МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

# АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

**Всеукраинский межведомственный научно-технический сборник**

**Основан в 1965 г.**

## Выпуск 140

Харьков
2007

УДК 62-50

В сборнике представлены результаты исследований, касающихся компьютерной инженерии, управления, технической диагностики, автоматизации проектирования, оптимизированного использования компьютерных сетей и создания интеллектуальных экспертных систем. Предложены новые подходы, алгоритмы и их программная реализация в области автоматического управления сложными системами, оригинальные информационные технологии в науке, образовании, медицине.

Для преподавателей университетов, научных работников, специалистов, аспирантов.

У збірнику наведено результати досліджень, що стосуються комп'ютерної інженерії, управління, технічної діагностики, автоматизації проектування, оптимізованого використання комп'ютерних мереж і створення інтелектуальних експертних систем. Запропоновано нові підходи, алгоритми та їх програмна реалізація в області автоматичного управління складними системами, оригінальні інформаційні технології в науці, освіті, медицині.

Для викладачів університетів, науковців, фахівців, аспірантів.



Свидетельство о государственной регистрации
печатного средства массовой информации КВ № 12073-944ПР от 07.12.2006 г.

*Адрес редакционной коллегии:* Украина, 61166, Харьков, просп. Ленина, 14, Харьковский национальный университет радиоэлектроники, комн. 321, тел. 70-21-326

## СОДЕРЖАНИЕ

УДК 006.91

*И.П. ЗАХАРОВ, М.П. СЕРГИЕНКО*

# АВТОМАТИЗАЦИЯ ИЗМЕРЕНИЯ ЧАСТОТНЫХ ПАРАМЕТРОВ СВЧ-СИГНАЛА В ВОЛНОВОДЕ

Предлагается метод измерения длины волны, частоты и отклонения частоты СВЧ-колебаний в волноводе, основанный на измерении мощностей, рассеиваемых на двух чувствительных элементах, интегрированных в широкую стенку волновода, с последующим пересчетом их в отношения в искомый параметр. Метод технически легко реализуем в широком частотном диапазоне, включая миллиметровые и субмиллиметровые волны

Оперативный контроль частотных параметров излучения СВЧ-генераторов является необходимым на всех этапах их жизненного цикла. Для его осуществления используют волномеры на основе перестраиваемого объемного резонатора [1]. Их недостатками являются низкое быстродействие, а также невозможность измерения частоты на больших уровнях мощности из-за снижения электрической прочности волновода. Выходом из этой ситуации является применение измерительной линии на основе поглощающей стенки. В [2] рассмотрена тепловая измерительная линия, где в качестве датчика использована константановая фольга, впаянная в узкую стенку волновода, в качестве термодатчиков применены две хромель-копелевые (рабочая и компенсационная) микротермопары. Разность сигналов этих микротермопар регистрируют микроамперметром, получая при перемещении термозондов вдоль волновода распределение поля, по которому можно рассчитать длину волны в волноводе. Недостатками такого метода являются низкая чувствительность, вызванная искажениями поглощающей стенкой и контактным термодатчиком картины температурного поля в волноводе, низкая точность определения перемещения термозондов вдоль узкой стенки волновода, а также сложность автоматизации процесса измерения. В связи с этим актуальным является вопрос разработки новых методов измерения длины волны, частоты и отклонения частоты сигналов в волноводе, позволяющих проводить их оперативное измерение с необходимой точностью.

*Цель* данной работы – автоматизация процесса измерения частоты СВЧ-колебаний в волноводе в широком диапазоне длин волн.

*Задача* – разработка метода измерения частотных параметров СВЧ-сигнала в волноводе.

Предлагаемый метод основан на измерении мощностей, рассеиваемых на расположенных на широкой стенке волновода чувствительных элементах. В соответствии с [2] мощность, рассеиваемая на прямоугольном участке широкой стенки волновода, определяется выражением

$$P\left(\frac{\lambda}{\lambda_{кр}}\right)=\frac{R_s P_{10} h^* d}{Z_{C10} b\left(1-\frac{\lambda^2}{\lambda_{кр}^2}\right)}\left[1+\left(2-\frac{\lambda^2}{\lambda_{кр}^2}\right)\frac{\sin(\pi h^*)}{\pi h^*}\cos[\pi(x^*+h^*)]\right], \qquad (1)$$

где $R_s=\frac{1}{\sigma\delta}$ – поверхностное сопротивление поглощающей стенки с удельной проводимостью $\sigma$; $\delta$ – глубина скин-слоя; $Z_{10}, P_{10}$ – характеристическое сопротивление и мощность волны $H_{10}$; $\lambda$, $\lambda_{кр}=\frac{2}{\sqrt{\left(\frac{m}{a}\right)^2+\left(\frac{n}{b}\right)^2}}$ – длина волны и критическая длина волны; $a\times b$ – размеры поперечного сечения волновода ($a>b$); $m, n$ – индексы волны (для волны $H_{10}$

критическая длина волны $\lambda_{кр} = 2a$); d – продольный размер датчика; $h^* = h/a$; h – поперечный размер датчика; $x^* = x/a$; x – расстояние от ребра волновода до края датчика на широкой стенке.

**Определение длины волны**

Для определения длины волны $\lambda$, проходящей в волноводе, к широкой стенке волновода прикрепляют два датчика одинаковой длины l. Отношение мощностей $P_1\left(\frac{\lambda}{\lambda_{кр}}\right)$ и $P_2\left(\frac{\lambda}{\lambda_{кр}}\right)$, рассеиваемых на этих датчиках,

$$A\left(\frac{\lambda}{\lambda_{кр}}\right) = \frac{P_1\left(\frac{\lambda}{\lambda_{кр}}\right)}{P_2\left(\frac{\lambda}{\lambda_{кр}}\right)} = \frac{p_1\left(\frac{\lambda}{\lambda_{кр}}\right)}{p_2\left(\frac{\lambda}{\lambda_{кр}}\right)}, \qquad (2)$$

где

$$p_{1,2}\left(\frac{\lambda}{\lambda_{кр}}\right) = \frac{h^*_{1,2}}{\left(1-\frac{\lambda^2}{\lambda^2_{кр}}\right)}\left[1+\left(2-\frac{\lambda^2}{\lambda^2_{кр}}\right)\frac{\sin(\pi h^*_{1,2})}{\pi h^*_{1,2}}\cos[\pi(x^*_{1,2}+h^*_{1,2})]\right]. \qquad (3)$$

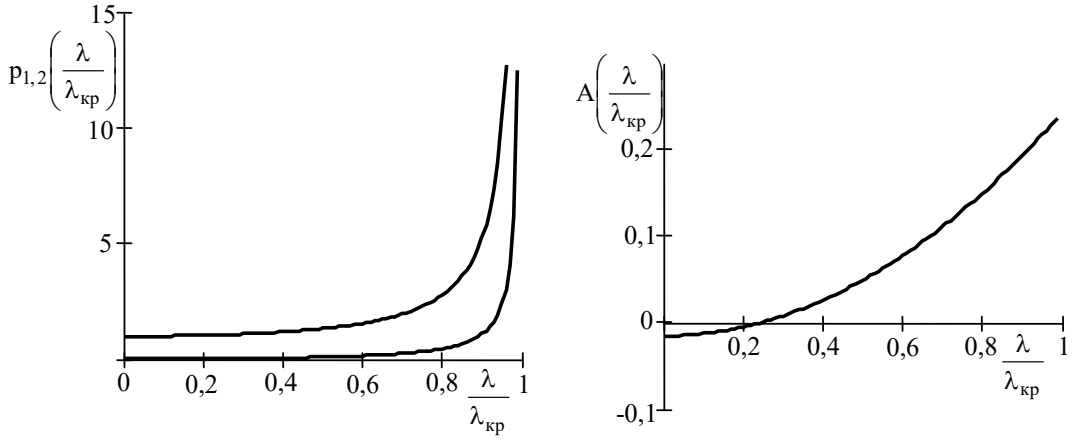
Таким образом, устраняется зависимость от уровня мощности СВЧ-сигнала, физических характеристик волновода и продольного размера поглощающих элементов.

Зависимость $A\left(\frac{\lambda}{\lambda_{кр}}\right)$ должна обладать большой крутизной для повышения точности определения частотных параметров СВЧ-сигнала в волноводе. Это достигается путем такого расположения датчиков, при котором одна из частотных зависимостей (1) намного отличается от другой. Поскольку вид зависимости $A\left(\frac{\lambda}{\lambda_{кр}}\right)$ полностью определяется поперечными размерами датчиков и их размещением на широкой стенке волновода (параметрами $h^*_{1,2}$ и $x^*_{1,2}$), остановимся на выборе этих размеров подробнее.

Для снижения погрешности от неточности установки датчиков предлагается один из них размещать по всей ширине стенки волновода, что легко реализуется в миллиметровом диапазоне волн, т.е. $x^*_1 = 0$, $h^*_1 = 1$. Требования к геометрическим параметрам второго датчика вытекают из технических возможностей и точности их исполнения и необходимости обеспечения значительной крутизны зависимости $A\left(\frac{\lambda}{\lambda_{кр}}\right)$. Методом математического моделирования было выявлено, что поставленные условия выполняются при соотношениях указанных параметров, приведенных в таблице.

| $h^*_2$ | 0,06 | 0,08 | 0,1 | 0,15 | 0,2 | 0,25 | 0,3 | 0,4 | 0,5 | 0,6 | 0,7 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^*_2$ | 0,3 | 0,29 | 0,28 | 0,26 | 0,23 | 0,21 | 0,19 | 0,16 | 0,15 | 0,17 | 0,15 |

Например, для $h_2^* = 0{,}5$ и $x_2^* = 0{,}15$ зависимости (3) и (2) показаны на рисунке (поз.а, б).




а б

Зависимости $p_{1,2}\left(\frac{\lambda}{\lambda_{кр}}\right)$ (а) и $A\left(\frac{\lambda}{\lambda_{кр}}\right)$ (б) при $h_2^* = 0{,}5$ и $x_2^* = 0{,}15$

По измеренному значению $A\left(\frac{\lambda}{\lambda_{кр}}\right)$ для определения частоты сигнала используем выражение (2), из которого следует

$$\lambda = \frac{\lambda_{кр}\sqrt{[q - s\cdot A(\lambda/\lambda_{кр})][\pi h_1^* + 2q] + \pi h_2^*[1 - A(\lambda/\lambda_{кр})] - 2s}}{q - s}, \qquad (4)$$

где $q = \sin(\pi h_1^*)\cos[\pi(2x_1^* + h_1^*)]$; $s = \sin(\pi h_2^*)\cos[\pi(2x_2^* + h_2^*)]$.

**Определение частоты сигнала**

Из выражения (4) нетрудно определить частоту сигнала

$$f = \frac{v}{\lambda}, \qquad (5)$$

здесь $v$ – фазовая скорость распространения электромагнитных колебаний в среде (для свободного пространства $v$ – скорость распространения света).

**Определение отклонения частоты сигнала**

*Если частота сигнала* $f_0$ *(длина волны* $\lambda_0$*) неизвестна*, отклонение частоты $\delta f$ определяют следующим образом. Определяют максимальное и минимальное из наблюдаемых значения $A(\lambda/\lambda_{кр})$ для значений длин волн соответственно $\lambda_{max}$ и $\lambda_{min}$, находят по каждому из них частоты $f_{min}$ и $f_{max}$ в соответствии с выражениями (4) и (5), после чего получают половину разности между этими частотами $\Delta f = |f_2 - f_1|/2$. В этом случае

$$f_0 = \frac{f_1 + f_2}{2}; \quad \delta f = \pm\frac{\Delta f}{f_0}. \qquad (6)$$

Недостатком такого определения отклонения частоты является наличие погрешности определения частоты $f_0$, которая может иметь смещение.

*Если частота сигнала* $f_0$ *(длина волны* $\lambda_0$*) известна*, алгоритм определения отклонения частоты следующий.

По измеренному значению $A(\lambda/\lambda_{кр})$ в соответствии с изложенным выше алгоритмом определяют частоту f (по формулам (4), (5)). Тогда отклонение частоты

$$\delta f = \frac{f - f_0}{f_0}. \tag{7}$$

Однако при этом появляется погрешность определения длины волны по выражению (4), связанная с неточным определением $A(\lambda/\lambda_{кр})$, вследствие нелинейных преобразований.

Чтобы этого избежать, можно воспользоваться следующим алгоритмом. Измеряют значения $A(\lambda_0/\lambda_{кр})$ и $A(\lambda_1/\lambda_{кр})$ для $\lambda_1$, имеющей искомое отклонение от $\lambda_0$, и рассчитывают их разность:

$$\Delta A = A\left(\frac{\lambda_1}{\lambda_{кр}}\right) - A\left(\frac{\lambda_0}{\lambda_{кр}}\right). \tag{8}$$

С другой стороны, эту разность можно найти по формуле

$$\Delta A = W\left(\frac{\lambda}{\lambda_{кр}}\right) \cdot \Delta\left(\frac{\lambda}{\lambda_{кр}}\right), \tag{9}$$

где $W\left(\frac{\lambda}{\lambda_{кр}}\right) = \frac{dA\left(\frac{\lambda}{\lambda_{кр}}\right)}{d\left(\frac{\lambda}{\lambda_{кр}}\right)}$ – коэффициент влияния, рассчитанный для длины волны $\lambda_0$.

