Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

ZASTOSOWANIE RELACYJNYCH BAZ DANYCH W PROJEKTOWANIU I SYMULACJI STEROWNIKÓW LOGICZNYCH

Małgorzata Kołopieńczyk

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

e-mail: m.kolopienczyk@iie.uz.zgora.pl

STRESZCZENIE

Celem niniejszego artykułu jest przedstawienie metody opisu układu sterowania logicznego przy wykorzystaniu mechanizmów relacyjnej bazy danych. Jako formalny język specyfikacji układu wybrano sieć Petriego, jako język dobrze sformalizowany i wydajny przy automatyzacji przetwarzania. Przeniesienie sieci Petriego do relacyjnej bazy danych to nie tylko efektywne przeniesienie opisu układu do systemu komputerowego. Zastosowanie standardowego języka zapytań SQL pozwala na przeprowadzenie symulacji układu bezpośrednio w bazie danych.

1. UZASADNIENIE TEMATU

Ciągły postęp w elektronice pozwala na implementację coraz to bardziej rozbudowanych sterowników logicznych PLC (Programmable Logic Controllers). Równolegle z rosnącymi możliwościami technologicznymi poszukiwane są nowe metody projektowania takich układów. Wśród nowoczesnych technik projektowania sterowników PLC wymienić można język Grafcet [5] i języki wyspecyfikowane w międzynarodowej normie IEC:1131-3 [4].

W celu weryfikacji poprawności oraz w celu zbadania własności projektowanego układu stosowane są formalne metody opisu układu. Najczęściej stosowanym formalizmem na etapie weryfikacji są sieci Petriego [6, 3]. Sieć Petriego stanowi uogólnioną, pośrednią formę w projektowaniu i badaniu właściwości sterowników. Istnieją techniki łatwej i szybkiej translacji innych języków projektowania sterowników do sieci Petriego [6]. Dostępnych jest wiele programów komputerowych wspierających projektowanie sterowników logicznych i ich weryfikację przy wykorzystaniu sieci Petriego oraz bardzo im bliskich języków Grafcet i SFC. Wymienić tu można m.in. aplikacje MX-SCADA, ISaGRAF, PC WORX.

Istotnym utrudnieniem w integracji różnych aplikacji jest sposób opisu projektowanego układu w strukturach danych programów. Brak jest obecnie spójnych i ogólnie uznanych metod opisu projektowanego układu, w tym samej sieci. Istnieją co prawda pewne formaty takie jak PNSF, PNSF2 oraz zbliżone do nich opisy oparte o język XML. Jednak stanowią one wyłącznie formę pośrednią pozwalającą na przeniesienie opisu sieci z jednego narzędzia do drugiego. Formaty takie nie pozwalają na bezpośrednią analizę układu w jej bieżącej

postaci, nie pozwalają na badanie właściwości dynamicznych, są mało czytelne i trudne do bezpośredniego przetwarzania przez programy komputerowe. Analiza i symulacja zaprojektowanego układu wciąż odbywa się przez dedykowane aplikacje, które przechowują opis sieci w swych wewnętrznych strukturach danych.

W artykule przedstawiono sposób zamodelowania układu logicznego w języku sieci Petriego przy wykorzystaniu mechanizmów relacyjnej bazy danych. Relacyjne bazy danych to obecnie najpopularniejszy sposób gromadzenia danych dla aplikacji [7, 9]. Sformatowana postać danych oraz jasno sprecyzowane relacje umożliwiają przejrzyste opisanie nawet skomplikowanych struktur, zezwalając jednocześnie na automatyczne przetwarzanie danych przez programy komputerowe. Wykorzystanie baz danych pozwala także na zwiększenie wydajności przetwarzania – wysoka skalowalność systemów zarządzania bazami danych RDBMS (Relational Database Management Systems) pozwala na zakodowanie nawet bardzo złożonych układów; techniki indeksowania i partycjonowania pozwalają na szybkie przetwarzanie ogromnych ilości danych, dając nieporównywalnie większą wydajność w stosunku do przetwarzania standardowych plików komputerowych oraz niwelując ograniczenia związane z koniecznością zapamiętania całej struktury układu w pamięci operacyjnej.

Zapisanie układu sterowania w relacyjnej bazie danych pozwala ponadto na symulację projektowania układu bezpośrednio przy wykorzystaniu standardowych języków zapytań, w tym przede wszystkim języka SQL [8]. Daje to możliwość badania własności sterownika bezpośrednio przez projektanta stosującego standardowy język zapytań i mającego bezpośredni wpływ na operacje wykonywane na modelu.

