УДК 681.325

Н.В. Белова[1], Д.А. Петросов[2], В.Ю. Цуканов[3], И.В. Чалый[3]

[1]*Харьковский национальный университет радиоэлектроники, Харьков*
[2]*Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков*
[3]*Харьковский национальный технический университет сельского хозяйства, Харьков*

ИСПОЛЬЗОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ НА БАЗЕ СЕТЕЙ ПЕТРИ ПРИ ПРОЕКТИРОВАНИИ ПЕРЕСТРАИВАЕМОГО ПРОЦЕССОРНОГО МОДУЛЯ

Проведены формализация и алгоритмизация процесса проектирования перестраиваемого процессорного модуля путем формирования решения с изменяющейся структурой и набором компонентов с использованием генетических алгоритмов на основе сетей Петри.

генетический алгоритм, сети Петри, перестраиваемый процессорный модуль

Введение

Задача проектирования перестраиваемого процессорного модуля (ПМ) является разновидностью более общей научной и практической проблемы: проектирование системы с заданным поведением, которая преобразует входные данные в необходимые выходные данные. В качестве данных может выступать как информация, так и нечто материальное (вещество, энергия и т.п.). Сегодня, в частности, эта проблема актуальна при проектировании информационных и организационных систем [1 – 3].

В работе [4] рассмотрены вопросы построения ПМ, реализующего режимы полноразрядного функционального преобразования входных слов и унитарного (инкрементного) преобразования. Проведен структурно-функциональный анализ составляющих компонентов и синтез общей функциональной структуры ПМ. Осуществлен выбор технических модулей для моделирования работы ПМ на программном уровне и в среде VHDL. Концепция построения структур функционально ориентированных вычислительных устройств рассмотрена в работе [5]. Модель перепрограммируемого процессора с гибкой архитектурой обсуждалась в работе [6].

В работе [7] рассматривалась проблема формализации и алгоритмизации процесса генерации вариантов проектных решений в САПР компьютерной техники (КТ). Для решения этой проблемы предлагалось использовать генетические алгоритмы (ГА), которые являются одним из направлений исследования в области искусственного интеллекта [8 – 10].

ГА являются стратегическим подходом к решению проблемы, который необходимо адаптировать к конкретной предметной области. В САПР КТ особое внимание уделяется методам моделирования, одним из которых являются сети Петри (СП) [11]. Поэтому в работе [7] для представления ГА предлагалось использовать СП. Актуальность и перспективность такого подхода подтверждается современными научными исследованиями [12 – 13].

Основным результатом работ [14 – 15] является формализация и алгоритмизация процесса проектирования объектов КТ путем формирования решения с фиксированной структурой из заданных компонентов. Хотя эта задача и принадлежит к простейшему классу задач проектирования КТ, однако ее решение существенно осложняется огромным количеством возможных вариантов реализации, из которых необходимо выбрать один, удовлетворяющий заданному условию. Для преодоления этой проблемы были использованы ГА, адаптированные к решению поставленной задачи путем использования двухуровневых СП.

Целью данной работы является решение более сложной задачи формализации и алгоритмизации процесса проектирования перестраиваемого ПМ путем формирования решения с изменяющейся структурой и набором компонентов с использованием ГА на основе СП.

Постановка задачи. Рассмотрим следующий класс задач проектирования КТ.

$$\text{Дано: } S = \left\langle \text{In}, \text{Out}, \{S_k\}_{k=1}^{K}, \{f_a\}_{a=1}^{A}, \{F_b\}_{b=1}^{B} \right\rangle,$$

где S – система, которую необходимо спроектировать; In – множество входных данных системы S; Out – множество выходных данных системы S; S_k – k-ая подсистема системы S; f_a – функция, определяющая, какие выходные данные соответствуют входным данным: $f_a : \text{In} \to \text{Out}$; F_b – бинарное отношение на множестве $\{S_k\}_{k=1}^{K}$:

$$F_b \subset \{S_k\}_{k=1}^{K} \times \{S_k\}_{k=1}^{K}.$$

Требуется: для заданной функции f_{a_0} подобрать бинарное отношение F_{b_0} такое, чтобы множество подсистем $\{S_k\}_{k=1}^{K}$ обеспечивало обработку системой S входных данных в выходные данные в соответствии с функцией f_{a_0}.

