Karol SZOSTEK
POLITECHNIKA RZESZOWSKA, KATEDRA TERMODYNAMIKI I MECHANIKI PŁYNÓW,
ul. Powstańców Warszawy 8, 35-959 Rzeszów

Projektowanie procesora sekwencyjnego i symulacja w środowisku MATLAB/simulink

Dr inż. Karol SZOSTEK

Ukończył studia wyższe oraz doktoranckie na Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie na Wydziale Elektrotechniki, Automatyki Informatyki. Dyplom magistra inżyniera otrzymał w 2001 roku, stopień doktora nauk technicznych w 2007. Obecnie pracuje jako pracownik naukowo-dydaktyczny na Politechnice Rzeszowskiej w Zakładzie Mechaniki Płynów i Aerodynamiki. Prowadzi badania w dziedzinie modelowania matematycznego i optymalizacji układów pneumatycznych oraz hydraulicznych.

e-mail: kszostek@prz.edu.pl

Streszczenie

W artykule omówiono projekt procesora dedykowanego do realizacji tabeli przejść i wyjść dowolnego automatu sekwencyjnego. Celem budowy było skonstruowanie procesora o jak najprostszej budowie reprezentującego podstawowe cechy procesora oraz zaprojektowanie takiego procesora i uruchomienie w środowisku Matlab/simulink. Założono, że projektowany procesor powinien realizować dowolny automat Moore'a, lub Mealy'ego. Proces projektowania został podzielony na części. Osobno zostały zaprojektowane poszczególne części składowe realizujące przypisane zadania. Wyróżniono następujące części składowe: licznik pamięci, akumulator, moduły wejścia wyjścia, moduł funkcji logicznej NXOR, moduł ustawiania jedynek do akumulatora oraz układ sterujący CU. Poszczególne części procesora są to logiczne układy sekwencyjne i kombinacyjne. Do budowy automatów kombinacyjnych zostały wykorzystane bloki Combinational logic programu Matlab/simulink. W artykule został przedstawiony schemat blokowy przedstawiający wszystkie składowe procesora oraz układ sterujący CU (Control unit), który jest najbardziej złożoną częścią składową procesora.

Słowa kluczowe: procesor sekwencyjny, automat sekwencyjny, funkcje logiczne, matlab simulink.

Sequential processor design and simulation in MATLAB/Simulink environment

Abstract

This paper discusses the design of a processor dedicated to the implementation of the state table and output table of finite-state machines. The aim was to construct a CPU of simplest construction that represents the basic processor features and run it in the Matlab/simulink environment. It was assumed that the processor should be designed to implement any Moore or Mealy machines. The design process is divided into parts. Components performing assigned tasks were individually designed. They are: memory counter, accumulator, input-output blocs, logical NXOR function module, module setting the ones for the accumulator and control unit CU. The different parts of the processor are combination and sequential logic machines. Combinational logic blocks of Matlab/Simulink were used for the construction of the combinatorial machines. In the paper a block diagram showing all components of the processor and the control unit CU is shown. The control unit is the most complex part of the processor.

Keywords: sequential processor, finite-state machine, logic function, matlab simulink.

1. Wstęp

W pracy przedstawiono projekt procesora. Celem pracy budowy było skonstruowanie procesora o jak najprostszej budowie reprezentującego podstawowe cechy procesora oraz zaprojektowanie takiego procesora i uruchomienie w środowisku MATLAB/ simulink.

Procesor jest to urządzenie sekwencyjne, które pobiera dane z pamięci interpretuje je i wykonuje.

Zadanie projektowania procesora realizuje się przez oddzielne projektowanie części składowych przyjętej struktury blokowej. Poszczególne bloki są to logiczne układy sekwencyjne i kombinacyjne realizują one przypisane zadania. Pomiędzy blokami przesyłane są sygnały. Dla projektowanego procesora przyjęto, że sygnały przyjmują dwie wartości 0 1 tak, więc przesyłane sygnały są to sygnały binarne. Układy takie realizują operacje zgodne z algebrą Boole'a dlatego, nazywane są układami logicznymi.

W układach logicznych kombinacyjnych sygnały wyjściowe zależą tylko od sygnałów wejściowych. W układach sekwencyjnych sygnały wejściowe mogą zależeć od sygnałów wejściowych oraz zależą od wartości sygnałów odpowiadających stanom wewnętrznym [1].