2. SIEĆ PETRIEGO

Sieci Petriego [2, 3] umożliwiają badanie zjawisk współbieżnych zachodzących w systemach wzajemnie warunkujących się w czasie i przestrzeni warunków i zdarzeń. Oprócz projektowania i programowania sterowników logicznych, znajdują one zastosowanie także w różnych gałęziach przemysłu, w zadaniach planowania i sterowania przepływem produkcji, w syntezie oprogramowania systemowego itp.

W sieci Petriego zdarzenia i warunki reprezentowane są abstrakcyjnymi symbolami należącymi do dwóch rozdzielnych alfabetów (zbiorów). Zdarzenia odpowiadają przejściom (tranzycjom), warunki miejscom, a związek przeyczynowo-skutkowy między zdarzeniami i warunkami oraz warunkami i zdarzeniami za pomocą relacji przepływu.

Graficznie sieć Petriego reprezentowana jest jako struktura złożona z prostokątów i kółek połączonych łukami (strzałkami). Prostokąty reprezentują przejścia, kółka – miejsca, a łuki relację przepływu między miejscami i przejściami oraz przejściami i miejscami. Miejsca, z których prowadzą łuki z rozpatrywanego przejścia nazywane są miejscami wyjściowymi przejścia. Analogicznie określane są miejsca wejściowe przejścia oraz przejścia wejściowe i

wyjściowe dla miejsc. Miejsca wejściowe przejścia przedstawiają warunki – przyczyny zdarzenia określonego tym przejściem. Miejsca wyjściowe przejścia reprezentują warunki – skutki zdarzenia określonego tym przejściem. Wystąpienie (utrzymanie się) pewnego warunku przedstawione jest znakiem (znacznik, marker) wewnątrz miejsca odpowiadającego temu warunkowi.

Dynamika działania modelowanego systemu dyskretnego znajduje odzwierciedlenie w przesuwaniu się znaczników pomiędzy miejscami sieci. Lokalne zdarzenia i związane z nimi akcje przedstawiane są realizacją (zadziałaniem, zapaleniem) określonego przejścia. Przejście może być zrealizowane (wystąpić), gdy wszystkie jego miejsca wejściowe zawierają co najmniej po jednym znaku. W kategoriach zdarzeń i warunków potencjalna realizacja przejścia odpowiada spełnieniu warunków zajścia określonego zdarzenia (wzbudzenie przejścia). Realizacja przejścia modeluje wystąpienie określonego zdarzenia. Realizacji przejścia towarzyszy usunięcie po jednym znaku miejsc wejściowych przejścia i dodanie po jednym znaku do miejsc wyjściowych przejścia. W ten sposób wskazuje się nowe warunki – skutki wystąpienia rozpatrywanego zdarzenia.

Formalnie sieć Petriego definiujemy jako uporządkowaną trójkę PN = (P, T, F) gdzie:
1) $P \cap T = \varnothing$,
2) $P \cup T \neq \varnothing$,
3) $F \subseteq (P \times T) \cup (T \times P)$,
4) $\mathrm{dom}(F) \cup \mathrm{cod}(F) = P \cup T$.

F jest relacją przepływu w sieci PN, $\mathrm{dom}(F)$ jest dziedziną relacji F, czyli $\mathrm{dom}(F) = \{x: \underset{y}{\forall} (x, y) \in F\}$, $\mathrm{cod}(F)$ jeset przeciwdziedziną relacji F, czyli $\mathrm{dom}(F) = \{y: \underset{x}{\forall} (x, y) \in F\}$. Elementy zbioru F nazywane są łukami, elementy zbioru P nazywane są miejscami (P-elementami), elementy zbioru T nazywane są tranzycjami (T-elementami).

Znakowana sieć Petriego to uporządkowana szóstka $MPN = (P, T, F, K, W, M_0)$ gdzie:

1) (P, T, F) jest siecią Petriego,
2) $K: P \to N^+ \cup \{\infty\}$ jest funkcją pojemności miejsc,
3) $W: F \to N^+$ jest funkcją wagi łuków,
4) $M_0: P \to N$ jest funkcją znakowania początkowego spełniającą warunek $\underset{p \in P}{\forall} M_0(p) \leq K(p)$.

Za pomocą znakowań (markowań) opisujemy stan w jakim znajduje się układ, czyli jakie miejsca są w danej chwili czasowej aktywne. Markowania zmieniają się w czasie, pokazując które miejsca zostały „wykonane" i w jakiej kolejności. Formalnie opiszemy to za pomocą realizacji tranzycji:

Niech $MPN = (P, T, F, K, W, M_0)$ będzie znakowaną siecią Petriego, wówczas:

1) funkcja $M: P \to N$ jest nazywana znakowaniem $MPN \Leftrightarrow \underset{p \in P}{\forall} M(p) \leq K(p)$,
2) tranzycja $t \in T$ jest przygotowana do realizacji (dozwolona) w oznakowaniu $M \Leftrightarrow \underset{p \in P}{\forall} W(p, t) \leq M(p) \leq K(p) - W(t, p)$,

3) jeżeli tranzycja $t \in T$ jest tranzycją, która jest dozwolona w oznakowaniu M to t może wystąpić i zostać zrealizowana. Nowe znakowanie określone $M'(p) = M(p) - W(p, t) + W(t, p)$ dla wszystkich $p \in P$,

4) wystąpienie tranzycji $t \in T$ zmienia znakowanie M na znakowanie M', co oznacza się poprzez $M\,[t\rangle M'$ lub $M \xrightarrow{t} M'$.