Его значение можно получить из выражения (2):

$$W\left(\frac{\lambda}{\lambda_{кр}}\right) = \frac{2\left(\frac{\lambda}{\lambda_{кр}}\right)}{\pi h_2^* p_2\left(\frac{\lambda}{\lambda_{кр}}\right)}\left[\frac{h_1^*}{h_2^*} A\left(\frac{\lambda}{\lambda_{кр}}\right) s - q\right], \tag{10}$$

где q и s взяты из (4).

Выражение (10) рассчитываем при длине волны $\lambda_0$.

Из формулы (9) следует, что

$$\Delta\left(\frac{\lambda}{\lambda_{кр}}\right) = \frac{\Delta A}{W\left(\frac{\lambda}{\lambda_{кр}}\right)}, \tag{11}$$

где $\Delta A$ определено по результатам эксперимента из выражения (8).

Разность частот $\Delta f$ определяется по формуле (5).

Отклонение частоты

$$\delta f = \frac{\Delta f}{f_0}. \tag{12}$$

Преимуществом данного алгоритма по сравнению с предыдущими является устранение погрешности, возникающей при пересчете выражения (2) в (4) при неточном определении $A(\lambda/\lambda_{кр})$, и отсутствие погрешности определения частоты $f_0$.

## Выводы

Разработан метод измерения частотных параметров СВЧ-сигнала в волноводе. По сравнению с существующими на сегодняшний день методами предложенный метод отличается применимостью в широком диапазоне длин волн, включая миллиметровые и субмиллиметровые, простотой технической реализации, возможностью автоматизации измерительного процесса, а также возможностью получения результатов измерения в реальном времени.

*Научная новизна* проведенных исследований заключается в нахождении оптимальных соотношений геометрических параметров датчиков и способов их расположений, позволивших получить максимальную чувствительность метода.

*Практическая значимость* состоит в упрощении процесса измерения частоты СВЧ-колебаний за счет его полной автоматизации.

**Список литературы: 1.** *Измерения* на миллиметровых и субмиллиметровых волнах: Методы и техника/ Р.А. Валитов, С.Ф. Дюбко, Б.И. Макаренко и др.; Под ред. Р.А. Валитова, Б.И. Макаренко. М.: Радио и связь, 1984. 256 с. **2.** *Волков. В.М.* Проектування засобів вимірювання прохідної потужності: Навч. посібник. Харків: ХТУРЕ, 2000. 160 с.

*Поступила в редколлегию 29.08.2007*

**Захаров Игорь Петрович**, д-р техн. наук, профессор кафедры метрологии и измерительной техники ХНУРЭ. Научные интересы: неопределенность измерений различных физических величин. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-1331.

**Сергиенко Марина Петровна**, канд. техн. наук, ассистент кафедры метрологии и измерительной техники ХНУРЭ. Научные интересы: динамические измерения. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-1331.

---

УДК 519.713:681.326

*И.В.ХАХАНОВА*

# СИНТЕЗ МОДЕЛЕЙ УПРАВЛЯЮЩИХ АВТОМАТОВ ДЛЯ SOC-ФИЛЬТРОВ С КОНВЕЙЕРНОЙ АРХИТЕКТУРОЙ

Анализируются существующие модели цифровых автоматов и способы их реализации на микросхемах программируемой логики. Предлагаются две модели управляющих автоматов для устройств с конвейерной архитектурой, реализующие DSP задачи обработки изображений на основе SoC. Разрабатывается программный модуль автоматической генерации HDL-кода для упомянутых типов управляющих автоматов.

## 1. Введение

Сложность цифровых систем и сетей, имплементированных в кристаллы PLD, согласно закону Мура [1] удваивается каждые полтора года. Они представляют собой функционально и конструктивно законченные устройства, реализованные в чипе, и содержат микропроцессоры, блоки памяти, контроллеры, периферийные устройства, порты ввода-вывода информации. В такие архитектуры в последнее время довольно часто включаются блоки, решающие задачи цифровой обработки сигналов (DSP – digital signal processing). Для повышения быстродействия DSP-приложений применяется конвейерная архитектура, которая позволяет обрабатывать большие потоки входных данных при высокой частоте синхронизации. DSP-модуль можно представить в виде структуры, показанной на рис. 1, которая состоит из операционного автомата (ОА) в виде конвейера, управляющего автомата (УА) и блоков памяти различной размерности, используемой для хранения исходных данных, промежуточных результатов и выходной информации DSP-преобразования.

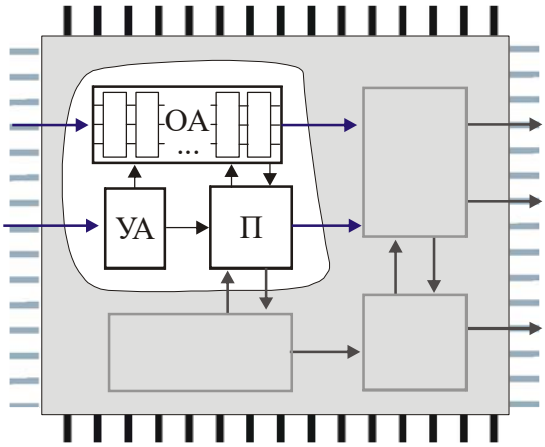



Рис. 1. DSP-модуль в системе на кристалле

*Цель работы* – автоматическая генерация управляющих автоматов для конвейерных вычислительных архитектур SoC фильтров, реализуемых в кристаллах PLD, что дает возможность существенно (x10, x100) повысить быстродействие обработки видеоинформации по сравнению с программным способом.

Для достижения поставленной цели в работе решаются следующие *задачи*.

1. Анализ моделей цифровых автоматов и способов их реализации в аппаратуре на основе PLD с помощью языков VHDL и Verilog.

2. Разработка моделей управляющих автоматов для SoC фильтров, реализующих стандарт JPEG 2000 с конвейерной обработкой данных.

3. Разработка шаблонов VHDL- и Verilog-моделей в целях автоматической генерации управляющих автоматов для операционных устройств, реализующих стандарт JPEG 2000 с конвейерной обработкой данных.

4. Программная реализация процедур автоматической генерации VHDL и Verilog-моделей управляющего автомата для конвейерных архитектур.

**2. Модели цифровых автоматов. Абстрактный автомат**

Абстрактный автомат [2] является математической моделью дискретного устройства и определяется вектором $S=(A,Z,W,\varphi,\psi,a_1)$, где векторы $A=(a_1,...,a_m,...a_M)$; $Z=(z_1,...,z_m,...z_M)$ и $W=(w_1,...,w_g,...,w_G)$ – множества состояний входных и выходных сигналов; $\varphi: A\times Z\to A$ – функция переходов, реализующая отображение $D_\varphi =\subseteq A\times Z$ в A, когда парам <состояние - входной сигнал> $(a_m,z_f)$ функция $\varphi$ ставит в соответствие состояния автомата $a_s=\varphi(a_m,z_1)$, $a_s\in A$; $\psi: A\times Z\to W$ – функция выходов, реализующая отображение $D_\lambda =\subseteq A\times Z$ в W, когда функция $\psi$ парам <состояние - выходной сигнал> $(a_m,z_f)$ ставит в соответствие выходные сигналы автомата $w_s=\psi(a_m,z_1)$; $a_1\in A$ – начальное состояние автомата.

Абстрактный автомат (рис.2) имеет один вход и один выход. Он функционирует в дискретном времени, принимающем целые неотрицательные значения t = 0, 1, 2, ... . В каждый момент времени t автомат находится в некотором состоянии a(t) из множества состояний автомата, причем в начальный момент времени t=0 он всегда находится в начальном состоянии $a(0)=a_1$. В момент t, в состоянии a(t), автомат может принимать букву входного алфавита $z(t)\in Z$. Функцией выходов $w(t)=\psi(a(t),z(t))$ формируется соответствующая буква выходного алфавита, а функцией перехода $a(t+1)=\varphi(a(t),z(t))$ – следующее состояние. Идея абстрактного автомата состоит в реализации некоторого отображения множества слов входного алфавита Z в множество слов выходного алфавита W.

Рис 2. Абстрактный автомат

На практике наиболее распространены два класса автоматов – Мили и Мура (G.H.Mealy и E.F.Moore), функционирование которых задается уравнениями:

1) $a(t+1) = \varphi(a(t), z(t));\ w(t) = \psi(a(t), z(t)), t = 0,1,2,\ldots$ .

2) $a(t+1) = \varphi(a(t), z(t));\ \ w(t) = \psi(a(t)), t = 0,1,2,\ldots$ .

Автомат называется конечным, если конечны множества A, Z и W. Автомат называется полностью определенным, если $D_\varphi = D_\psi = A \times Z$. Для полностью определенного автомата области определения функций $\varphi$ и $\psi$ совпадают со множеством пар вида $(a_m, z_f)$. У неполностью определенного или частичного автомата функции $\varphi$ и $\psi$ определены не для всех пар $(a_m, z_f) \in A \times Z$.

### 3. Структурный автомат

Здесь учитывается структура входных и выходных сигналов автомата, а также внутреннее устройство на уровне структурных схем. Основной задачей структурной теории автоматов является нахождение общих приемов построения структурных схем автоматов на основе композиции элементарных автоматов. При построении конечных автоматов функции переходов $\varphi$ и выходов $\psi$ реализуются с помощью комбинационных схем $CL_\varphi$ и $CL_\psi$ соответственно, а память автомата реализуется с помощью регистра RG, который хранит код внутреннего состояния в каждый момент автоматного времени. В зависимости от способов определения функций переходов и выходов в [3] представлена детальная классификация конечных автоматов (рис. 2). Классы A (рис. 3, а) и B (рис. 3, б) соответствуют общей форме автоматов Мили и Мура. Если каждый выходной набор w(t) автомата Мура совпадает с кодом его внутреннего состояния a(t), то данный автомат относится к классу C [2, 3] (рис.3, в) и его поведение описывается уравнениями: $a(t+1) = \varphi(a(t), z(t));$ $w(t) = a(t)$. Особенностью автомата C является отсутствие комбинационной схемы $CL_\psi$, что делает его выходы регистровыми. Если каждый выходной набор w(t) автомата Мили совпадает с кодом его следующего состояния a(t+1), то рождается автомат класса D [3,5] (рис. 3, г). Его особенностью также является отсутствие комбинационной схемы $CL_\psi$, однако значения входных функций формируются на входах памяти автомата RG.

Автомат D описывается уравнениями: $a(t+1) = \varphi(a(t), z(t));\ \ w(t) = a(t+1)$. Различие между автоматами классов C и D: в первой модели выходной набор w(t) автомата совпадает с кодом его текущего состояния a(t), во втором – с кодом следующего состояния a(t+1). Поскольку в автомате класса C выходной набор не зависит от значений входов, то он является частным случаем автомата Мура. Автомат D является частным случаем автомата Мили, поскольку выходные значения зависят от текущего состояния a(t) и входов z(t). Преимущество автоматов классов C и D перед автоматами A и B заключается в более простой схемной реализации. Однако не всякое устройство может быть построено с помощью автоматов классов C и D. Коды внутренних состояний конечного автомата могут также полностью определяться значениями входных переменных. Если каждый входной набор z(t) автомата Мили совпадает с кодом следующего состояния a(t+1), то такой автомат соответствует классу E (рис. 3, д), уравнения функционирования которого имеют вид: $a(t+1) = z(t);\ \ w(t) = \psi(z(t), a(t))$. Если каждый входной набор z(t) автомата Мура совпадает с кодом следующего состояния a(t+1), то такой автомат соответствует классу F (рис. 3, е), поведение которого может быть описано уравнениями: $a(t+1) = z(t);\ \ w(t) = \psi(a(t))$. Особенностью архитектур автоматов класса E и F является отсутствие комбинационной

схемы $CL_\varphi$. В автомате класса E комбинационная схема $CL_\psi$ имеет связь со входом автомата, а в автомате класса F ее нет. Автоматы E и F не имеют цепей обратной связи. В автомате класса G (рис. 3, ж) каждый выходной набор w(t) автомата Мили совпадает с кодом его внутреннего состояния a(t), а каждый входной набор – с кодом состояния a(t+1). Автомату класса G соответствует обычный регистр. На практике редко удается непосредственно реализовывать автоматы классов C - F. В работе [3] выполнен анализ синтеза управляющих автоматов на микросхемах программируемой логики SPLD и CPLD.