Abstrakcyjnymi modelami układów sekwencyjnych są automaty sekwencyjne. Można wyróżnić dwa rodzaje automatów sekwencyjnych są to automat Moore'a i Mealy'ego. W automacie Moore'a wyjścia zależą wyłącznie od stanów automatu natomiast w automacie Mealy'ego wyjścia zalezą zarówno od stanów, jaki i sygnałów wejściowych dzięki temu redukuje się liczbę stanów w porównaniu z automatem Moore'a.

Najbardziej interesujące jest projektowanie układów sekwencyjnych.

Przy projektowaniu automatu sekwencyjnego najczęściej dostępny jest opis działania następnie na podstawie opisu tworzy się pierwotną tablice przejść i wyjść etap ten nazywa się synteza abstrakcyjną. Kolejnymi etapami jest minimalizacja stanów oraz kodowanie stanów automatu w wyniku uzyskuje się minimalna tablice automatu Mealy'ego. Końcowy etapem projektowania automatu sekwencyjnego jest wyznaczenie funkcji logicznych. W przypadku automatu kombinacyjnego jego działanie opisane jest przez podanie wartości sygnałów wejściowych i odpowiadające im wartości sygnałów wyjściowych, dlatego możliwe jest wyznaczenie bezpośrednio minimalnej tablicy Mealy'ego, w której nie ma stanów. Projektowanie takiego układów sprowadza się do wyznaczeniu funkcji logicznych.

Do wyznaczania funkcji logicznych są stosowane metody projektowania takie jak graficzna metoda tablic Karnaugha, czy algorytm Quine'a McCluskeya z usuwaniem nadmiarowych składowych [7].

Założono, że projektowany procesor powinien realizować dowolny automat Moore'a, albo Mealy'ego.

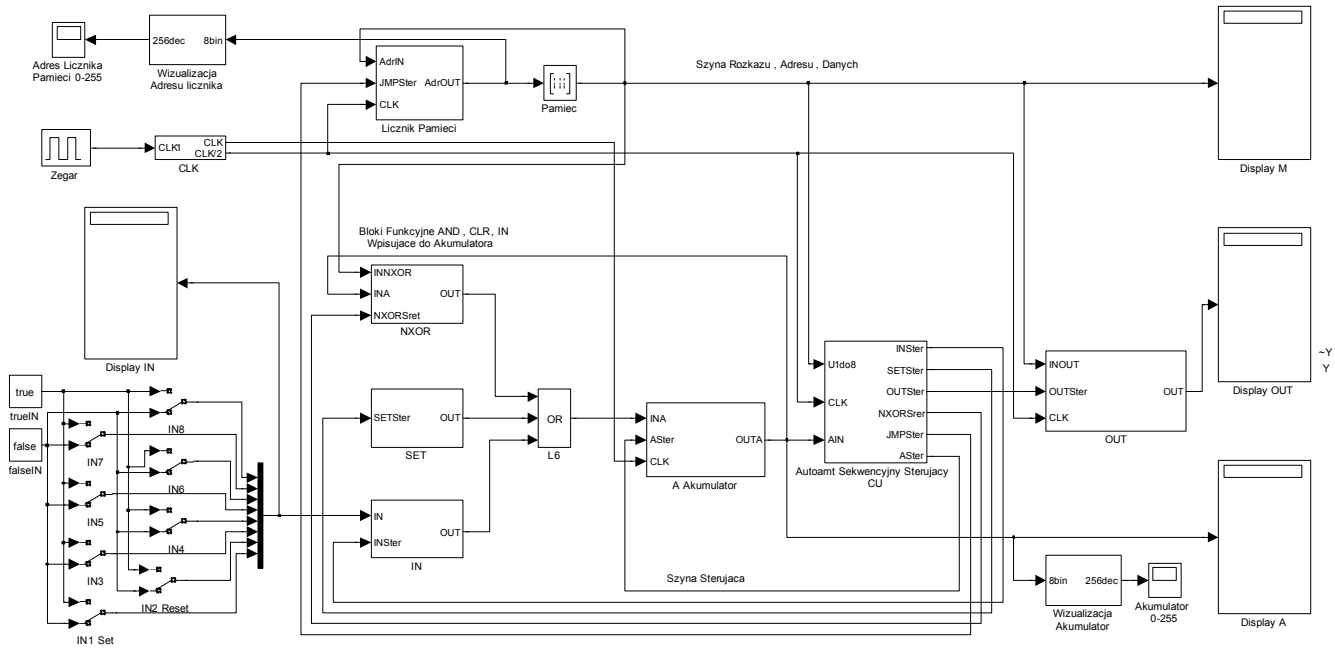
Obecnie dostępne są układy programowalne *FPGA* (Field Programmable Gate Array). Układy te zaczęto budować w latach 2005, 2008. Układy *FPGA* zbudowane są z matrycy bramek *NAND*, o zazwyczaj pięciu wejściach, których połączenie można w dowolny sposób konfigurować. Obecnie rozwiązania te znajdują różne zastosowania [1, 3].