1. Формализация сетями Петри

Входы системы S будут моделироваться множеством позиций $P_{in} = \{p_m^{in}\}_{m=1}^{M}$, где M – количество входов системы, а ее выходы – множеством позиций $P_{out} = \{p_n^{out}\}_{n=1}^{N}$, где N – количество выходов системы (рис. 1).

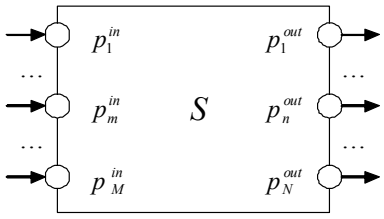

Рис. 1. Контекстная модель системы

Входы и выходы подсистем системы S также будут моделироваться множеством позиций. Входы подсистемы S_k – это множество $P_k^{in} = \{p_{k,m}^{in}\}_{m=1}^{M(k)}$, где $M(k)$ – количество входов подсистемы S_k, а ее выходы – множество $P_k^{out} = \{p_{k,n}^{out}\}_{n=1}^{N(k)}$, где $N(k)$ – количество выходов подсистемы S_k (рис. 2).

Подсистемы $\{S_k\}_{k=1}^{K}$ могут быть связаны как между собой, так и с входами и выходами системы S. Эти связи будут моделироваться множеством переходов $T = \{t_q\}_{q=1}^{Q}$, где Q – количество переходов. Каждый переход t_q характеризуется своими входными и выходными позициями.

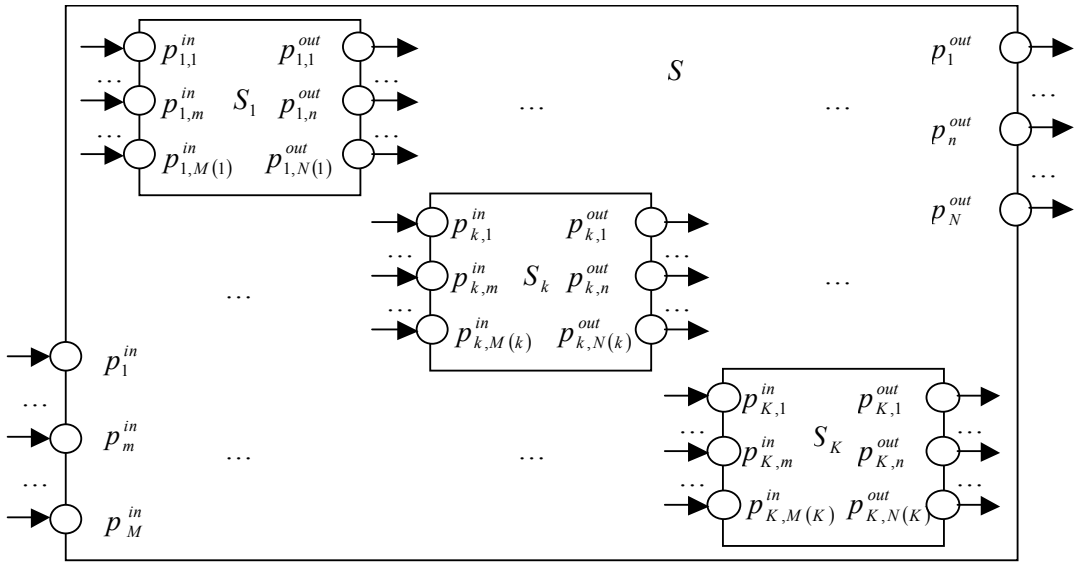

Рис. 2. Множество подсистем системы S

Входами перехода t_q могут быть как любые входы системы S, так и любые выходы подсистем $\{S_k\}_{k=1}^{K}$. Выходами перехода t_q могут быть как любые выходы системы S, так и любые входы подсистем $\{S_k\}_{k=1}^{K}$. Обозначим входы перехода t_q через In_q, а выходы – через Out_q. Тогда:

$$In_q \subset P_{in} \bigcup \left(\bigcup_{k=1}^{K} P_k^{out} \right); \quad Out_q \subset P_{out} \bigcup \left(\bigcup_{k=1}^{K} P_k^{in} \right).$$