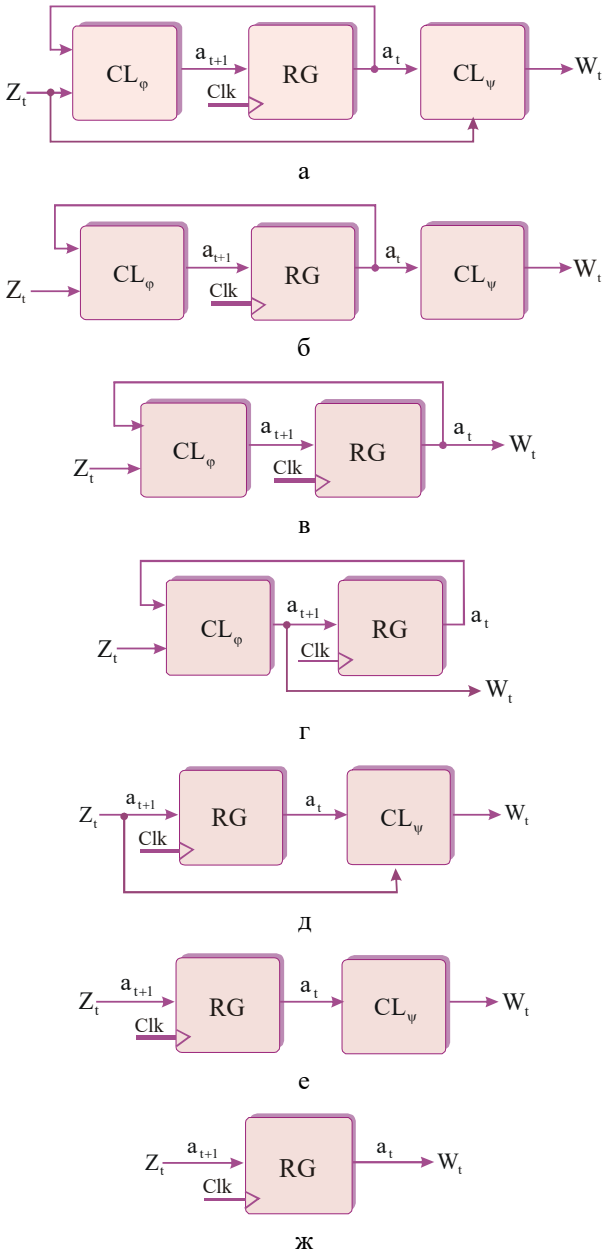



Рис. 3. Основные структурные схемы конечных автоматов: а – Мили класса A; б – Мура класса B; в – Мура класса C; г – Мили класса D; д – Мили класса E; е – Мура класса F; ж – Мура класса G

## 4. Сложность реализации автоматов

Пусть конечный автомат характеризуется числом L входных переменных множества $X=\{x_1, ..., x_L\}$, числом N выходных переменных множества $Y = \{y_1, ..., y_N\}$ и числом

разрядов R кода внутренних состояний, которое может быть определено в диапазоне от $\log_2 M$ до M, где M – число внутренних состояний автомата.

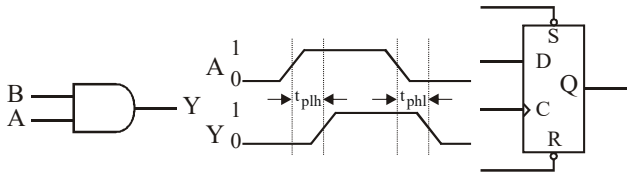
В работе [3] указывается, что на сложность реализации конечного автомата на современных PLD и на выбор модели влияет: тип входа или выхода – регистровый ($t_r$) или комбинационный ($t_c$); число входных ячеек $n_B$; число триггеров $n_{FF}$; число скрытых макроячеек $n_{BMC}$ , когда буферизация входных сигналов осуществляется с помощью скрытых макроячеек; суммарное число внешних выводов $n_{P+BMC}$ и скрытых макроячеек. Значение этих параметров остается важным для реализации автоматов на SPLD или CPLD. Однако в связи с современным развитием электроники и появлением систем и даже сетей на кристаллах уже нет таких жестких требований к уменьшению размерности устройства, уменьшению входных и выходных переменных, которые являются внутренними линиями микросхемы. Основным параметром, определяющим сложность реализации управляющего автомата, является его размерность, которая выражается в эквивалентных вентилях. Современный рынок электронных технологий предлагает многообразие архитектур и технологий изготовления микросхем PLD и ASIC, которые своими компонентами образуют функции устройства. Каждому модулю микросхемы ставится в соответствие некоторое количество эквивалентных вентилей в качестве метрики (системы) оценивания сложности проектов, реализованных в различных chipset платформах.

Быстродействие и аппаратная сложность являются взаимно противоречивыми понятиями при оптимизации устройства в процессе его реализации. Для повышения рабочей частоты вводятся дополнительные входные и выходные регистры, что увеличивает размерность схемы. Если необходимо получить устройство с меньшим количеством ресурсов, то для этого нужно пожертвовать быстродействием или потребляемой мощностью.

**5. Оценка временных характеристик автомата**

При вычислении временных характеристик устройства используются следующие параметры:

1) Propagation Delay – задержка распространения – время между поступлением входного сигнала и появлением выходной реакции. Интервал [tplh, tphl], изображенный на рис. 4, а, есть время между событием на входе элемента и изменением его выхода (0–1) и (1–0) соответственно. Величина измеряется между 50% точками уровней входного и выходного сигналов (рис. 4, б). Фрагмент индекса lh и hl обозначает изменение выходного сигнала. Для триггеров (рис. 4, в) определены следующие временные параметры: $T_{CQ}$ – время, необходимое для изменения выхода Q в ответ на поступление активного фронта синхросигнала. При этом выходное значение Q зависит от входа данных D; $T_{SQ}$, $T_{RQ}$ – задержка изменения выхода Q в ответ на поступление асинхронных сигналов установки (S) и сброса (R). Задержка между синхронным входом данных D и выходом Q не определена, поскольку изменение входа D не приводит к непосредственной модификации состояния триггера;




а б в

Рис. 4. Задержки комбинационного (а, б) и последовательностного (в) элементов

2) Setup Time – время установки $T_{su}$ (рис. 5) – это интервал времени до появления фронта синхросигнала, в течение которого синхронный входной сигнал не может изменять свое значение;

3) Hold Time – время хранения $T_{hd}$ (см. рис. 5) – это интервал времени, следующий за фронтом синхроимпульса, в течение которого синхронный входной сигнал не может изменять значение.

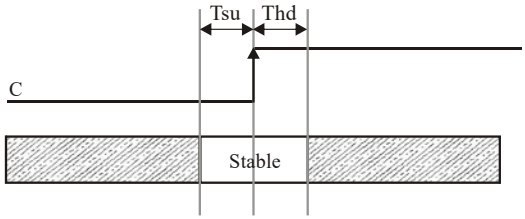



Рис. 5. Время установки и время хранения

Модель последовательностного устройства для вычисления временных параметров представлена на рис. 6. Здесь определены три пути следования сигналов:

1) Clock to Output – период времени между поступлением активного фронта синхросигнала и появлением соответствующей реакции на выходе:

$$T_{CO} = T_{c2q} + T_{comb\_Q2Omax}, \quad (1)$$

где $T_{c2q}$ – задержка регистра RG; $T_{comb\_Q2Omax}$ – самый длинный путь от выхода регистра RG к любому выходу схемы $w_i(t)$;

2) Register to Register – период времени между поступлением активного фронта синхросигнала на вход CLK регистра RG и формированием значений на входах D:

$$T_{RR} = T_{c2q} + T_{comb\_Q2Dmax} + T_{su}, \quad (2)$$

где $T_{comb\_Q2D}$ – самый длинный путь от выхода триггера Qdff регистра RG до входа триггера Ddff регистра RG;

3) Pin to Pin – максимальный логический путь от входов автомата до его внешних выходов, не содержащий последовательностных элементов:

$$T_{PP} = T_{comb\_I2Omax}. \quad (3)$$

Максимальная рабочая частота любого последовательностного устройства определяется выражением:

$$F = \frac{1}{P_{min}}, \quad (4)$$

где F – максимальная рабочая частота; $P_{min}$ – период времени, соответствующий самому длинному логическому пути схемы (структурная глубина), который равен:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}). \quad (5)$$

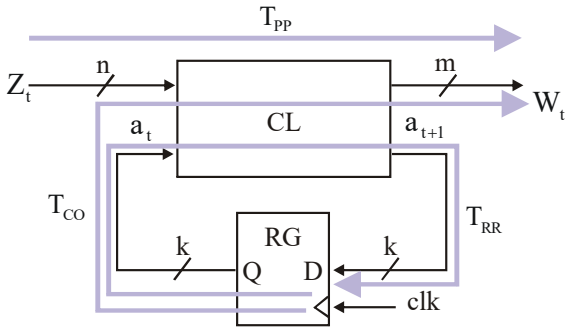



Рис. 6. Временная модель автомата

## 6. Сеть автоматов

Если цифровое устройство включает несколько автоматов управления, то их совместная работа может быть описана с помощью сети автоматов [2], которая задается вектором: $N = (Z, \{S_i\}, W, \{f_i\}, \{\psi_i\}, g)$, Z – входной алфавит, $\{S_i = (A_i, Z_i, \delta_i)\}$, $1 \le i \le n$ – множество компонентных автоматов (КА) сети, один из которых (полуавтомат) изображен на рис. 7:

$$Z_i = \begin{cases} Z_i^{'} \times Z_i^{''} & \leftarrow \quad Z_i^{'} \neq \varnothing; \\ Z_i^{''} & \leftarrow \quad Z_i^{'} = \varnothing, \end{cases}$$

$Z_i^{'}, Z_i^{''}$ – внутренний и внешний входные алфавиты $S_i$. Функция переходов $S_i$ $\delta_i : A_i \times Z_i \to A_i$;

W – выходной алфавит сети; $\{f_i : (\underset{j}{\times} A_j) \to Z_i^{'}\}$, $1 \le i, \quad j \le n$ – множество функций соединения

КА сети; $\{\psi_i : Z \to Z_i^{'}\}, 1 \le i \le n$ – множество входных функций; $g : (\underset{i}{\times} A_i) \times Z \to W$ – выходная функция сети. Множества $\{S_i\}$ и $\{f_i\}$ являются базисом и структурой сети [2].

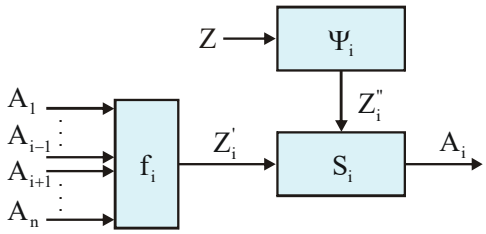



Рис. 7. Компонентный автомат сети

Сеть является общей моделью совместной работы совокупности из n автоматов, которая представлена на рис. 8.

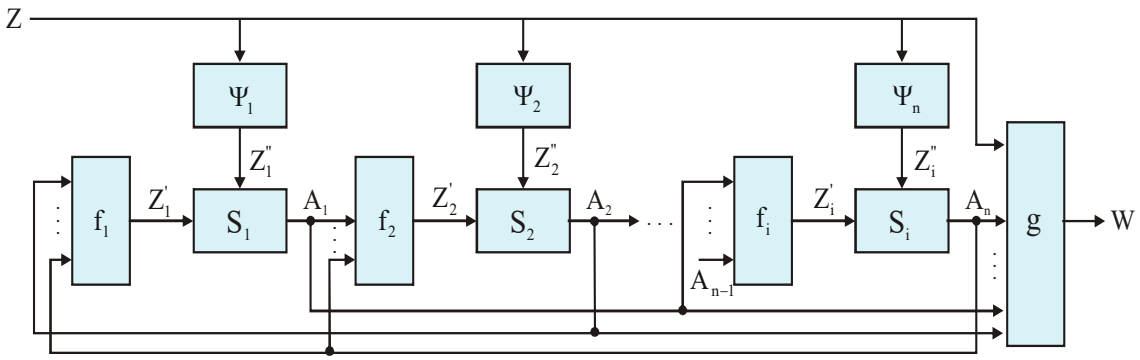



Рис. 8. Сеть из n компонентных автоматов

### 7. Создание HDL-моделей цифровых автоматов

В зависимости от задач, которые ставит перед собой проектировщик, и необходимой степени детализации управляющий автомат может быть реализован в виде HDL-моделей различных стилей и уровней описания. Однако наиболее популярным является системный (поведенческий) уровень представления моделей. Кодирование состояний и создание схемы возлагается на современные программы синтеза (генерирования), которые специально разрабатываются для конкретного аппаратного базиса. Такой подход позволяет проектировать переносимые на различные электронные базисы модели, что значительно упрощает работу проектировщика и позволяет сосредоточить усилия на разработке функциональности.

VHDL-модель автоматов Мура и Мили классов A (см. рис. 3, а) и B (см. рис. 3, б) может быть описана с помощью двух процессов (листинг 1): первый Process_1 реализует комбинационные схемы автомата $CL_\varphi$ и $CL_\psi$, второй Process_2 – описывает регистр RG. В программной модели используется символьное описание состояний State_type, реализуе-

мое с помощью VHDL-типа перечисления. Двоичные коды состояний определяются программой синтеза, что предоставляет свободу выбора значений кодов в зависимости от конкретного аппаратного базиса. Таблица переходов (для классов автомата), соответствующая функции переходов $a(t+1) = \varphi(a(t), z(t))$, реализуется в Process_1 с помощью операторов case/if . Поскольку входами комбинационной части $CL_\varphi$ являются входы автомата и его текущее состояние, то эти сигналы записываются в список чувствительности процесса. Стиль 1 предлагает задавать выходные функции $w(t) = \psi(z(t), a(t))$ автомата Мили $CL_\psi$ с помощью операторов case/if, или применять оператор case для автомата Мура: $w(t) = \psi(a(t))$ ). Стиль 2 использует дополнительные условные параллельные операторы для реализации выходных функций. Process_2, описывающий регистр состояний, представляет собой программную модель реализуемого на D-триггерах регистра с асинхронным сбросом reset в начальное состояние. Для дополнительного управления автоматом в модели может быть использован сигнал разрешения синхронизации enable.