Dzięki dostępności układów *FPGA* możliwe jest bezpośrednie zastosowanie projektowanych struktur logicznych. Językiem wykorzystywanym do programowania układów *FPGA* jest *VHDL*. Program Matlab Simulink posiada koder *HDL* umożliwiający zapis zbudowanych modeli w simulinku w języku *VHDL*. Po zaprojektowaniu procesora kolejnym zadaniem jest wprowadzenie procesora do struktury programowalnej *FPGA*.

2. Założenia

Zaprojektowany procesor będzie miał następujące cechy:

1) Posiadał cechy architektury von Neumanna dane przechowywane są wspólnie z instrukcjami.
2) Pamięć i procesor będą rozdzielone.
3) Będzie posiadał instrukcje skoków warunkowych.
4) Architekture sekwencyjną.



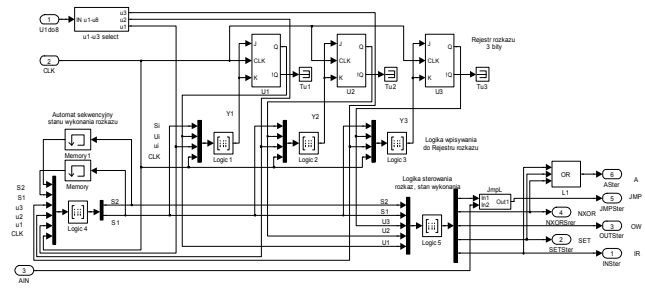
Rys. 1. Schemat procesora w Matlab/simulink
Fig. 1. Schema of the processor in Matlab/simulink

Pamięć procesora 256 komórek ośmiobitowych (adresowanie ośmiobitowe), licznik adresu, oraz jeden akumulator A. Ilość instrukcji została ograniczone do minimum. Przyjęto, że procesor będzie miał następujące instrukcje:

1) *IR* Odczyt z wejścia do *A*, jeden takt, przeskok adresu o jeden.
2) *OW* Zapis na wyjście wartości z pamięci, dwa takty, przeskok adresu o dwa.
3) *NXOR* działanie na wartości zawartej *A* i wartości z pamięci wynik zapisany w *A*, dwa takty, przeskok adresu o dwa.
4) *JMP* Instrukcja skoku, gdy same jedynki *A*, dwa takty, przeskok do adresu lub o jeden dwa.
5) *SET* Ustawianie jedynek w akumulatorze, jeden takt, przeskok adresu o jeden.

Zostały zaprojektowane poszczególne składowe procesora schemat procesora w Mtlab/imulink przedstawia rys. 1. Układy kombinacyjne procesora zostały zrealizowane za pomocą bloków „*Combinational logic*". Składowe procesora to: Licznik pamięci, pamięć zrealizowana za pomocą bloku „*Combinational logic*", układ sterujący z rejestrem rozkazu *CU*, akumulator *A*. W modułach układu wykorzystano synchroniczne przerzutniki JK przełączane zboczem opadającym w liczniku adresu, akumulatorze *A*, dzielniku częstotliwości zegara *CL*K oraz na wyjściu *OUT*. Bloki poszczególnych rozkazów *NXOR*, *SET*, *IN*, są to układy kombinacyjne wpisujące do akumulatora. Blok *NXOR* realizuje negacje bitową sygnałów *INNXOR* oraz *INA* po podaniu na wejście sterując sygnału 1. Blok *SET* po podaniu na wejście sterujące sygnału 1 wystawia na wyjście sygnały binarne jedynek. Blok *IN* przepisuje wejście na wyjście po podaniu jedynki na wejście sterujące. Blok *OUT* jest układem sekwencyjnym z rejestrem na przerzutnikach *JK*. W rejestrach licznika rozkazu, akumulatora oraz modułu wyjścia wartość sygnału podana na wejście jest zapamiętywana na zboczu opadającym sygnału *CLK*. Licznik pamięci jest to układ sekwencyjny posiada trzy wejścia: adres wejściowy *AdrIN*, wejście sterujące *JMPSter*, które jest ustawiane na 1 przy skoku do adresu *AdrIN*. Jeśli na wejściu sterującym *JMPSter* jest sygnał 0 adres zwiększany jest o jeden. Zmiana adresu następuje na zboczu opadającym sygnału *CLK*.