W modelowaniu sterowników logicznych stosowane są sieci 1-ograniczne, czyli takie w których pojemność miejsc wynosi 1. Definicja sieci n-ograniczonej jest następująca:

1) miejsce $p \in P$ jest n-ograniczone ($n \in \mathrm{N}$) $\Leftrightarrow \underset{M \in [M_0\rangle}{\forall} M(p) \le n$,

2) *MPN* jest n-ograniczona ($n \in \mathrm{N}$) $\Leftrightarrow \underset{p \in P}{\forall}$ p jest n-ograniczone.

3. STRUKTURA TABLIC W RELACYJNEJ BAZIE DANYCH

Sieć Petriego opisująca układ sterowania logicznego jest siecią 1-ograniczoną. Podstawowe elementy sieci to miejsca i tranzycje. W relacyjnej bazie danych do opisu miejsc i tranzycji wykorzystamy dwie tablice, o nazwach `STATE` i `TRANSITION`. `STATE` to lista stanów (miejsc, kroków) układu, `TRANSITION` to lista tranzycji układu.

Relacje zachodzące pomiędzy miejscami i tranzycjami opisane zostaną w tablicy `RELATION`. W tablicy tej dla jednej tranzycji może wystąpić kilka rekordów. Zbiór tranzycji z prawdziwymi warunkami przejścia opisano za pomocą kolejnej tablicy `TRANSACTIONACTIVE`. Tablica opisuje, które tranzycje są aktywne, czyli które z nich mają prawdziwe warunki i pozwalają na aktywację miejsca. Wartości w tej tablicy powinny być ustawiane na podstawie warunków przy tranzycjach, jednak ten element systemu nie jest jeszcze opracowany. Wprowadzamy więc do tej tablicy wartość true/false, w zależności od tego czy dany warunek ma być prawdziwy czy fałszywy. Aktywne stany układu (czyli stany ze znacznikiem) opisane są w tablicy `STATEACTIVE`. Tablica informuje, które miejsca są aktualnie aktywne. W tablicy tej mamy więc wyjście układu. Miejsca aktywne zmieniają się w miarę wykonywania kolejnych kroków symulacji w zależności od warunków na tranzycjach, z zachowaniem współbieżności i synchronizacji.

4. SYMULACJA UKŁADU PRZY WYKORZYSTANIU JĘZYKA SQL

Symulacja składa się z dwóch podstawowych kroków: inicjalizacji i wykonania kolejnych kroków symulacji. Inicjalizacja (restart) tworzy tablice pomocnicze, które wykorzystywane są dalej w trakcie symulacji układu. Wykonanie kroku (step) sprawdza warunki na tranzycjach i odpowiednio aktywuje kolejne miejsca układu, symulując jego działanie.

Restart to inicjalizacja automatu, wygenerowanie tablic pomocniczych wykorzystywanych w trakcie symulacji. Obecnie w momencie inicjalizacji jako aktywne ustawiane jest miejsce o numerze 1, oczywiście łatwo rozszerzyć system o możliwość wyboru miejsca startowego. W

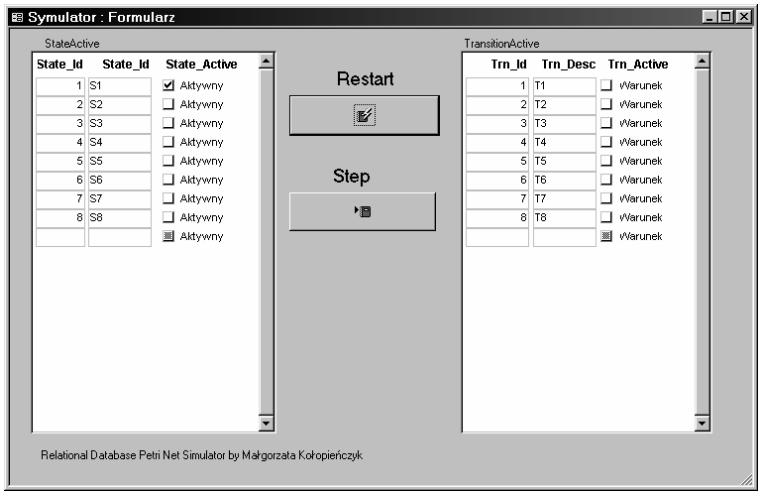
trakcie inicjalizacji następuje wyzerowanie miejsc i tranzycji „aktywnych”. Tabela `STATECOUNT` jest tablicą pomocniczą w której przechowywana jest ilość miejsc wejściowych dla każdej tranzycji.