Листинг 1. Шаблон VHDL-модели автомата

```
library IEEE;
use  IEEE.std_logic_1164.all;
entity Имя_интерфейса is
 port ( Clk: in STD_LOGIC;
     Reset: in STD_LOGIC;
     Управляющие входы Z_i: in STD_LOGIC;
     Выходы W_i: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
-- Тип, использующий символьное кодирование состояний автомата
type State_type is (состояние1, состояние2, ...);
signal State, NextState: State_type;
begin

-- Процесс для вычисления значения следующего состояния и выходов
Process_1: process (State, Управляющие входы Z_i)
begin
 -- инициализация значений выходов
     W_i <= '0';
 case State is
     when состояние1 =>
          -- Присвоение значений выходам для автомата Мура, стиль 1
          if условие then
               NextState <= состояние i;
-- Присвоение значений выходам для автомата Мили, стиль 1
          else    NextState <= состояние j;
-- Присвоение значений выходам для автомата Мили, стиль 1
          end if;
     when ...
     when others => NextState <= состояние0;
-- Присвоение значений выходам для состояния -- по умолчанию. Установка автомата в на-
чальное
-- состояние
     end case;
end  process;

Process_2: process (Clk, reset)
begin
     if Reset='1' then
     -- Или Reset = '0',
     -- если активным является низкий уровень
     -- Начальное состояние
```

```
        State <= состояние0;
    elsif Clk'event and Clk = '1' then
        State <= NextState;
    end if;
end process;
-- Условные параллельные операторы
-- назначения, стиль 2
W_i <= '1' when условие else
        '0';
. . .
end ex1_arch;
```

Аналогичным образом создается Verilog-модель автомата (листинг 2). В Verilog нет типа перечисления, существующего в VHDL, поэтому для удобства работы с состояниями используются параметры, в которых задается конкретный двоичный код для каждого состояния. Однако эти значения являются достаточно условными, потому что программы синтеза сами выбирают коды для представления состояний автомата в зависимости от выбранного chipset – класса микросхем для реализации автомата. Подобно VHDL-модели, Verilog-структура автомата состоит из двух блоков always, реализующих комбинационную и последовательностные части автомата. Первый блок always соответствует комбинационным модулям автомата $CL_{\varphi}$ и $CL_{\psi}$ (см. рис. 2), второй блок always описывает регистр RG. Так же как и в модели VHDL, таблица переходов автомата обоих классов реализуется с помощью операторов case/if в always-блоке 1. Управляющие входы автомата и текущее состояние указываются в списке чувствительности блока. Подобно модели VHDL, описание выходных функций $w(t) = \psi(z(t), a(t))$ автомата Мили реализуется с помощью операторов case/if в первом блоке always, а описание выходных функций $w(t) = \psi(a(t))$ автомата Мура ( $w(t) = \psi(a(t))$ ) – с помощью только оператора case. Стиль 2 предлагает использовать для реализации выходных функций условный оператор assign, который при синтезе реализуется комбинационной схемой. Регистр состояния RG описывается вторым блоком always. Для установки автомата в начальное состояние используется асинхронный управляющий сигнал сброса reset. Для дополнительного управления автоматом вводится сигнал разрешения синхронизации enable.

Листинг 2. Шаблон Verilog-модели автомата

```
module FSM1_synplify (clk, reset, enable, Управляющие входы Z_i, Выходы W_i);
    input clk, reset, enable;
    input Управляющие входы Z_i;
    output Выходы W_i;
/* Определение метки состояний, m – разрядность регистра состояний */
    parameter deflt = m'bxxx; /* Состояние по умолчанию */
    parameter состояние1 = m'b ...
    . . .
    reg Выходы W_i;
    reg [m:0] state, next_state;

/* Блок always для комбинационной схемы */
always @(state or enable or data_in) begin
/* Значения выходов по умолчанию*/
    W_i <= 1'b0;
    case (state)
        Состояние1 :
            if (Условие)
                begin NextState <= состояние i;
    /* Присвоение значений выходам для автомата Мили, стиль */
                end
             else
                begin NextState <= состояние i;
    /* Присвоение значений выходам для автомата Мили, стиль */
                end
```

```
    /* присвоение значений выходам для автомата Мура, стиль 1 */
        Состояние2 :
        . . .
        default : next_state <= deflt;
/* Присвоение значений выходам для состояния по умолчанию */
 /* Установка автомата в начальное состояние */
    endcase
end
/* Блок always для реализации регистра
состояний */
always @(posedge clk or negedge rst)
    if (!rst) state <= idle;
    else state <= next_state;
/* Условные параллельные операторы
назначения, стиль 2 */
assign W_i = условие ? 1'b1: 1'b0;
    . . .
endmodule
```

## 8. Системные иерархические модели автоматов

Современные устройства цифровой обработки сигналов представляют собой конвейер с управляющим автоматом (рис. 9), который подсчитывает число обработанных элементов и формирует управляющие сигналы в исключительных ситуациях: первый или последний элемент, первая или последняя строка изображения. Поскольку входные последовательности таких автоматов содержат большое количество элементов, то для их проектирования нельзя использовать классические методы синтеза структурных автоматов Мили или Мура, ввиду большого количества (100-1000) состояний. Использование комбинации {управляющий автомат Мили (Мура) – счетчик} не обеспечивает высокого быстродействия DSP-преобразователя. Как правило, управляющий блок для конвейерных вычислительных устройств строится на счетчиках, которые идентифицируют состояние управляющего автомата. Все выходные инициирующие сигналы генерируются на основе значений этих счетчиков.

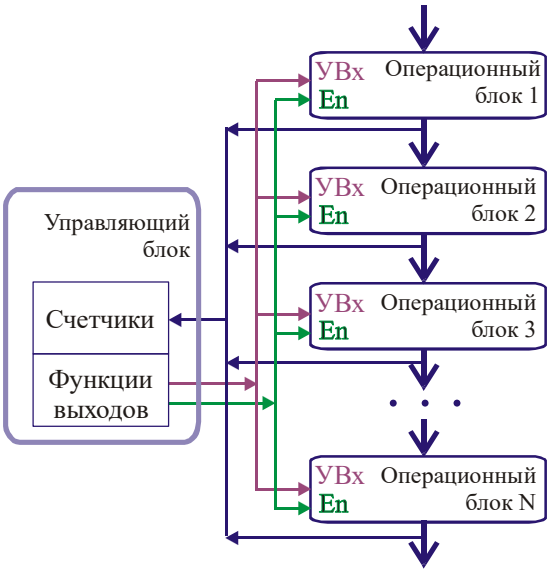



Рис. 9. Конвейерная структура DSP с макроавтоматом

Специфика управляющего модуля конвейерной архитектуры DSP-преобразователя , требует, чтобы предложенная в [2] классификация моделей автоматов, изображенных на рис. 3. была дополнена еще одним классом моделей управляющего автомата, который специализируется на конвейерный тип вычислений для DSP (рис. 10). Такая модель инте-

ресна тем, что она не имеет управляющих входных сигналов, за исключением входа синхронизации (Clk) и сброса автомата в начальное состояние (reset), и по сути является моделью автомата Мура, описываемой уравнениями:

$$a(t+1) = \varphi(a(t)); \quad w(t) = \psi(a(t)), t = 0,1,2,\ldots .$$

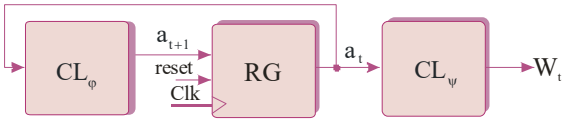



Рис. 10. Модель управляющего автомата конвейерного устройства

Блоки $CL_\varphi$ и RG реализуют счетчик, заменяющий автомату таблицу переходов, а блок $CL_\psi$ – выходные функции $w(t) = \psi(a(t))$.

Размерность схемы автомата обусловлена длиной регистра RG, а следовательно, рабочим диапазоном счетчика, который, в свою очередь, зависит от структуры входной информации. Таким образом, размерность управляющего блока зависит от двух параметров: R– разрядности счетчика и $N_W$ – количества формируемых выходных функций. Быстродействие такого устройства, как следует из формул (1) и (2), определяется самым длинным путем схемы – структурной глубиной. В данной модели: 1) логический путь "Pin to Pin" начинается от асинхронного сброса reset и заканчивается выходами W: $T_{PP} = T_{RG\_reset} + T_{CL_\psi max}$; 2) путь "Register to Register" определяет задержку логического пути между регистрами счетчика: $T_{RR} = T_{RG} + T_{CL_\varphi max} + T_{su}$; 3) путь "Clock to Output": $T_{CO} = T_{RG} + T_{CL_\psi max}$.

Минимальный рабочий период автомата определяется следующим выражением:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}) =$$
$$= \max[(T_{RG} + T_{CL_\psi max}), (T_{RG} + T_{CL_\varphi max} + T_{su}),$$
$$(T_{RG\_reset} + T_{iCL_\psi})].$$

Как видно из формулы, минимальный рабочий период, а значит и максимальная рабочая частота, зависят от аппаратной сложности схем $CL_\varphi$ и $CL_\psi$. Часто для повышения скорости работы устройства выходные функции W дополняются выходным регистром $RG_W$ (рис. 11). Это позволяет сделать работу устройства более устойчивой к состязаниям и, благодаря делению логического пути, повысить максимальную рабочую частоту всего устройства, включая и управляющий автомат. При этом изменяются временные параметры системы. Такая модель не будет иметь путей "Clock to Output" и "Pin to Pin". Минимальный рабочий период устройства зависит только от самого длинного пути "Register to Register" и описывается формулой:

$$P_{min} = \max(T_{RR}) = \max[(T_{RG} + T_{CL_\psi max} + T_{su}),$$
$$T_{RG} + T_{CL_\varphi max} + T_{su})] =$$
$$= T_{RG} + T_{su} + \max(T_{CL_\psi max}, T_{CL_\varphi max}).$$

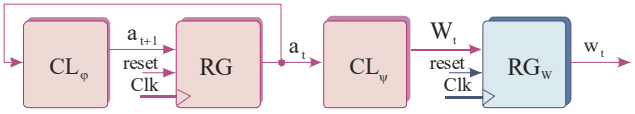



Рис. 11. Модель автомата с регистровыми выходами

При обработке двумерных данных, изображений, управляющий блок должен иметь два счетчика: 1) Сч 1 – для подсчета столбцов матрицы; 2) Сч 2 – для подсчета столбцов матрицы, если данные обрабатываются по строкам. Иначе – назначение счетчиков изменяется. Модель такого управляющего блока может быть представлена в виде сети автоматов (рис. 12).

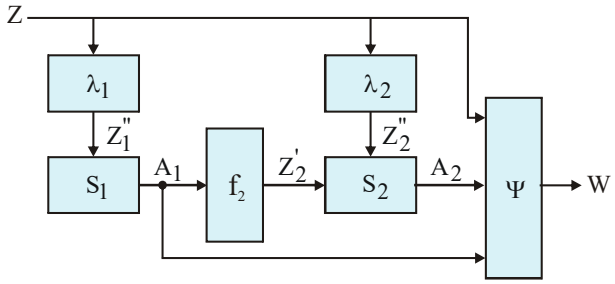



Рис. 12. Сеть n-компонентных автоматов

Сеть, содержащая два автомата, задается вектором:

$$N = (Z, \{S_1, S_2\}, W, \{f_1, f_2\}, \{\lambda_1, \lambda_2\}, \Psi).$$

Здесь параметры сети представлены в виде: 1) Входной алфавит Z={reset}. 2) Множество компонентных автоматов сети $S_i = \{S_1, S_2\}$ :КА$_1$: $S_1 = (A_1, Z_1, \varphi_1)$ ,КА$_1$: $S_2 = (A_2, Z_2, \varphi_2)$; 3) Функции переходов для данных автоматов: $S_1\,\varphi_1 : A_1 \times Z_1 \to A_1$ $\delta_1 : A_1 \times \{\text{reset}\} \to A_1$; $S_2\,\varphi_2 : A_2 \times Z_2 \to A_2$ $\delta_2 : A_2 \times \{\text{rst}, \text{en}\} \to A_2$. 4) W – выходной алфавит сети. 5) Множество функций соединения компонентных автоматов сети представлено одной функцией: $f_1 = \varnothing$, $f_2 : A_1 \to Z_2^{'}$. 6) Множество входных функций : $\lambda_1 : Z_1 = Z_1^{''} = \{\text{reset}\}$; многокомпонентных автоматов сети $\lambda_2 : Z_2 = Z_2^{'} \times Z_2^{''} = \{\text{rst}, \text{en}\} \times \{\text{reset}\}$; 6) Выходная функция сети; $\psi : (A_1 \times A_2) \times Z \to W$.