Najbardziej skomplikowanym modułem procesora jest układ sekwencyjny *CU* przedstawiony na rys. 2.



Rys. 2. Schemat procesora w Matlab/simulink
Fig. 2. Schema of the CU (Control Unit) in Matlab/simulink

Układ posiada rejestr rozkazu zbudowany z przerzutników *JK*, układ sekwencyjny stanu wykonania rozkazu oraz dwa układy kombinacyjne tj: układ kombinacyjny wpisywania do rejestru rozkazu, oraz układ kombinacyjny sterowania sygnałami wyjściowymi: *ASter*, *JMPSter*, *NXORSter*, *OUSter*, *SETSter*, *INSter*.

3. Architektura procesora

Przez architekturę procesora rozumiane sposób organizacji połączeń pomiędzy procesorem pamięcią i układami wejścia wyjścia.

Za względu na organizacje połączeń pomiędzy pamięcią a procesorem stosuje się Taksonomia Flynna, która opiera się na liczbie przetwarzanych strumieni danych. W przypadku tej klasyfikacji rozważany procesor to *SISD* (Single Instruction, Single Data) - przetwarzany jest jeden strumień danych przez jeden wykonywany program.

Rozważany procesor ze względu na sposób organizacji pamięci ma architekturę von Neumanna cechą charakterystyczną tej architektury jest to, ze dane przechowywane są razem z instrukcjami.

W układzie został pominięta jednostka arytmetyczna, ponieważ w przypadku realizacji tablicy automatu sekwencyjnego nie jest konieczne wykonywanie działań arytmetycznych.

Procesory klasyfikuje się ze względu na dostęp procesora do pamięci, ale klasyfikacja ta jest stosowana do układów wieloprocesorowych, dlatego rozważany procesor nie podlega tej klasyfikacji. Przykłady projektowania układów współbieżnych można znaleźć w pracy [5], optymalizacji układu w pracy [4] natomiast metod analizy modeli w pracy [2].

4. Programowanie procesora

Poniżej został przedstawiony przykład programu realizującego przerzutnik SR na zaprojektowanym procesorze.

Dla asynchronicznego przerzutnika minimalna tablica przejść i wyjść automatu Mealy'ego została podana poniżej.

Tab. 1. Minimalna tablica przejść i wyjść automatu dla przerzutnika SR
Tab. 1. Minimum state transition and output table for SR flip-flop

Stan	*SR*				*Y*	*~Y*
	00	01	11	10		
0	**0**	**0**	-	1	0	1
1	**1**	0	-	**1**	1	0

Przyjęto, że wejścia procesora pierwszy bit IN_1 odpowiada sygnałowi *Set*, drugi bit IN_2 odpowiada sygnałowi *Reset*. Pierwsze wyjście OUT_1 odpowiada sygnał *Y*, drugie OUT_2 sygnał $\sim Y$ rys. 1. Stąd program napisany w asemblerze dla zaprojektowanego procesora można zapisać.

```
000 OW 00000010        //Stan 0, wstawienie wyjście na 01000000
001 IR                 //Odczyt wejścia do A
002 NXOR 00000001      //Wynik porównania w A
003 JMP 006            //Skok do kroku 006
004 SET                //Wystawianie jedynek do akumulatora
005 JMP 000            //Skok do kroku 000
006 OW 00000001        //Stan 1, wstawienie wyjście na 10000000
007 IR                 //Odczyt wejścia do A
008 NXOR 00000010      //Wynik porównania w A
009 JMP 000            //Skok do kroku 000
010 SET                //Wystawianie jedynek do akumulatora
011 JMP 006            //Skok do kroku 006
```

Na etapie projektowania zostały przyjęte, wartości binarne rozkazów.

```
IR      00000001
SET     00000011
OW      00000010
AND     00000100
JMP     00000101
```

Po zapisaniu programu w kodzie binarnym otrzymano:

```
00000000 00000010      //Stan 0, OW
00000001 00000010
00000010 00000001      //IR
00000011 00000100      //NXOR
00000100 00000001
00000101 00000101      //JMP Skok do stanu 1
00000110 00001010
00000111 00000011      //SET
00001000 00000101      //JMP Skok do stanu 0
00001001 00000000
00001010 00000010      //Stan 1, OW
00001011 00000001
00001100 00000001      //IR Odczyt wejścia do A
00001101 00000100      //NOR
00001110 00000010
00001111 00000101      //JMP Skok do stanu 0
00010000 00000000
00010001 00000011      //SET
00010010 00000101      //JMP Skok do stanu 1
00010011 00001010
```

5. Zapis procesora do struktury FPGA

Przy zapisie zbudowanego modelu do struktury *FPGA* możliwe jest wykorzystanie koder *HDL* programu Matlab/simulink pozwalającego na zapis zbudowanych w simulinku modeli w języku *VHDL*. Możliwe jest również zapisanie poszczególnych składowych modelu w postaci bramek *NAND* i opis układu w języku *VHDL*.