Step - wykonanie kolejnego kroku symulacji automatu. W trakcie takiego kroku sprawdzane są warunki tranzycji oraz odpalane są te miejsca, dla których tranzycje wyjściowe są aktywne. Spradzane są przy tym warunki współbieżności automatu.

Tabela `STATECOUNTACTIVE` jest tablicą pomocniczą w której przechowywana jest ilość miejsc wejściowych aktywnych dla każdej tranzycji. Jak łatwo zauważyć, kroki 6 i 7 mogłyby zostać zapisane w jednej instrukcji SQL wykorzystującej polecenie `OUTER JOIN`. Jednak system testowany był na bazie danych MS Access, w której `OUTER JOIN` nie jest zaimplementowany.

5. PRZYKŁAD SYMULACJI UKŁADU

W celu weryfikacji modelu wykonana została prototypowa aplikacja pozwalająca na symulację układów opisanych za pomocą sieci Petriego. Jak system zarządzania bazami danych wybrano popularną aplikację Microsoft Access w wersji 7.0. Cała symulacja odbywa się poprzez wywołanie odpowiednich skryptów SQL, tak więc do symulacji układu nie są potrzebne żadne inne mechanizmy oprócz tych, jakie udostępnia język SQL. W celu lepszej prezentacji wyników przygotowana została prosta formatka do prezentacji danych, której wygląd przedstawiono na rys. 1.



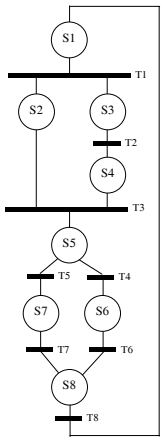
Rys. 1. Ekran symulatora

W celu demonstracji działania systemu przeprowadzono symulację prostego układu którego schemat zamieszczono na rys. 2.

Jak widać w układzie mamy sekwencję współbieżną oraz sekwencję dywergentną, pozwala to zbadać zachowanie modelu w przypadku współbieżnego wykonania kilku zadań, synchronizacji współbieżnych sekwencji oraz alternatywny wybór następnej procedury sekwencyjnej.

Model układu po przeniesieniu do relacyjnej bazy. Ze względu na ograniczoną objętość publikacji, szczegóły translacji pominięto w artykule.

Symulacja układu polegała na zmianie warunków i odpalaniu kolejnych tranzcyji. Testy wykazały poprawną symulację układu. Prawidłowo zasymulowana została współbieżność i synchronizacja układu, jak również alternatywne wybranie jednej z dwóch sekwencji.



Rys. 2. Schemat przykładowego układu

6. PODSUMOWANIE

Możliwe jest zamodelowaniu sterownika opisanego siecią Petriego za pomocą relacyjnej bazy danych. Istnieje możliwość symulacji sterownika przy pomocy języka SQL. W dalszych pracach planowane jest zbadanie możliwości weryfikacji układu oraz badania jego własności jakościowych i ilościowych przy wykorzystaniu SQL i innych mechanizmów relacyjnej bazy danych. W chwili obecnej system nie jest wystarczająco odporny na pewne nieprawidłowości w symulowanym układzie.

Kolejnym zagadnieniem wartym dogłębniejszej analizy jest zbadanie możliwości wykorzystania systemu do symulacji sieci Peteriego n-ograniczonych i nieograniczonych a także bezpośredniej symulacji układów opisanych za pomocą np. języków SFC i Grafcet.

LITERATURA

[1] M. Adamski, M. Chodań: *Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC*, Wydawnictwo PZ, 2000
[2] P. H. Starke: *Sieci Petri*, WNT, 1987
[3] W. Reisig: *Sieci Petri*, WNT, 1988
[4] International Electrotechnical Commision: *International Standard IEC1131-3*, 1993
[5] R. David, H. Alla: *Petri Nets & Grafcet. Tools for modelling discrete event systems*, Prentice Hall, 1992
[6] M. Kołopieńczyk: Modelling Process of Programmable Logic Controllers, Proc. Modelling and Symulation of Systems, 2002
[7] R. Muller: *Bazy danych – język UML w modelowaniu danych*, Mikom, 2000
[8] American National Standards Institute: *Databas Language-SQL ANSI X3.135*, 1992
[9] T. J. Teorey: *Database Modelling and Design*, Morgan Kaufmann, 1999