Структурная модель сети, содержащей два автомата, представлена на рис. 13.

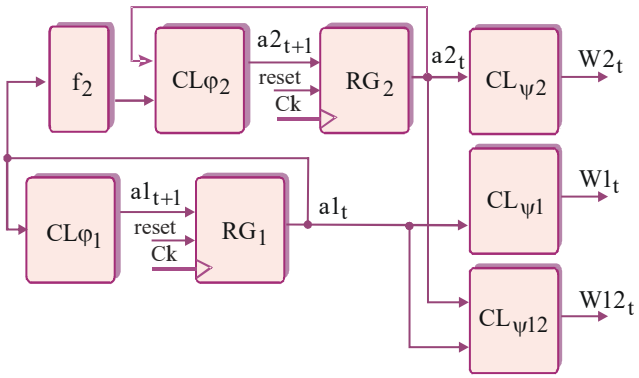



Рис. 13. Структурная модель сети автоматов

Блоки $CL\varphi_1$ и $RG_1$ реализуют первый счетчик Сч1, блоки $CL\varphi_2$ и $RG_2$ – второй счетчик Сч2. Блок $f_2$ является комбинационным и реализует функцию соединения двух компонентных автоматов Сч1 и Сч2. Совокупный автомат содержит три вида выходных функций: $\psi_1$ – зависит от состояния автомата Сч1; $\psi_2$ – от состояния автомата Сч2; $\psi_{12}$ – от состояний автоматов Сч1 и Сч2:

$$w_1(t) = \psi_1(a1(t)); \; w_2(t) = \psi_2(a2(t)); \; w_{12}(t) = \psi_{12}(a1(t), a2(t)).$$

Функциям $\psi_1, \psi_2$ и $\psi_{12}$ соответствуют комбинационные блоки $CL\psi_1, CL\psi_2$ и $CL\psi_{12}$.

Также как и для автомата с одним счетчиком, минимальный рабочий период будет равен максимальному пути:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}).$$

1) Комбинационный (логический) путь "Pin to Pin" проходит от асинхронного сброса reset до выходов W:

$$T_{PP} = \max[(T_{RG1\_reset} + T_{CL_{\psi 1}\max}), (T_{RG2\_reset} + T_{CL_{\psi 1}\max}), (T_{RG\_reset_{max}} + T_{CL_{\psi 12}\max})].$$

2) Путь "Register to Register" определяет задержку комбинационного пути между регистрами счетчика:

$$T_{RR} = \max[(T_{RG1} + T_{CL_{\varphi 1 \max}} + T_{su}), (T_{RG2} + T_{CL_{\varphi 2 \max}} + T_{su}), (T_{RG1} + T_{f2_{max}} + T_{su})].$$

3) Путь "Clock to Output" определяется компонентами:

$$T_{CO} = \max[(T_{RG1} + T_{CL_{\psi 1}\max}), (T_{RG2} + T_{CL_{\psi 2}\max}), (T_{RG_{max}} + T_{CL_{\psi 12}\max})].$$

При реализации устройства на микросхемах программируемой логики PLD или ASIC все триггеры однотипны и поэтому имеют одинаковое время установки Tsu и задержки переключения $T_{RG}$. Тогда представленные выше формулы трансформируются к виду:

$$T_{PP} = \max[(T_{RG1\_reset} + T_{CL_{\psi 1}\max}), (T_{RG2\_reset} + T_{CL_{\psi 1}\max}), (T_{RG\_reset_{max}} + T_{CL_{\psi 12}\max})] =$$

$$= T_{RG\_reset} + \max(T_{CL_{\psi 1}\max}, T_{CL_{\psi 2}\max}, T_{CL_{\psi 12}\max}).$$

$$T_{CO} = T_{RG} + \max(T_{CL_{\psi 1}\max}, T_{CL_{\psi 2}\max}, T_{CL_{\psi 12}\max}).$$

Если триггеры регистров RG1 и RG2 переключаются по одному фронту, то имеет место формула:

$$T_{RR} = T_{RG} + T_{su} + \max(T_{CL_{\varphi 1 \max}}, T_{CL_{\varphi 2 \max}}, T_{f2_{max}}).$$

Если триггеры регистров RG1 и RG2 переключаются по разным фронтам, тогда действительно выражение:

$$T_{RR} = T_{su} + \max[(T_{RG} + T_{CL_{\varphi 1 \max}}), (T_{RG} + T_{CL_{\varphi 2 \max}}), 2(T_{RG} + T_{f2_{max}})].$$

Для описания алгоритма функционирования предложенных системных иерархических моделей управления конвейерными вычислениями DSP-преобразования традиционные формы синтеза на основе графа или таблицы переходов не являются эффективными. Исходной информацией для системных иерархических моделей может быть структура, состоящая из следующих параметров:

Модель 1: 1) $Cnt_{min}$ и $Cnt_{max}$ минимальная и максимальная границы счета; 2) $Cnt_{S0}$ – начальное состояние; 3) $N_F$ – количество выходных функций; 4) тип выходов: комбинацион-

ный или регистровый; 5)множество выходных функций $\Psi = \{\psi_1, \psi_2, ..., \psi_{N_F}\}$. Модель 2: 1) $Cnt1_{min}$ и $Cnt1_{max}$ – минимальная и максимальная границы счета счетчика 1,

$Cnt2_{min}$ и $Cnt2_{max}$ – минимальная и максимальная границы счета счетчика 2; 2) $Cnt1_{S0}$ и $Cnt2_{S0}$ – начальное состояние счетчиков 1 и 2; 3) $N_F$ – количество выходных функций; 4) тип выходов: комбинационный или регистровый; 5) множество выходных функций $\Psi = \{\psi_1, \psi_2, ..., \psi_{N_F}\}$; 6) f2 – функции соединения компонентных автоматов сети.

На рис. 14 схематично представлен способ создания HDL-кода для модели 1. Блоки $CL_\varphi$ и RG, соответствующие счетчику, могут быть реализованы одним процессом для VHDL-модели или одним блоком always для Verilog. Для реализации выходных функций используются параллельные условные операторы (VHDL) или условный оператор assign (Verilog). Шаблоны VHDL- и Verilog-кода для модели 1 представлены листингами 3 и 4 соответственно.

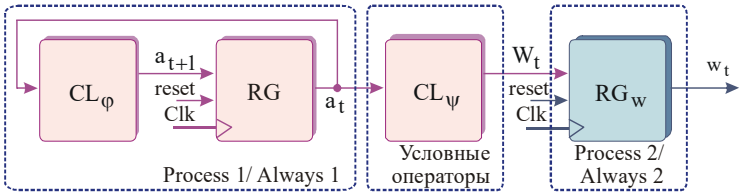



Рис. 14. Модель управляющего автомата с регистровыми выходами

Листинг 3. VHDL-шаблон для реализации модели 1

```
library IEEE;
use  IEEE.std_logic_1164.all;
-- Пакеты, поддерживающие арифметические операции над данными std_logic_vector
use  ieee.std_logic_unsigned.all;
entity Имя_интерфейса is
 port (En, Clk, reset: in STD_LOGIC;
          Выходы Wi или W_i_reg: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
     -- Разрядность счетчика
constant m: natural:= ...;
     -- Нижняя граница счетчика
constant Cnt_min: std_logic_vector(m-1 downto 0):=...;
         -- Верхняя граница счета
constant Cnt_max: std_logic_vector(m-1 downto 0):=...;
         -- Состояние сбросв
constant Cnt_S0: std_logic_vector(m-1 downto 0):=...;
     signal count1: std_logic_vector(m-1 downto 0);
     --Определение сигналов Wi,
     -- Если выходы устройства регистровые
     signal Wi : std_logic;
begin
-- Модель счетчика
Process_1 : process(reset, En, clk)
          begin
               if reset='1' then
     -- Сброс счетчика в начальное состояние
                    count1 <= Cnt_S0;
     -- Описание проверки фронта синхросигнала
               elsif clk='1'and clk'event then
-- Или задний фронт elsif clk='0'and clk'event then
                    if en='1' then
     -- Проверка верхней границы счета
                         if count1 = Cnt_max then
     -- Присвоение нижней границы счета
```

```
                    count1 <= Cnt_min;
                else
                    count1 <= count1 + '1';
                end if;
            end if;
        end if;
    end process;
-- Реализация выходных функций с помощью
-- условных параллельных операторов
W_i <= '1' when условие else
    '0';
. . .
-- Процесс для реализации регистровых выходов
Process_2: process(reset, clk)
    begin
        if reset='1' then
    -- Сброс в нулевое состояние
            W_i_reg <= (others =>'0');
            . . .
-- Описание проверки фронта синхросигнала
        elsif clk='1'and clk'event then
-- Или задний фронт elsif clk='0'and clk'event then
            W_i_reg <= W_i;
            . . .
        end if;
    end process;
end Имя_арх;
```

Листинг 4. Verilog-шаблон для реализации модели 1

```
module Имя_интерфейса (En, Clk, reset, Выходы Wi или W_i_reg)
    input En, Clk, reset;
    output Выходы Wi;
    // Если выходы комбинационные
    // или, если выходы регистровые
    //   output Выходы W_i_reg;    reg Выходы W_i_reg; wire Выходы Wi;
        // Параметры конфигурации устройства
    parameter m = ...;          // разрядность счетчика
    parameter Cnt_min =...; // нижняя граница
    parameter Cnt_max =...; // верхняя граница
    parameter Cnt_S0 =...;   // состояние сбросов
        reg[m-1:0]    count1;
begin
// Являющийся моделью счетчика блок always 1
always @(posedge reset or posedge En or posedge clk)
    if (reset)
    // Сброс счетчика в начальное состояние
        count1 <= Cnt_S0;
    else if (en)
    // Проверка верхней границы счета
            if (count1 == Cnt_max)
    // Присвоение нижней границы счета
                count1 <= Cnt_min;
            else
                count1 <= count1 + 1'b1;
// Реализация выходных функций с помощью условных операторов
assign W_i = условие ? 1'b1: 1'b0;
. . .
// Блок always 2, для регистровых выходов
always @(posedge CLk or negedge Reset)
    begin
        if (Reset) W_i_reg <= 'b0;
```

```
        else W_i_reg <= W_i;
        . . .
    end
endmodule
```

Принцип построения HDL-кода для иерархической модели 2 представлен на рис. 15.

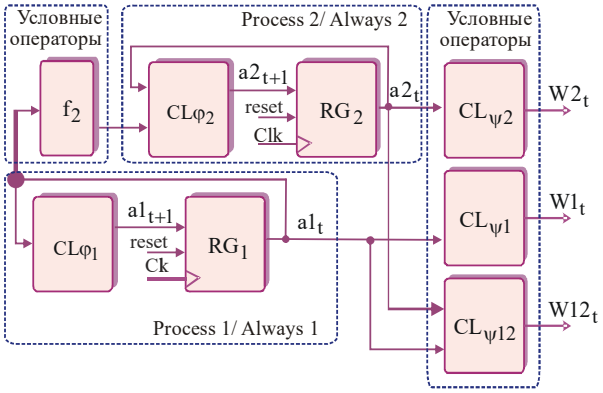



Рис. 15. Структурная модель сети автоматов

Каждому счетчику, реализуемому блоками $CL_\varphi$ и RG, ставится в соответствие процесс для VHDL-модели или блок always для Verilog. Таким образом, счетчик 1 кодируется оператотом Process 1 или Always 1, счетчик 2 – оператором Process 2 или Always 2. Для реализации выходных функций используются параллельные условные операторы (VHDL) или операторы assign (Verilog). Условные операторы также применяются для реализации функций соединения компонентных автоматов сети $f_2$. Шаблоны VHDL- и Verilog-кода для модели 1 представлены листингами 5 и 6 соответственно. Условия в операторах для функций $W1_t$ зависят только от состояния счетчика 1, а для функций $W2_t$ – от состояния счетчика 2, условия функций $W2_t$ – от значений обоих счетчиков. Для реализации регистровых выходных функций HDL-модель может быть дополнена соответственным оператором Process или Always, как это сделано в модели 1.