Естественно, что для каждого входа системы S должен существовать переход t_q, соединенный с этим входом. Кроме этого, для каждого выхода каждой подсистемы S_k должен существовать переход t_q, соединенный с этим выходом. Формально эти требования можно записать в виде равенства

$$\bigcup_{q=1}^{Q} In_q = P_{in} \bigcup \left(\bigcup_{k=1}^{K} P_k^{out} \right).$$

Аналогично, для каждого выхода системы S должен существовать переход t_q, соединенный с этим выходом, и для каждого входа каждой подсистемы S_k должен существовать переход t_q, соединенный с этим входом. Формально эти требования можно записать в виде равенства

$$\bigcup_{q=1}^{Q} Out_q = P_{out} \bigcup \left(\bigcup_{k=1}^{K} P_k^{in} \right).$$

Разные переходы могут иметь некоторые общие входные и выходные позиции, но полностью совпадающих переходов быть не должно. Это требование формально записываются следующим образом:

$In_{q_1} = In_{q_2} \Rightarrow Out_{q_1} \neq Out_{q_2}$ (при $q_1 \neq q_2$);

$Out_{q_1} = Out_{q_2} \Rightarrow In_{q_1} \neq In_{q_2}$ (при $q_1 \neq q_2$).

Множество подсистем $\{S_k\}_{k=1}^{K}$ вместе с подмножеством множества переходов $T = \{t_q\}_{q=1}^{Q}$ полностью определяют текущую структуру системы S.

Не обязательно все подсистемы $\{S_k\}_{k=1}^{K}$ должны быть задействованы в текущей структуре. При этом перестройка структуры системы S определяется изменением подмножества задействованных в данный момент переходов. Пример структуры системы показан на рис. 3.

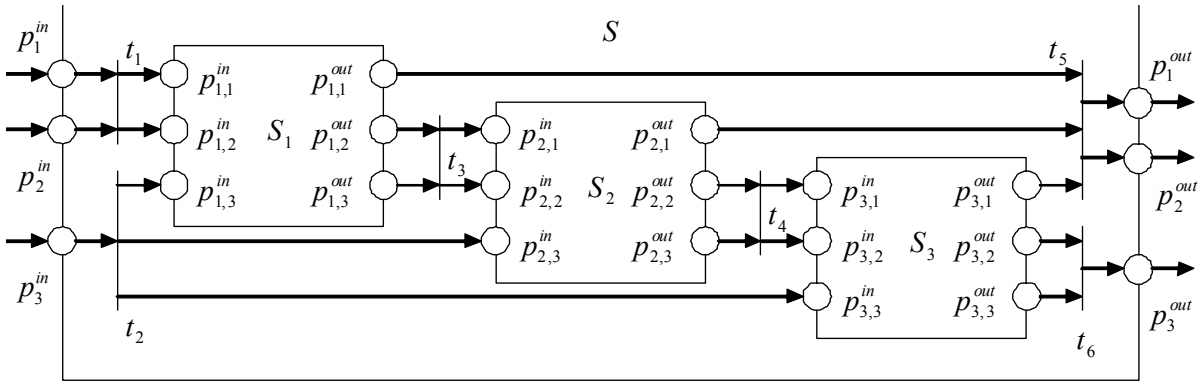

Рис. 3. Пример структуры системы

Перестройка функций (изменение алгоритма функционирования системы S) происходит в подсистемах $\{S_k\}_{k=1}^{K}$. В общем случае каждой подсистеме S_k можно сопоставить множество сетей Петри, которые будут моделями программ обработки данных. Обозначим это множество через PN_k и определим его следующим образом $PN_k = \{PN_{k,r}\}_{r=1}^{R(k)}$, где $PN_{k,r}$ – r-й алгоритм обработки данных подсистемой S_k, представленный в виде сети Петри (рис. 4).

