie dobranej funkcji celu i metody optymalizacji, tak aby uzyskać jakość sterowania jak najbardziej zbliżoną do uzyskiwanej z zastosowaniem ciągłego w czasie sterowania zmiennoprzecinkowego. Środowisko Simulinka umożliwia łatwą ocenę nowej postaci sterownika.



Rys.8. Procedura implementacji algorytmu sterowania w układach ASIC/FPGA

Następny, także realizowany ręcznie etap wykorzystuje oprogramowanie firmy Frontier Design. A|RT Library jest biblioteką dla kilku popularnych kompilatorów C++, dostarczającą klasy stałoprzecinkowe, która umożliwia emulację obliczeń stałoprzecinkowych w programach pisanych w języku C++. A|RT Builder umożliwia automatyczną syntezę kodu w języku VHDL lub Verilog (są to jedne z najbardziej popularnych języków HDL) z programu

w C++, pisanego z użyciem biblioteki stałoprzecinkowej. A|RT Library używa takich samych typów co bloki Simulinka z Fixed-Point Blockset, tak więc możliwe jest wzajemnie jednoznaczne odwzorowanie między schematem blokowym a kodem C++, co ułatwia ten etap procedury i redukuje ilość możliwych do popełnienia błędów.
Po dodaniu kodu interfejsu Simulinka do algorytm sterowania w języku C++ można utworzyć tzw. s-funkcję. S-funkcja definiuje działanie nowego bloczka, który może zostać elementem dowolnego schematu blokowego w Simulinku. Równocześnie źródło tej samej s-funkcji może być zsyntetyzowane przez A|RT Builder do kodu VHDL. Umożliwia to symulację tego samego kodu sterownika, który jest używany do syntezy układu FPGA/ASIC, razem z modelem układu sterowanego.
Ważną zaletą A|RT Buildera jest jego „przeźroczystość", która oznacza zależność generowanego kodu VHDL tylko i wyłącznie od stylu programowania w C++. Dzięki wyborowi odpowiedniego stylu, topologia otrzymywanego układu elektronicznego może być optymalizowana pod względem szybkości działania lub rozmiaru poprzez automatyczne syntezowanie bardziej złożonych metod zarządzania przepływem danych, jak przetwarzanie potokowe czy współdzielenie zasobów. Dokładnie przy tym wiadomo jaki kod w VHDL jest generowany dla każdej instrukcji i wyrażenia w C++, zachowywane są także wszystkie identyfikatory z C++. Pozwala to na zachowanie pełnej kontroli nad generowanym kodem w VHDL, uwalniając od konieczności ręcznego kodowania. Co więcej, możliwe jest łączenie otrzymanego zapisu algorytmu sterowania w VHDL z innym kodem w tym języku, np. opisującym interfejsy sprzętowe z urządzeniami peryferyjnymi, takimi jak przetworniki analogowo-cyfrowe i cyfrowo-analogowe czy też synchronizację czasową obliczeń. ,
Tworzy to możliwości realizacji idei szybkiego prototypowania na docelowej platformie sprzętowej przez stosowanie niezmiennej części organizacyjnej i „peryferyjnej" kodu VHDL i zmianę oraz badanie działania części opisującej algorytm sterowania.
Wygenerowany zapis algorytmu sterowania w VHDL, razem z opisem niezbędnych interfejsów urządzeń peryferyjnych jest następnie kompilowany w celu otrzymania listy połączeń na poziomie bramek, która opisuje architekturę układu scalonego przy użyciu bramek, przerzutników i innych elementów podstawowych dostępnych w danej technologii. Później lista połączeń wymaga dopasowania, podczas którego fizyczne zasoby konkretnego układu scalonego są alokowane dla każdego elementu podstawowego z listy. W trakcie opisywanych badań do kompilacji był używany program Synopsys FPGA Express, a do dopasowywania Altera MAX+plus II.
W przypadku konieczności dokonania ręcznych modyfikacji kodu w VHDL potrzebna jest metoda sprawdzania jego poprawności. Analiza może być przeprowadzona za pomocą symulatora VHDL. Tworzenie i weryfikacja wektorów testowych dla symulatorów VHDL są wspomagane przez A|RT Library.