Листинг 5. VHDL-шаблон для реализации модели 2

```
library IEEE;
use  IEEE.std_logic_1164.all;
-- Пакеты, поддерживающие арифметические
-- операции над данными std_logic_vector
use  ieee.std_logic_unsigned.all;
entity Имя_интерфейса is
 port (En, Clk, reset: in STD_LOGIC;
          Выходы W1i, W2i, W12i: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
     -- Разрядность счетчика 1
constant m1: natural:= ...;
     -- Разрядность счетчика 2
constant m2: natural:= ...;
     -- Нижняя граница счетчика
constant Cnt1_min: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_min: std_logic_vector(m-1 downto 0):=...;          -- Верхняя граница счета
constant Cnt1_max: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_max: std_logic_vector(m-1 downto 0):=...;
          -- Состояние сброса
constant Cnt1_S0: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_S0: std_logic_vector(m-1 downto 0):=...;           signal  count1:
```

```
std_logic_vector(m1-1 downto 0);
    signal count2: std_logic_vector(m2-1 downto 0);
    -- Сигналы, формируемые функцией f2
    signal en2, rst2: std_logic;
begin
-- Процесс счетчика 1
Process_1 : process(reset, En, clk)
    begin
        if reset='1' then
        -- Сброс счетчика в начальное состояние
            count1 <= Cnt1_S0;
    -- Описание проверки фронта синхросигнала
        elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and clk'event then
                if en='1' then
    -- Проверка верхней границы счета
                    if count1 = Cnt1_max then
    -- Присвоение нижней границы счета
                        count1 <= Cnt1_min;
                    else
                        count1 <= count1 + '1';
                    end if;
                end if;
            end if;
        end  process;
-- Процесс счетчика 2
Process_2 : process(reset, En, clk, rst2, en2)
    begin
        if (reset='1')or(rst2='1') then
        -- Сброс счетчика в начальное состояние
            count2 <= Cnt_S0;
    -- Описание проверки фронта синхросигнала
            elsif clk='1'and clk'event then
 -- Или задний фронт elsif clk='0'and clk'event then
                if (en='1') and (en2='1') then
    -- Проверка верхней границы счета
                    if count2 = Cnt2_max then
    -- Присвоение нижней границы счета
                        count2 <= Cnt2_min;
                    else
                        count2 <= count2 + '1';
                    end if;
                end if;
            end if;
        end  process;
-- Функции соединения компонентных автоматов сети f2
    rst2 <= '1' when условие else
        '0';
    en2    <= '1' when условие else
        '0';
-- Реализация выходных функций условными
--  параллельными  операторами
    W1_i <= '1' when условие else
            '0';
    . . .
    W2_i <= '1' when условие else
        '0';
    . . .
    W12_i <= '1' when условие else
        '0';
    . . .
```

```
end Имя_арх;
```

Листинг 6. Verilog-шаблон для реализации модели 2

```
module Имя_интерфейса (En, Clk, reset, Выходы W1i, W2i, W12i)
    input En, Clk, reset;
    output Выходы W1i, W2i, W12i;
    // Параметры конфигурации счетчика 1
    parameter Cnt1_min =...;    // нижняя граница
    parameter Cnt1_max =...;    // верхняя граница
    parameter Cnt1_S0 =...;     // состояние сбросв
    parameter m1 = ...;       // разрядность счетчика
    reg[m1-1:0] count1;
        // Параметры конфигурации счетчика 1
    parameter Cnt2_min =...;    // нижняя граница
    parameter Cnt2_max =...;    // верхняя граница
    parameter Cnt2_S0 =...;     // состояние сбросов
    parameter m2 = ...;       // разрядность счетчика
    reg[m2-1:0] count2;
    // Сигналы, формируемые функцией f2
    wire en2, rst2;
// Счетчик 1, блок always 1
always @(posedge reset or posedge En or posedge Clk)
    if (reset)
        // Сброс счетчика в начальное состояние
        count1 <= Cnt1_S0;
    else if (En)
        // Проверка верхней границы счета
            if (count1 == Cnt1_max)
                count1 <= Cnt1_min;
            else
                // Присвоение нижней границы счета
                count1 <= count1 + 1'b1;
    // Счетчик 2, блок always 2
always @(posedge reset or posedge En or posedge Clk or posedge en2 or posedge rst2)
    if (reset||rst2)
        // Сброс счетчика в начальное состояние
        count2 <= Cnt2_S0;
    else if (En)
        // Проверка верхней границы счета
            if (count1 == Cnt2_max)
                count1 <= Cnt2_min;
            else
        // Присвоение нижней границы счета
                count2 <= count2 + 1'b1;
 // Функции соединения компонентных автоматов сети f2
    assign rst2 = условие ? 1'b1: 1'b0;
    assign  en2   = условие ? 1'b1: 1'b0;
// Реализация выхондых функций с помощью
// условных операторов
    assign W1_i = условие ? 1'b1: 1'b0;
    . . .
    assign W2_i = условие ? 1'b1: 1'b0;
    . . .
    assign W12_i = условие ? 1'b1: 1'b0;
        . . .
endmodule
```

### 9. Синтез HDL-модели автомата для DSP-фильтра

Управляющий блок фильтра для обработки изображений размерностью $256 \times 256$ пикселов имеет следующие параметры: 1) минимальная и максимальная границы счета счетчика 1: $Cnt1_{min}$=1 и $Cnt1_{max}$ =256; Минимальная и максимальная границы счета счетчика 2: $Cnt2_{min}$ =1 и $Cnt2_{max}$ =258. 2) Начальное состояние счетчиков 1 и 2: $Cnt1_{S0} = 0$ и $Cnt2_{S0} = 0$. 3) Количество выходных функций $N_F$ =3. 4) Тип выходов – комбинационный. 5) Множество выходных функций:

$$\Psi = \{\psi_1, \psi_2, \psi_3\} = \{\text{First}, \text{Last}, \text{Enable}\};$$

$$\text{First} = \begin{cases} 1, & \text{count2} = 2; \\ 0, & \text{else.} \end{cases} \quad \text{Last} = \begin{cases} 1, & \text{count2} = \text{Cnt2}_{\min} - 1; \\ 0, & \text{else.} \end{cases}$$

$$\text{Enable} = \begin{cases} 1 & (\text{count2} > 1) \,\&\, (\text{count2} <= \text{Cnt2}_{\max} - 1); \\ 1 & (\text{count2} = \text{Cnt2}_{\max}) \text{ и } (\text{count1} = 1); \\ 0 & \text{else.} \end{cases}$$

$$W1_i = \varnothing, \; W2_i = \{\text{First}, \text{Last}\}, \; W12_i = \{\text{Enable}\}.$$

Здесь count1 – счетчик 1, count2 – счетчик 2. 6) Функции соединения компонентных автоматов сети f2: $f_2 = \{en2\}$.

VHDL и Verilog-модели управляющего блока представлены листингами 7 и 8.

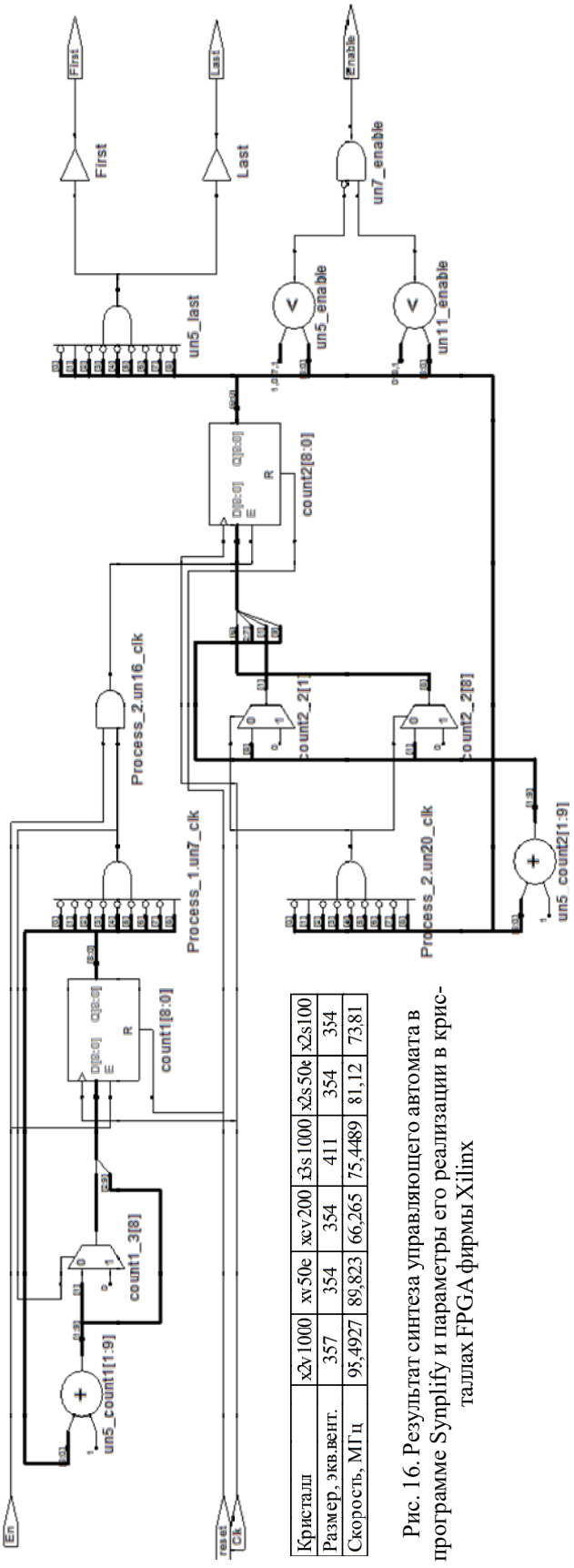
Листинг 7. VHDL-модель управляющего блока

```
library IEEE;
use  IEEE.std_logic_1164.all;
use  ieee.std_logic_unsigned.all;
use  ieee.std_logic_arith.all;
entity Control1 is
 port (En, Clk, reset: in STD_LOGIC;
          First, Last, Enable: out STD_LOGIC);
end entity Control1;
architecture Control1 of Control1 is
-- разрядность счетчика 1
constant m1: natural:= 9;
-- разрядность счетчика 2
constant m2: natural:= 9;
     -- нижняя граница счетчика
constant Cnt1_min: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(1, m1);
constant Cnt2_min: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(1, m2);
     -- верхняя граница счета
constant Cnt1_max: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(256, m1);
constant Cnt2_max: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(258, m2);
     -- состояние сбросв
constant Cnt1_S0: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(0, m1);
constant Cnt2_S0: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(0, m2);
signal count1: std_logic_vector(m1-1 downto 0);
signal count2: std_logic_vector(m2-1 downto 0);
     -- сигналы, формируемые функцией f2
signal en2: std_logic;
begin
-- Процесс счетчика 1
Process_1 : process(reset, En, clk)
          begin
               if reset='1' then
     -- сброс счетчика в начальное состояние
                    count1 <= Cnt1_S0;
     -- описание проверки фронта синхросигнала
               elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and clk'event then
                    if en='1' then
```

```
        -- проверка верхней границы счета
                    if count1 = Cnt1_max then
                        count1 <= Cnt1_min;
                    else
        -- присвоение нижней границы
счета
                        count1 <= count1 + '1';
                    end if;
                end if;
            end if;
        end process;
-- Процесс счетчика 2
Process_2 : process(reset, En, clk, en2)
        begin
            if (reset='1') then
    -- сброс счетчика в начальное состоя-
ние
                count2 <= Cnt2_S0;
    -- описание проверки фронта синхро-
сигнала
            elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and
clk'event then
                if (en='1') and (en2='1') then
                -- проверка верхней границы
счета
                    if count2 = Cnt2_max then
                -- присвоение нижней
границы счета
                        count2 <= Cnt2_min;
                    else
                        count2 <= count2 + '1';
                    end if;
                end if;
            end if;
        end process;
-- Функции соединения компонентных
автоматов сети f2
    en2    <= '1' when count1 = Cnt1_max
else
        '0';
-- реализация выходных функций условны-
ми
--  параллельными  операторами
    First <= '1' when count2=2 else '0';
    Last <= '1' when count2=Cnt2_min+'1'
else '0';
    Enable <= '1' when ((count2>1) and
(count2<=Cnt2_max-1) ) or ((count2 =
Cnt1_max) and (count1=1)) else   '0';
end Control1;
```

Листинг 8. Verilog-модель управляющего блока

```
module Control1 (En, Clk, reset, First, Last,
Enable);
    input En, Clk, reset;
    output First, Last, Enable;
    // параметры конфигурации счетчика 1
    parameter Cnt1_min = 'd1;  // нижняя
```




| Кристалл | x2v1000 | xv50e | xcv200 | i3s1000 | x2s50e | x2s100 |
|---|---|---|---|---|---|---|
| Размер, экв.вент. | 357 | 354 | 354 | 411 | 354 | 354 |
| Скорость, МГц | 95,4927 | 89,823 | 66,265 | 75,4489 | 81,12 | 73,81 |

Рис. 16. Результат синтеза управляющего автомата в программе Synplify и параметры его реализации в кристаллах FPGA фирмы Xilinx

```
граница
    parameter Cnt1_max = 'd256;// верхняя граница
    parameter Cnt1_S0 = 'd0;   // состояние сбросв
    parameter m1 = 8;          // разрядность счетчика
    reg[8:0] count1;
    // параметры конфигурации счетчика 1
    parameter Cnt2_min = 'd1;  // нижняя граница
    parameter Cnt2_max = 'd258;// верхняя граница
    parameter Cnt2_S0 = 'd0;   // состояние сбросов
    parameter m2 = 8;    // разрядность счетчика
    reg[8:0] count2;
    // сигналы, формируемые функцией f2
    wire en2;
    //wire en_in;//, rst_in;
// Счетчик 1, блок always 1
always @(posedge reset or posedge Clk)
if (reset)    // сброс счетчика в начальное состояние
        count1 <= Cnt1_S0;
else
    begin
        if (En)
        // проверка верхней границы счета
        begin
            if (count1 == Cnt1_max)
                count1 <= Cnt1_min;
            else
            // присвоение нижней границы счета
                count1 <= count1 + 1'b1;
     end
    end
// Счетчик 2, блок always 2
always @(posedge reset or posedge Clk)
    if (reset)
        // сброс счетчика в начальное состояние
        count2 <= Cnt2_S0;
    else
    begin
        if (en2)
        // проверка верхней границы счета
        begin
            if (count2 == Cnt2_max)
                count2 <= Cnt2_min;
            else
                // присвоение нижней границы счета
                count2 <= count2 + 1'b1;
        end
    end
// Функции соединения компонентных автоматов сети f2
assign en2= (En &&(count1 == Cnt1_max))?1'b1: 1'b0;
// реализация выходных функций с помощью
// условных операторов
assign First = (count2 == 2) ? 1'b1: 1'b0;
assign Last = (count2 == Cnt2_min+1) ? 1'b1: 1'b0;
assign Enable = ((count2>1)&&(count2<=Cnt2_max-1))|| ((count2==Cnt2_max)&&(count1==1))?
1'b1: 1'b0;
endmodule
```

Структурная схема управляющего автомата для конвейерного DSP-преобразователя, сгенерированная программой синтеза Synplify, представлена на рис. 16. При синтезе использовались микросхемы программируемой логики FPGA фирмы Xilinx [12].