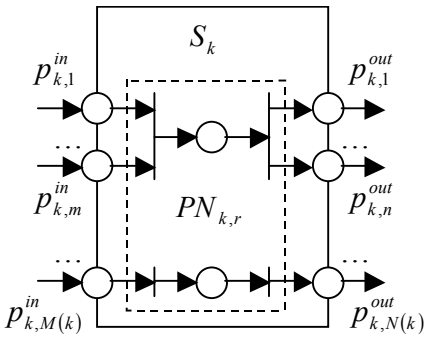

Рис. 4. Представление алгоритма обработки сетью Петри

Формально необходимо для каждой сети Петри $PN_{k,r}$ описать ее позиции, переходы и дуги. Но в общем случае сеть $PN_{k,r}$ моделирует некоторое действие по преобразованию входных данных в выходные данные. Поэтому далее для простоты в качестве моделей алгоритмов будут рассматриваться не сети Петри $PN_{k,r}$, а переходы («вырожденные» сети Петри). Таким образом, множество PN_k будет выглядеть так $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$. Входами $In_{k,r}$ каждого перехода $t_{k,r}$ могут любые входы подсистемы S_k: $In_{k,r} \subset P_k^{in}$, а выходами $Out_{k,r}$ каждого перехода $t_{k,r}$ могут любые выходы подсистемы S_k: $Out_{k,r} \subset P_k^{out}$ (рис. 5).

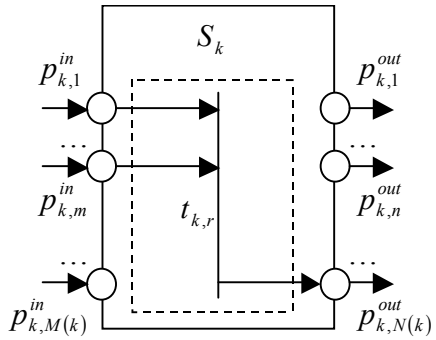

Рис. 5. Представление алгоритма обработки переходом

Исходя из всего вышесказанного, модель системы S в виде сети Петри может выглядеть, например, так, как показано на рис. 6. В этой модели полужирным выделен путь, по которому входные данные, поступив на верхние два входа системы S (на рисунке в виде двух меток), проходят через ее подсистемы и появляются на нижнем выходе системы S. Пунктиром выделены незадействованные переходы и дуги.

2. Описание генотипа

ГА необходимо адаптировать к конкретной предметной области путем задания параметров и определения операторов ГА. При использовании ГА альтернативные решения представляются в виде строки символов, имеющей фиксированную длину и называющейся генотипом. Поэтому особое внимание следует уделить формальному описанию генотипа.

В нашем случае генотип G должен определять:

– какие переходы из множества $T = \{t_q\}_{q=1}^{Q}$ будут представлены в модели системы S;