4. ZASTOSOWANIE UML DO PROJKETOWANIA MECHATRONICZNEGO

Proces projektowania mechatronicznego najczęściej oparty jest o modele. Jako pierwsze do modelowania konstrukcji mechatronicznego zastosowano grafy wiązań [8](ang. Bond graph), później stosowano modele analityczne natomiast obecnie stosuje się schematy blokowe (ang. block diagrams)[7] oraz podejście obiektowe sformalizowane w języku UML [11]. Podejście oparte o zapis logiki projektowanego układu w postaci języka UML może być zastosowane do realizacji procedur przedstawionych w poprzednim rozdziale. UML (unified modeling language) stworzono w celu modelowania systemów informatycznych. Jest to uniwersalny, obiektowo zorientowany język przydatny do projektowania systemów niezależnie od ich przeznaczenia i od języka, w którym będzie zaimplementowany docelowy system. UML jest

efektem ujednolicenia trzech znanych wcześniej języków: OMT2 (object modelling technique) Jamesa Rumbaugha, metody Booch'a (Grady Booch) oraz OOSE (object oriented software engineering) Ivara Jacobsona. Zawiera on też pewne elementy języka SDL (Specification and Design Language, 1976 CCITT) oraz metodyki E-R (entity relationship). Rezultatem pracy dwu pierwszych autorów był UML 0.8 powstały w roku 1995 [11]. Model projektowanego systemu ma postać diagramów graficznych. Każdy rodzaj diagramu pokazuje te elementy przyszłego systemu, które są istotne z wybranego punktu widzenia, a pomija bądź upraszcza inne elementy. Wysoki poziom abstrakcji pozwala na zwięzły opis nawet bardzo dużych systemów składających się z elementów o różnej naturze fizycznej. Jest on uzupełniany o opis werbalny w postaci scenariuszy oraz komentarze i inne informacje. Oczywistym jest, że użycie tylko jednego typu diagramu nie jest wystarczające. Praktycznie zawsze wykorzystuje się opisany dalej dwa diagramy statyczne: diagram przypadków użycia i diagram klas. Wybór dodatkowych diagramów jest uzależniony od aktualnych potrzeb i doświadczenia projektantów. Mogą oni wybierać spośród diagramów dynamicznych: diagram aktywności, diagram sekwencyjny, diagram współpracy, diagram stanu oraz spośród diagramów implementacyjnych; diagram komponentów, diagram konfiguracji, zwany też diagramem wdrożeniowym. Diagramy i ich elementy można grupować w pakietach i w podsystemach. Pakiety i podsystemy mogą wchodzić w skład innych pakietów i podsystemów. Sformalizowanie UML pozwoliło na stworzenie oprogramowania CASE (ang. computer aided system engineering) do komputerowego wspomagania projektowania z wykorzystaniem UML. Pakiety CASE oferują narzędzia wspomagające kolejne etapy projektowania oraz służą do dokumentowania postępu prac i wprowadzanych zmian. CASE sprawdza formalną poprawność i zgodność kolejnych diagramów z diagramami wcześniej wykonanymi. Pozwala to na automatyczne wychwycenie błędów formalnych i braku spójności elementów projektu. W rezultacie koszt i strata czasu na dokonanie poprawek i modyfikację projektu na wczesnym etapie jego realizacji są relatywnie niskie. Oprogramowanie zbudowane do wspomagania projektów informatycznych może być bezpośrednio wykorzystane do projektowania złożonych systemów mechatronicznych. Zastosowanie to zostało przedstawione w [11].

5. WNIOSKI O UWAGI KOŃCOWE

Przestawiona w pracy koncepcja projektowania mechatronicznego jest jedną z wielu możliwych i stosowanych w praktyce. Autor kierując zespołem zrealizował w oparciu o przedstawioną metodykę projektowania szereg projektów uzyskując efekty skrócenia czasu realizacji projektu. Dodatkową zaleta stosowania tego podejścia jest również możliwość optymalizacji wielokryterialnej produktu a przede wszystkim możliwość optymalizacji współpracy elementów o różnej naturze fizycznej.

6. LITERATURA

[1] Ackar M., Makra I., Perrey E., *Mechatronics, the basis for new Industrial development*, Computational mechanics Publications, New York, 1994.

[2] Hall M.,J., *Designing for the life cycles*, in Concurrent engineering, Ed. Medhat S. RSP, New York, 1997.

[3] Uhl T., *Hardware in the loop simulation – nowa metoda prototypowania układów sterowania.* PAK no.4, 1996.

[4] Uhl T., Bojko T., Mrozek Z., Szwabowski W.- *Rapid prototyping of mechatronic systems* , Jurnal of Theoretical and Applied Mechanics no. 3,vol.38, 2000

[5] T. Uhl, Z. Mrozek, M. Petko, *Rapid control prototyping for flexible arm*, 1-st IFAC Conference on Mechatronic Systems, Preprint, vol. 2, pp. 489-494, Darmstadt, 2000