## 10. Генератор HDL-кода моделей автоматов

На основе предложенной системной иерархической модели управляющего автомата разработана программа автоматической генерации VHDL- и Verilog-кодов блока управления для вычислителя с конвейерной архитектурой. При создании программного продукта использовался скриптовый язык пакета Matlab [13].

Интерфейс программы генерирования управляющих автоматов для конвейрных архитектур представлен на рис. 17. Выбор языка для генерирования модели и ее тип определяются входными параметрами, реализованными с помощью переключателей: HDL selection – выбор языка для генерирования кода; Number of Counters – модель с одним или двумя счетчиками. Для каждого счетчика задается нижняя Min и верхняя Max границы счета, значение параметра инициализации Init. Для модели с одним счетчиком параметры панели Counter 2 игнорируются. Панель F2 function предназначена для создания функций соединения компонентных автоматов. Программа позволяет задавать две функции En2 и Rst2. Первая подключается к входу разрешения синхронизации, а вторая – к сбросу устройства в начальное состояние. Для того чтобы задать выходные функции, необходимо указать их количество. Затем для каждой функции следует ввести ее имя (Name) и условие формирования единичного значения (Conditions of 1 value). Условия должны быть записаны с использование синтаксиса HDL-языка, который выбран для генерирования модели. Текущее значение счетчика 1 обозначается переменной u1, счетчика 2 – u2.

Для разработки приложения использовался инструмент GUIDE( MATLAB® Graphical User Interface development environment) программного пакета MATLAB. Полный код программы генерирования содержит 1110 строк М-скрипта. Генератор работает в программной среде MATLAB, операционная система Windows.

## 11. Выводы

Предложены две модифицированные модели управляющих автоматов, предназначенные для использования в устройствах с конвейерной архитектурой. Приведено их математическое и структурное описание. Разработаны HDL-шаблоны реализации разработанных моделей управления для DSP-преобразователей.

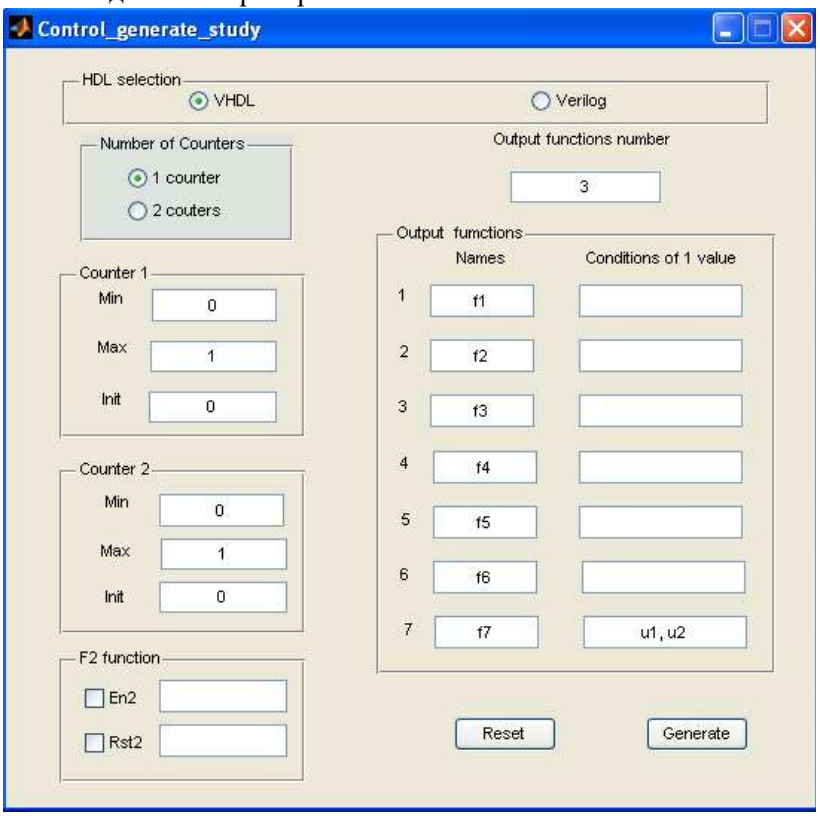



Рис. 17. Интерфейс программы генерирования HDL-модели

Созданные модели предназначены для использования в DSP-устройствах с конвейерной архитектурой. В отличие от существующих классических моделей предложенные структуры не используют таблицы переходов для исходного описания поведения автомата.

Предложена технология системного проектирования, которая позволяет автоматизировать процесс синтеза цифровых устройств на кристаллах с конвейерной архитектурой. Создан программный продукт, который выполняет автоматический синтез VHDL и Verilog-кода предложенных моделей управляющих автоматов. Генераторы моделей автоматов упрощают процесс проектирования DSP-преобразователей, позволяют эффективно создавать и верифицировать системные HDL-модели управляющего блока.

**Список литературы: 1.** *Philip E. Ross.* 5 Commandments. IEEE Spectrum. December. 2003. P. 30-35. **2.** *Баранов С.И.* Синтез микропрограммных автоматов. Л.: Энергия, 1979. 232 с. **3.** *Соловьев В.В.* Проектирование цифровых автоматов на основе программируемой логики интегральных схем. М.: Горячая линия. Телеком. 2001. 636 с. **4.** *Solowjew W., Chyzy M.* Synteza automatow skonczonych na ukladah PAL . Electronika. XXXVII. 1996. No 10. P.23 - 27. **5.** *Solovjev V., Mazalewski J., Chyzy M.* Models of robotics control systems on programmable logic devices. Proc. of the 4th Int. Symposium on Methods and Models an Automation and Robotics (MMAR'97). August 1997. Miedzyzdroje. Poland. Vol. 3. P. 1019 – 1024. **6.** *Кеэваллик А.Э.* Теорема декомпозиции конечных автоматов. Автоматика и вычислит. техника. 1974. № 1. С. 77-81. **7.** *Hartmanis J., Sterns R.* Algebraic Structure Theory of Sequential Machines. New York, Prentice-Hall. 1966. 464 p. **8.** *Хаханов В.И., Хаханова И.В.* VHDL + Verilog = Синтез за минуты. Харьков: СМИТ. 2007. 264 с. **9.** *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ. 2003. 492 с. **10.** *Charles H. Roth, Jr.* Digital Systems Design UsingVHDL. Boston. PWS Publishing Company. 1998. 470 p.**11.** *Ashenden, Peter J.* The designer's guide to VHDL. San Francisco. Calis. California. Morgan Kaufmann Publishers, Inc. 1996. 688 p. **12.** Xilinx.com. **13.** www.mathworks.com

*Поступила в редколлегию 23.08.2007*

**Хаханова Ирина Витальевна**, докторантка кафедры АПВТ ХНУРЭ. Научные интересы: Проектирование цифровых систем на кристаллах. Увлечения: английский язык, музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanova@mail.ru

УДК 004.5; 004.7; 004.8

*А.Я. КУЗЕМИН, В.М. ЛЕВЫКИН*

# АДАПТИВНЫЙ МЕТОД РАЗРАБОТКИ СТРУКТУРЫ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ К УСЛОВИЯМ ЕЕ ИСПОЛЬЗОВАНИЯ

Предлагается адаптивный метод разработки обеспечивающего комплекса информационно аналитической системы (ИАС), который отличается от существующих более качественной оценкой ситуаций в заданный срок и с минимальными экономическими и социальными рисками в чрезвычайных природных ситуациях, что позволяет получить адаптивную к условиям чрезвычайных природных ситуаций структуру обеспечивающего комплекса ИАС.

**Актуальность.** В настоящее время для жизнедеятельности человека в природной сфере постоянно возникают опасности и угрозы. Это происходит, когда характеристики природных процессов и явлений достигают и превышают определенный критический предел, после чего природный процесс выходит из нормального состояния. Такой процесс может сопровождаться разрушительным или другим негативным воздействием на окружающую среду, приводящим к природному бедствию различной интенсивности и масштаба - источнику чрезвычайной природной ситуации (ЧПС). ЧПС - это обстановка на определенной территории, сложившаяся в результате опасного природного явления, катастрофы, стихийного бедствия, которая может повлечь за собой человеческие жертвы, ущерб здоровью людей или природной среде, значительные материальные потери и нарушение условий жизнедеятельности людей.

Для ЧПС характерны следующие особые явления: геофизические, геологические, метеорологические, агрометеорологические, морские гидрологические, гидрологические, природные пожары, инфекционная заболеваемость людей и сельскохозяйственных животных, поражение сельскохозяйственных растений болезнями и вредителями. Близкими к ЧПС являются экологические чрезвычайные ситуации, вызванные экологическими обратимыми и необратимыми явлениями. Результатом ЧПС является вред, наносимый природе, человеку, объектам экономики, социальной сферы, окружающей природной среде, а также изменения обстановки, произошедшие вследствие этого. В этой связи *актуальным* является мониторинг и прогнозирование ЧПС - наблюдение, контроль и предвидение опасных процессов и явлений природы, являющихся источниками чрезвычайных ситуаций, а также динамики их развития, определения их масштабов в целях решения задач предупреждения и организации ликвидации бедствий.

Для этого мониторинга и прогнозирования ЧПС в условиях сложной и изменчивой внешней природной среды необходимо использовать информационно-аналитические системы (ИАС). Структура обеспечивающего комплекса ИАС прежде всего должна быть *гибкой* и *адаптивной*. ИАС должна быть приспособлена к определению новых проблем и выработке новых решений в большей степени, чем к контролю уже принятых решений и их реализации. В ней должна быть обеспечена возможность *максимальной концентрации* всех ресурсов, объединения информационных, организационных и других типов резервов для ликвидации в кратчайшие сроки создавшейся экстремальной ситуации.

В ИАС в условиях ЧПС должны быть реализованы два, казалось бы, взаимоисключающих принципа: специализация и универсализм. Структура обеспечивающего комплекса ИАС должна быть *структурой с локальной автономией и глобальной координацией*. Ее различные комплексы участвуют в достижении локальных целей и решении задач. Они должны удовлетворять требованиям функционального комплекса ИАС по затратам времени и средств на разработку ИАС и обеспечить минимальный экономический и социальный риск при достижении глобальной цели функционирования ИАС.

*Гибкость и адаптивность* обеспечивающего комплекса ИАС, необходимая в условиях ЧПС, ориентирована на принятие эффективных мер при воздействии широкого спектра возможных неблагоприятных изменений с минимумом экономических и социальных рисков.

**Постановка задачи.** Необходимо разработать адаптивный метод разработки обеспечивающего комплекса ИАС для предупреждения и ликвидации последствий ЧПС.

**Новизна.** Предложен адаптивный метод разработки обеспечивающего комплекса ИАС, который отличается от существующих более качественной оценкой ситуаций в заданный срок и с минимальными экономическими и социальными рисками в ЧПС. Метод позволяет получить адаптивную к условиям ЧПС структуру обеспечивающего комплекса ИАС и своевременный прогноз развития ЧПС с минимальным экономическим и социальным риском.

**Решение.** В практике проведения работ в области разработки обеспечивающего комплекса ИАС с оценкой экономического и социального риска можно использовать как классические, так и оригинальные методы оценки риска.

Можно представить себе природную ситуацию (рис.1) контролируемого участка, как некоторую среду, которой требуется выделить ресурс $\Sigma(t)$ для предупреждения или ликвидации последствий ЧПС $Z = f(\Sigma(t-\tau))$. На вход системы поступает поток данных о состоянии природной среды X(t) (температура воздуха, влажность, давление, характеристики контролируемого горного склона и т.п.). На выходе (см. рис.1) показаны принятые для использования в данной ситуации спасательные средства, план выделения технических средств для эвакуации, количество необходимых заградительных строительных конструкций и т. п. – $\Sigma(t)$. Основная цель мониторинга природной ситуации – защита от неблагоприятных природных явлений. Лицо, принимающее решение (ЛПР) выступает как некий регулятор f. Конечно, он не совершенен, в частности, он реагирует на природную ситуацию с запаздыванием $\tau$.