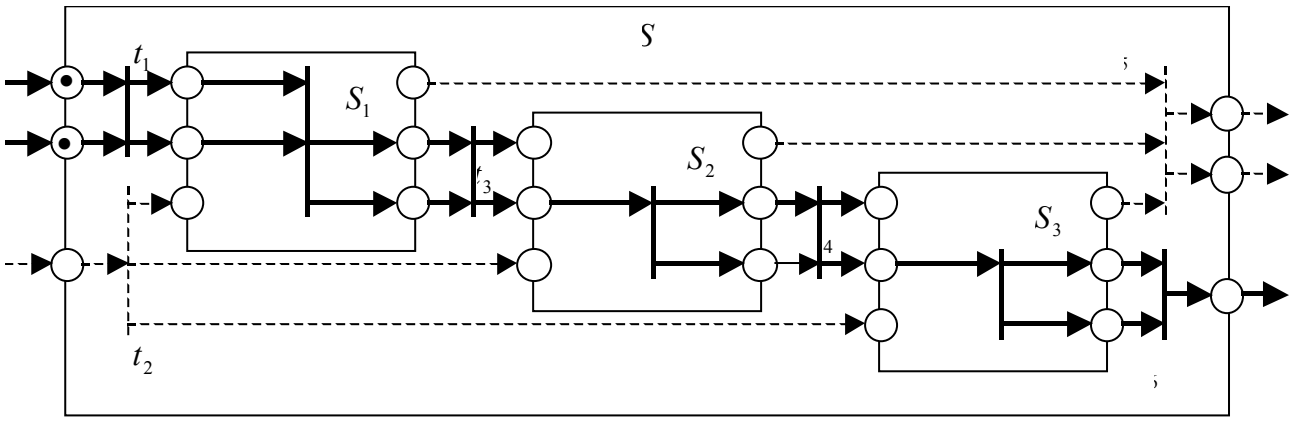

Рис. 6. Пример модели системы S

– какие переходы из множества $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$ будут представлены в модели каждой подсистемы S_k.

Таким образом, генотип G будет иметь следующий вид:

$$G = (g_1,\ldots,g_q,\ldots g_Q, h_1,\ldots,h_k,\ldots,h_K),$$

где $g_q \in \{0,1\}$ и $h_k \in \{0,1,\ldots,r,\ldots,R(k)\}$.

Если $g_q = 0$, то переход t_q отсутствует в модели системы S, иначе ($g_q = 1$) переход t_q присутствует в модели системы S. Если $h_k = 0$, то в модели подсистемы S_k отсутствует алгоритм обработки данных (не загружена программа). Если $h_k = r$ ($r \in \{1,\ldots,R(k)\}$), то в модели подсистемы S_k представлен переход $t_{k,r}$.

Таким образом, число C всех гипотетически возможных моделей системы S определяется следующей формулой:

$$C = 2^Q \prod_{k=1}^{K} (R(k)+1).$$

Поэтому даже при небольшом количестве связей, подсистем и алгоритмов обработки данных количество моделей системы S очень велико. Например, при $Q = 10$, $K = 10$ и R(1)=R(2)=…=R(10)=10 получаем $C = 2^{10} \times 11^{10} = 22^{10} = 26\,559\,922\,791\,424$.

3. Целевая функция

Среди огромного количества всех гипотетически возможных моделей системы S необходимо найти именно ту, которая решает поставленную задачу: из имеющегося множества подсистем $\{S_k\}_{k=1}^{K}$ и системы переходов $T = \{t_q\}_{q=1}^{Q}$ построить такую систему S, которая на входной вектор реагировала бы требуемым выходным вектором. Поэтому чем ближе выходной вектор к требуемому, тем построенная система лучше.

Для оценки этой близости введем обозначение для требуемого вектора: $V = (v_1,\ldots,v_n,\ldots v_N)$, где v_n – количество меток, которые должны находиться в позиции p_n^{out} системы S. Естественно, что $v_n \in \{0,1,2,3,\ldots\}$. Вектор, который получился в результате работы системы S, обозначим через $W = (w_1,\ldots,w_n,\ldots w_N)$, где w_n – количество меток, которые реально находятся в позиции p_n^{out} системы S. Аналогично, $w_n \in \{0,1,2,3,\ldots\}$.

Разницу (расстояние) между требуемым и реальным вектором можно оценивать, например, по формуле $\rho(V,W) = \sum_{n=1}^{N} |v_n - w_n|$.