[6] Petko M., *Implementacja algorytmów sterowania w układach ASIC. FPGA*, PAK, no.1, 2002.
[7] Uhl T., (Ed.) *Projektowanie Mechatroniczne*, Wydawnictwo KRiDM, AGH, 1999.
[8] Wojnarowski J., *Grafy i Mechatronika*, XV Ogólnopolska Konferencja Naukowo - Techniczna, Teorii Maszyn i Mechanizmów, Białystok, 1996.
[9] Uhl T., Ród J., *Modeling and identification of robotic systems using MATLAB/SIMULINK einviroment*, Proc. of ICRAM 95, Istanbul, September, 1995
[10] Uhl T., Kohut P., *Zastosowanie obliczeń symbolicznych w modelowaniu układów mechanicznych*, Materiały Konferencji Diagnostyki 95, Węgierska Górka, Marzec, 1995.
[11] Mrozek Z., *Metodyka wykorzystania UML w projektowaniu mechatronicznycm*. PAK, no.1, 2002.
[12] Dinsdale J.,Mechatronics:*The International Scene, Mechatronic Systems Engineering*, Int. Journal of Mechtronics,vol.1,pp101-105,1990.
[13] Salminen V.,Verho A.,*Systematic and innovative design of a mechatronic product* Mechatronics,vol.2,No.3,1992,pp 257-275
[14] VDI- Richtlinien, VDI 2221, *Systematic Approach to Design of Technical Systems and Products*, VDI-Verlag GmbH,Dusseldorf,1987
[15] Buur,J.,*A theoretical Approach to Mechatronics Design*, Ph.D. Dissertation,Technical University of Denmark,Lyngby,1990.
[16] Konstantinow M.,Patarinski,Sotirov Z.,Markow L.,*Mechatronics*,Proc. 7th IFToMM World Congress,Sevilla,vol.2,pp.1453-1456,1987.
[17] Deasley P.J, *Control: the integrating factor in mechatronics*, Mechatronics System Engineering,vol.1,no.1,pp.11-17,1990.
[18] Keys L.,K.,*Mechatronics,systems and elements*, Proceedings , IEMT Symposium,Washigton,1990,pp.329-333.
[19] Bradley D.A,Dorey A.P,*Engineering design and Mechatronics*, Proc IEE Coloquim on Mechatronics,1991,pp.1-12.
[20] Simonds R., *Mechatronics; the best of the both worlds*, Electrical Review,vol.224,No.1,pp 21-22.
[21] Bradley D.A,Dawson D, *Mechatronics-on integrating approach in engineering design*, Proc. of the IMechE,Cambridge,UK,1990,pp.195-199.
[22] French M.J.,*The partition of functions in mechatronic design*, Proc of IMechE, Cambridge, UK, 1990,pp.209-211.
[23] Taylor G.T, *The mechatronics curriculum for modern engineer*, Proc of IMechE, Cambridge, UK, 1990,pp.2271-279.
[24] Keys L.,K.,Darks C.M, *Mechatronics, systems and Technology: a perspective*, IEEE Trans. on Components, Hybrids and Manufacturing Technology, vol.14, no.3, 1991, pp.457-461.
[25] Murahami T.,Ohnishi K., *Advance motion control in mechatronics - a tutorial*, IEEE International Workshop ,in Inteligent Motion Control, Istambule,1990,vol.1,pp SL9-
Cop V. *Methodology of mechatronics system decomposition*, Mechatronics,1992, vol.2, No 2, pp.199-206.
[26] HANSON M., *'Teaching mechatronics at tertiary level'*, Mechatronics, Vol 4. No 2,217-225,1994
[27] YAMAZAKIK, SUZUKI H, HOSTI T., *„Methodology of education and R&D in mechatronics"*, Appl. Eng. Edu., Vol 1, No 1 ,35-41,1985
[28] BECKMAN J,. *'USA joints the mechatronics race"*, New Scientist, Vol 108, 28 Nov 1985
[29] WILLGOSS R.A. *„A Taxonomy for Mechatronic Education"*, proc. of ICRAM, August 1995, Vol 1,240-245

[30] DAY .B, *The Mechatronic Manufacturing Solution - Myth of Panacea*", Proc. of ICRAM, August 1995, Vol 1,247-254
[31] HEWIT J., *Mechatronics Design - „The Key to Performance Enhancment*", Proc of ICRAM, August 1995, Vol 1,18-24
[32] FROELICHER M., *"Mechatronics : from automobile to watch*", proc of 1st Europe Asia Cngress on Mechatronics, Besancon, Vol 3 ,887-927, 1996
[33] PUGH A.,STUART Z., *"Total Design Integrated Method for Successful Product Engeneering"*, Addison-Wesley, MA,1994
[34] Gawrysiak M. *Mechatronika i Projektowanie Mechatroniczne*, Wydawnictwo Pol. Białostockiej, 1997.
[35] GASCOIGNE B., *Concurrent Engineering, a mudate for PDM*, CiME, December/January, 1995
[36] HANSELMANN H , *Hardware in-the-loop simulation as a standard approach for development, customization and production test.* SEA 930207, March 1-5,1993
[37] Uhl T., *Mechatronika nowe podejście do projektowania i kształcenia*, PAR vol.1, no.1, 1997.