W przypadku zapisu układu w postaci elektronicznych bramek logicznych konieczne jest wyznaczenie funkcji logicznych automatów kombinacyjnych, które w modelu zostały zrealizowane z wykorzystanie bloków „*Combinational logic*". Układ ten będzie odporny na zjawisko hazardów i wyścigów, ponieważ wszystkie rejestry są taktowane. Konieczne jest jedynie zagwarantowanie odpowiedniej częstotliwości taktowania, takiej, przy której sygnały na wyjściach układów kombinacyjnych zdążą przejść do stanu stabilnego w ustalonym przedziale czasu.

Dla omawianego procesora układ kombinacyjny wpisywania rozkazu do rejestry blok *CU* opisany jest tabelą minimalizacji 1. Gdzie S_1 - stan wykonania rozkazu, U_i - stan rejestru rozkazu, u_i - wartość bitu *i* rozkazu *i*=1,2,3, *CLK* – sygnał zegara.

Tab. 2. Minimalizacja funkcji wpisywania adresu do rejestru *CU*
Tab. 2. Minimization of the function entering the address in the register of *CU*

S_1,*CLK*	u_i, U_i			
	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	0	0	0	0
10	0	0	0	0

Stąd funkcja logiczna ma postać jak w równaniu 1.

$$Y_i = \overline{S_1} \wedge CLK \wedge (\overline{U_i} u_i \vee U_i \overline{u_i}), \tag{1}$$

6. Podsumowanie i Wnioski

W artykule omówiono projekt procesora sekwencyjnego zrealizowanego w środowisku Matlab/simulink. Procesor został zaprojektowany do realizacji tablicy dowolnego automatu sekwencyjny. Został omówiony przykładowy programu realizujący tablice automatu Mealy'ego.

Został określony minimalny zbiór instrukcji pozwalający zrealizować tablice automatu sekwencyjnego.

W przyszłej pracy zadaniem będzie wprowadzenie zrealizowanego procesora do struktury programowalnej *FPGA*.

Program MATLAB wykorzystany do przeprowadzenia badań został zakupiony w wyniku realizacji Projektu nr UDA-RPPK.01.03.00-18-003/10-00 „Budowa, rozbudowa i modernizacja bazy naukowo-badawczej Politechniki Rzeszowskiej" współfinansowanego ze środków Unii Europejskiej w ramach Regionalnego Programu Operacyjnego Województwa Podkarpackiego na lata 2007-2013, Priorytet I. Konkurencyjna i Innowacyjna Gospodarka, Działanie 1.3 Regionalny system innowacji.

7. Literatura

[1] Clemente J.A., González C., Resano J., and Mozos. D.: A task graph execution manager for reconfigurable multi-tasking systems. Microprocess. Microsyst., vol. 34, no. 2–4, s. 73–83, Mar. 2010.
[2] Gamracki M.: Modelowanie matematyczne zjawiska wnikania pola elektromagnetycznego do stratnej ziemi, Przegląd Elektrotechniczny, ISSN 0033-2097, NR 3/2010, str. 109-111.
[3] Null L., Lobur J.: The essentials of computer organization and architecture. Jones & Bartlett Learning, s. 36, 199–203, 2010.
[4] Szostek R.: Zastosowanie teorii kolejek do modelowania procesów zachodzących w urządzeniach liczących. Elektrotechnika i Elektronika, Kraków, t. 18, z. 3, s. 81-88, 1999.
[5] Szostek R.: Modelowanie systemów współbieżnych za pomocą sieci kolejkowych. Zarządzanie i Marketing, Rzeszów, z. 11, nr 245, s. 153-165, 2007.
[6] Węsierski Ł.: Podstawy logiki i wnioskowania. Oficyna Wydawnicza Politechniki Rzeszowskiej, 2004.
[7] Żelazny M.: Podstawy automatyki. PWN 1976.

otrzymano / received: 23.07.2013
przyjęto do druku / accepted: 02.09.2013 *artykuł recenzowany / revised paper*