Выбор целевой функции влияет на эффективность работы генетического алгоритма. Поэтому на практике можно использовать и формулу, задающую классическое евклидово расстояние

$$\rho(V,W) = \sqrt{\sum_{n=1}^{N} (v_n - w_n)^2},$$

и редко используемую формулу

$$\rho(V,W) = \max_{1 \le n \le N} |v_n - w_n|.$$

При программной реализации генетического алгоритма имеет смысл заложить в систему возможность работы с формулой

$$\rho(V,W) = \left(\sum_{n=1}^{N} |v_n - w_n|^p \right)^{1/p},$$

которая при $p = 1$, $p = 2$ и $p \to \infty$ представляет собой рассмотренные выше формулы [16]. Изменяя параметр p, можно будет влиять на эффективность работы генетического алгоритма.

Чем меньше ρ, тем ближе система S к требуемой конфигурации. При $\rho = 0$ система полностью соответствует заданному требованию.

Но, тем не менее, нет гарантии того, что из заданного множества подсистем $\{S_k\}_{k=1}^{K}$, их моделей $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$ и связей $T = \{t_q\}_{q=1}^{Q}$ можно построить требуемую систему S.

4. Операторы генетического алгоритма

После выбора целевой функции можно описать операторы генетического алгоритма.

Оператор отбора должен отбирать для скрещивания те генотипы, которые ближе всего к требуемой конфигурации. Организовать этот процесс можно описать, например, следующим образом: упорядочить все имеющиеся генотипы по качеству (от лучшего к худшему) и скрещивать соседние пары. Можно также разрешить лучшим генотипам участвовать в нескольких скрещиваниях.

Оператор скрещивания можно описать следующим образом. Рассмотрим два генотипа:

$$G_1 = \left(g_1^1,\ldots,g_q^1,g_{q+1}^1,\ldots g_Q^1,h_1^1,\ldots,h_k^1,h_{k+1}^1,\ldots,h_K^1\right);$$

$$G_2 = \left(g_1^2,\ldots,g_q^2,g_{q+1}^2,\ldots g_Q^2,h_1^2,\ldots,h_k^2,h_{k+1}^2,\ldots,h_K^2\right).$$

Случайным образом выбираем два числа: q из диапазона (множества) $\{1,2,\ldots,Q\}$ и k из диапазона $\{1,2,\ldots,K\}$. А затем меняем соответствующие участки генотипов. Таким образом, из родителей G_1 и G_2 получаются потомки G_3 и G_4, наследующие свойства родителей:

$$G_3 = \left(g_1^1,\ldots,g_q^1,g_{q+1}^2,\ldots g_Q^2,h_1^1,\ldots,h_k^1,h_{k+1}^2,\ldots,h_K^2\right);$$

$$G_4 = \left(g_1^2,\ldots,g_q^2,g_{q+1}^1,\ldots g_Q^1,h_1^2,\ldots,h_k^2,h_{k+1}^1,\ldots,h_K^1\right).$$

Можно ограничиться и выбором какого-то одного числа (q или k). Сколько чисел будет выбрано (одно или два) и если одно, то какое именно, можно также определять случайным образом. Можно также выбрать для каждого диапазона ($\{1,2,\ldots,Q\}$ и $\{1,2,\ldots,K\}$) по две точки и меняться «серединками».

Оператор мутации можно описать следующим образом. Рассмотрим один генотип

$$G = \left(g_1,\ldots,g_q,\ldots g_Q,h_1,\ldots,h_k,\ldots,h_K\right).$$

Случайно выбираем два числа: q из диапазона $\{1,2,\ldots,Q\}$ и k из диапазона $\{1,2,\ldots,K\}$. А затем меняем значения g_q и h_k следующим образом. Если было $g_q = 0$, меняем его на единицу: $g_q = 1$. И наоборот: если было $g_q = 1$, то станет $g_q = 0$. Если было $h_k = r$, то меняем его на любое другое из диапазона $\{0,1,\ldots,r-1,r+1,\ldots,R(k)\}$. Естественно, при $r = 0$ это будет диапазон $\{1,\ldots,R(k)\}$, а при $r = R(k)$ это будет диапазон $\{0,1,\ldots,R(k)-1\}$.

Оператор редукции должен удалять слабые генотипы. Для этого после скрещивания необходимо определить качества всех потомков. Затем потомков и родителей объединять в одно множество генотипов и упорядочить их по качеству от лучшего к худшему. Кто оказался в худшей половине, тот удаляется.

5. Решение задачи

Перед началом работы операторов генетического алгоритма необходимо задать начальную популяцию генотипов. Если есть какие-либо гипотезы о структуре и функционировании системы S, то генотипы может описать сам проектировщик. Иначе можно случайно сгенерировать несколько генотипов. Размер популяции также определяется проектировщиком.

Процесс работы генетического алгоритма можно также описать сетью Петри (рис. 7) так, как это делалось в работе [14].

Начальная популяция генотипов G_1, G_2, … G_{2D} размещается в позиции сети Петри. Оператор sel осуществляет отбор генотипов для скрещивания. Операторы $cross_1$, …, $cross_d$, …, $cross_D$ осуществляют скрещивание генотипов, которые затем подвергаются мутации оператором mut. Цикл работы завершает оператор red, удаляющий слабые генотипы. Затем процесс повторяется.

Остановка этого циклического процесса может определяться разными способами:

– заданием определенного числа циклов;

– заданием нижнего предела качества ρ всей популяции;

– заданием нижнего предела качества ρ части популяции (например, половины).

Выводы

Основным результатом данной работы является формализация и алгоритмизация процесса проектирования перестраиваемого ПМ путем формирования решения с изменяющейся структурой и набором компонентов с использованием ГА на основе СП. Решение этой задачи существенно осложняется огромным количеством возможных вариантов реализации, из которых необходимо выбрать один, удовлетворяющий заданному условию. Для преодоления этой проблемы были использованы генетические алгоритмы, адаптированные к решению поставленной задачи путем использования сетей Петри.

В теоретическом плане перспективным представляется развитие предлагаемого подхода на задачи проектирования перестраиваемого процессорного модуля, адаптирующегося к изменяющимся задачам.

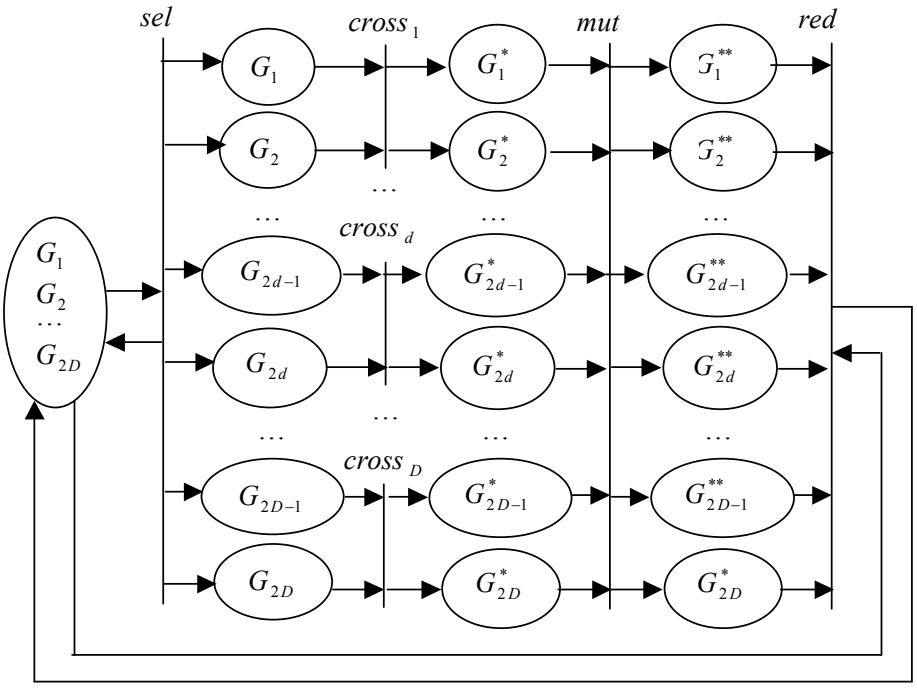

Рис. 7. Описание генетического алгоритма сетью Петри

В практическом плане эффективным будет внедрение данного подхода в САПР на основе языка VHDL. Это позволит автоматизировать процесс генерации вариантов различных проектных решений, который до сих пор производится проектировщиками вручную.

Список литературы

1. Бондаренко М.Ф., Маторин С.И., Ельчанинов Д.Б. Системная технология моделирования информационных и организационных систем: Учебное пособие. – Х.: ХНУРЭ, 2005. – 116 с.

2. Бондаренко М.Ф., Соловьева Е.А., Маторин С.И., Ельчанинов Д.Б. Объектная технология моделирования информационных и организационных систем: Учебное пособие. – Х.: ХНУРЭ, 2005. – 160 с.

3. Бондаренко М.Ф., Соловьева Е.А., Маторин С.И., Ельчанинов Д.Б. Системологическая технология моделирования информационных и организационных систем: Учебное пособие. – Х.: ХНУРЭ, 2005. – 136 с.

4. Белова Н.В., Лобода В.Г., Петросов Д.А. Функционально-ориентированный перестраиваемый процессорный модуль // Системи обробки інформації. – Х.: ХУ ПС, 2005. – Вип. 3 (43). – С. 8-18.

5. Лобода В.Г., Петросов Д.А. Концепция построения структур функционально ориентированных вычислительных устройств // АСУ и приборы автоматики. – 2003. – Вып. 122. – С. 61-71.

6. Белова Н.В., Долженкова Т.Г., Петросов Д.А. Модель перепрограммируемого процессора с гибкой архитектурой // Сб. мат. 8-го Межд. молод. форума «Радиоэлектроника и молодежь в XXI веке». – Х.: ХНУРЭ. – 2004. – Ч. 2. – С 277.

7. Ельчанинов Д.Б., Кривуля Г.Ф., Лобода В.Г., Механа Сами. Применение генетических алгоритмов и многоуровневых сетей Петри при проектировании компьютерной техники // Радиоэлектроника и информатика. – 2002. – № 1. – С. 89-97.

8. Программирование искусственного интеллекта в приложениях / М. Тим Джонс; Пер. с англ. – М.: ДМК Пресс, 2004. – 312 с.

9. Люгер Джордж, Ф. Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е изд.; Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 864 с.

10. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2000. – 352 с.

11. Советов Б.Я., Яковлев С.А. Моделирование систем. – М.: Высш. шк., 1998. – 319 с.

12. Григорьев А.В. Представление генетических алгоритмов сетями Петри в задаче размещения. Автореф. дис. канд. техн. наук. – Казань, 2002. – 20 с.

13. Молчанов Г.И. Повышение качества печатных плат в процессе автоматической трассировки соединений // Вестник НТУ «ХПИ». – Х.: НТУ «ХПИ», 2001. – Вып. 114. – С. 101-106.

14. Ельчанинов Д.Б., Петросов Д.А., Механа Сами. Применение генетических алгоритмов при проектировании компьютерной техники // Вестник Херсонского государственного университета. – 2003. – № 2 (18). – С. 45-49.

15. Петросов Д.А. Лобода В.Г., Ельчанинов Д.Б. Представление генетических алгоритмов сетями Петри в задачах проектирования компьютерной техники // Материалы научно-практической конференции «Информационные технологии – в науку и образование». – Х.: ХНУРЭ, 2005. – С. 48-51.

16. Колмогоров А.Н., Фомин С.В. Элементы теории функций и функционального анализа: Учебник для вузов. – 6-е изд., испр. – М.: Наука. Гл. ред. физ.-мат. лит., 1989. – 624 с.

Поступила в редколлегию 22.03.2007

Рецензент: д-р физ.-мат. наук, проф. С.В. Смеляков, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.