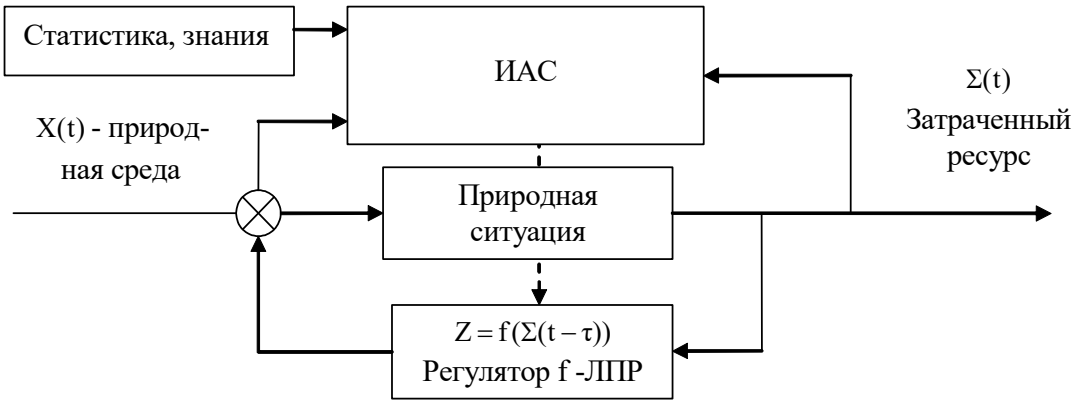



Рис.1. Модель мониторинга природной ситуации

В каких случаях регулятор *f* плохо работает? Очевидно, тогда, когда запаздывание $\tau$ слишком велико, и чтобы защитить от опасности среду обитания и интересов человека и его самого, нужно действовать гораздо быстрее (со временем запаздывания $\tau_1 << \tau$). Такие ситуации возникают в случае ЧПС. Быстрая реакция предполагает затраты ресурса $\Sigma(t)$ или определенный набор действий (распознавание опасности, оказание срочной помощи: обычно людей надо немедленно эвакуировать, накормить, обогреть, доставить к врачу). Необходима незамедлительная реакция специальных организационно-технических структур, способных действовать гораздо быстрее остальных, управление ЛПР на основании работы ИАС, адаптированной к возникшей природной ситуации. ЛПР должно иметь набор отработанных и заготовленных сценариев действий в случае ЧПС [1]. В частности,

отряды спасательной службы министерства по чрезвычайным ситуациям (МЧС) должны в максимально короткий период времени после поступления сигнала выдвинуться в район ЧПС с необходимым инструментом, оборудованием, снаряжением и по прибытии туда немедленно приступить к выполнению задач. Уменьшение времени реагирования сейчас рассматривается как одно из главных критериев обеспечивающего комплекса ИАС, т.е. возникает необходимость иметь адаптированную к возникшей ситуации ИАС или еще один, более быстрый контур обратной связи (верхний прямоугольник на рис.1).

Если сравнить возникшую системную, природную ситуацию с организмом человека, то аналогом системы мониторинга природной среды является иммунная система человека, которая, во-первых, специфически распознает миллионы различных чужеродных молекул и реагирует на них. Во-вторых, отличает эти чужеродные молекулы от своих. В-третьих, различает разные группы внедряющихся микроорганизмов и "рассчитывает" свой ответ таким образом, чтобы эффективно очищать от них организм. Эти же задачи в ЧПС должна решать ИАС, которая использует в базе знаний три компонента: «***субъект*** (ЛПР) – рекомендуемое ***управляющее воздействие*** (предлагаемое ИАС для ЛПР) – ***объект*** (требуемый для предотвращения или устранения последствий, материально-технические и организационные ресурсы)». На самом деле, для рассматриваемых подобных систем можно выделить следующие характерные черты:

– Если бы система мониторинга природной среды, которая имела три (указанных выше) компонента или организм человека функционировал только в нормальном, "штатном" режиме, то обе системы были бы *не нужны* для управления ресурсами. Их компетенция была бы определена задачами штатного функционирования.

– Так как обе системы являются *распределенными*, то их структуры, реагирующие на угрозу, должны быть адаптированы к возникшей ситуации и мобильны, т.е. как можно ближе к источнику опасности.

– Для обеих структур критическим параметром является *время реакции*, время запаздывания. Многие тяжелые болезни иммунной системы связаны с возрастанием этого времени.

– Обе системы с точки зрения системного подхода появляются *на поздней стадии эволюции*. (Развитая иммунная система появилась только у позвоночных, развитые системы МЧС, выполняющие функции, отличные от пожарных и полицейских, тоже родились только в последнее время.)

– Для обеих хорошо организованных и работающих систем необходим системный анализ ситуации. Они должны быть адаптивны к возникающим ситуациям и иметь *подсистемы анализа, распознавания и прогнозирования*. Они должны действовать заблаговременно, с минимальными затратами времени и средств. Они должны обеспечивать минимальный риск получить нежелательные последствия возникшей ситуации.

Сформулированные выше требования к решению поставленной проблемы могут быть решены, если будут:

1. сформулированы критерии разработки обеспечивающего комплекса ИАС;
2. разработаны инструментальные средства проектирования, которые обеспечат адаптивные требования мониторинга природной среды.

Критериями разработки обеспечивающего комплекса ИАС могут служить показатели экономического, социального и индивидуального риска, время и затраты на принятие решений по управлению в текущей ситуации.

ЧПС характеризуются тремя основными параметрами: местом возникновения, временем воздействия и мощностью воздействия на окружающую среду. Риск также определяется тремя основными параметрами: местом ЧПС, временем до наступления ЧПС и способность противостоять чрезвычайной ситуации. Для своевременного определения этих параметров необходим мониторинг опасностей и рисков (рис.2).

Существует два основных вида мониторинга: глобальный и локальный. Кроме того, мониторинговые сети могут быть стационарными и мобильными, непрерывными и временными.

Как правило, стационарный мониторинг (который используется в глобальных целях) делается непрерывным и ведется в интересах детального уточнения необходимых параметров на элементах контролируемой системы.

Локальный мобильный мониторинг организуется в случае выявления возможной зоны ЧПС с повышенной опасностью и риском. В зависимости от вида опасности и риска производятся .необходимые мониторинговые наблюдения.

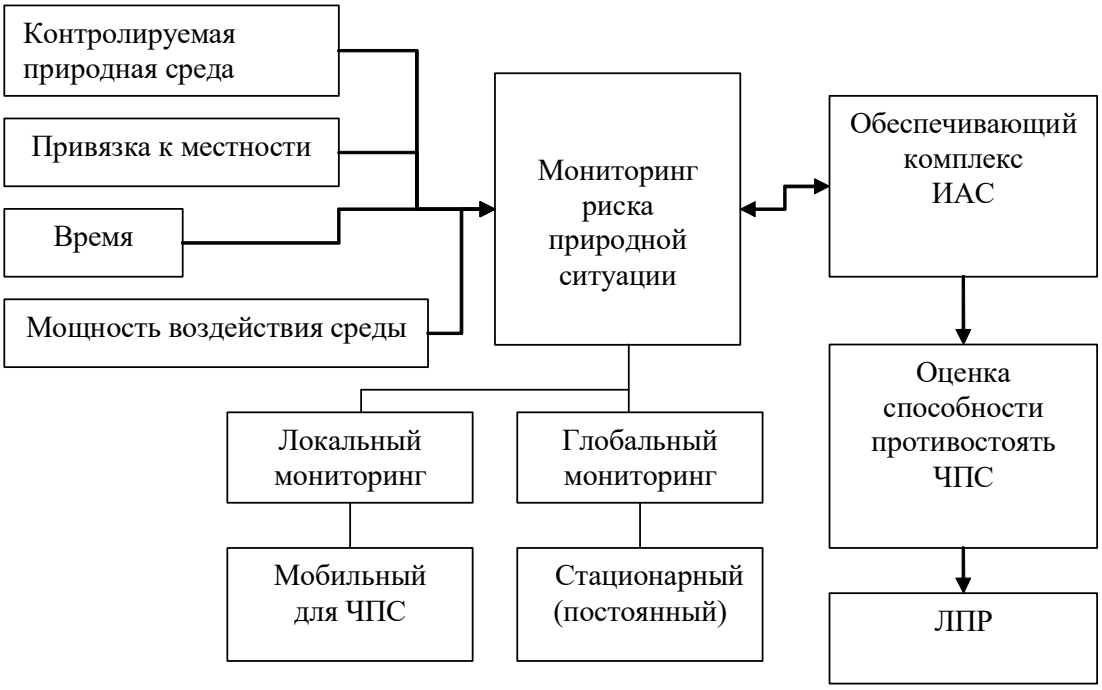



Рис.2. Алгоритм мониторинга риска

Нас интересуют мониторинговые наблюдения, которые носят универсальный характер, т.е. могут быть применены для оценки различных видов рисков.

Для оценки риска используют вероятностный подход [1]. Опасность выражается нагрузкой $P$, уязвимость системы – $U$. Предельное состояние системы характеризуется условием: $P - U < 0$.

Риск, соответствующий предельному состоянию системы, вычисляется как вероятность разности: $P - U < m > 0$, $R_e = \Phi(-m/\sigma_m)$, где $\Phi$ – функция Лапласа; $\sigma_m$ – среднеквадратичное отклонение; $R_e$ – величина риска.

Полученную величину сравнивают с нормируемыми величинами по пяти степеням. По ним определяют степень опасности или степень риска. Таким образом, для снижения возможных ущербов и рисков используют непрерывный мониторинг опасностей и уязвимости – это позволяет своевременно предупредить возникновение катастрофических рисков. По результатам мониторинговых наблюдений могут создаваться долгосрочные, среднесрочные и краткосрочные карты риска чрезвычайных ситуаций. На основе этих карт должны разрабатываться планы снижения риска чрезвычайных ситуаций.

В практике оценки индивидуального риска используют метод, при котором строят линии равного значения индивидуального риска (изолинии), рассчитанного по формуле:

$$R_u(x,y) = \sum_{m \in M} \sum_{l \in L} P_{Ql}(x,y) F(A_m),$$

где $P_{Ql}(x, y)$ – вероятность воздействия на человека в точке с координатами $(x, y)$ $Q_l$-го поражающего фактора с интенсивностью, соответствующей гибели (поражению) человека (здорового мужчины 40 лет) при условии реализации $A_m$ - го события (опасного природного явления, катастрофы, стихийного или иного бедствия); $F(A_m)$ – частота возникновения $A_m$ - го события в год; M - множество индексов, которое соответствует рассматриваемым событиям (опасным природным явлениям, катастрофам, стихийным или иным бедствиям); L – множество индексов, которые соответствуют перечню всех поражающих факторов, возникающих при рассматриваемых событиях.

*Социальный риск* - зависимость частоты возникновения событий, вызывающих поражение определенного числа людей, от этого числа людей. Результаты анализа риска изображаются в виде графиков (так называемых F/N-диаграмм). Социальный риск $R_c(N)$ характеризует масштаб возможных чрезвычайных ситуаций.

Социальный риск может быть рассчитан по формуле:

$$R_c(N) = \sum_{m \in M} \sum_{l \in L} P(N / Q_m) P(Q_m / A_l) F(A_l),$$

где $P(N / Q_m)$ – вероятность гибели (поражения) N людей от $Q_m$ -го поражающего фактора; $P(Q_m / A_l)$ – вероятность возникновения $Q_m$ -го поражающего фактора при реализации $A_l$ -го события (опасного природного явления, катастрофы, стихийного или иного бедствия).

В соответствии с приведенными зависимостями проблема комплексной оценки риска от чрезвычайных ситуаций природного и техногенного характера проводится на основе показателей социального риска (F/N и F/G диаграмм). На картах контролируемой местности отображаются изолинии индивидуального и социального риска ЧПС.

В основу разработки данного подхода положен метод «деревьев событий», основанный на представлении возможных сценариев возникновения и развития аварий и стихийных бедствий в виде графа. Каждая из вершин графа представляет собой некоторый этап развития аварии, характеризующийся количественными показателями (количество опасного вещества, участвующего в аварии; скорость ветра; интенсивность проявления поражающих факторов; плотность застройки; показатели защищенности людей и т.п.), а также вероятностью реализации рассматриваемого события. При этом каждая ветвь дерева событий представляет собой отдельный эффект (последовательность событий), который является точно определенным множеством функциональных взаимосвязей.

«Дерево событий» предоставляет возможность в строгой форме записывать последовательности событий и определять взаимосвязи между инициирующими и последующими событиями, сочетание которых приводит к чрезвычайной ситуации. Наиболее важные из рассматриваемых событий определяются или путем ранжирования, или путем количественного анализа.

В качестве комплексных показателей риска для населения и территории (с учетом используемых на практике определений индивидуального и социального риска) в данном случае рассматриваются:

а) частота гибели (в год) разного количества людей от всех природных и техногенных ЧС, характерных для региона;

б) частота возникновения (в год) материального ущерба различного масштаба от всех ЧПС, характерных для региона.

Построение деревьев событий для каждой из рассматриваемых чрезвычайных ситуаций и проведение расчетов с использованием деревьев событий позволяет оценивать частоту гибели людей и частоту возникновения материального ущерба различного масштаба. Получаемые при этом пары значений «частота – последствия» по каждому из рассмотренных сценариев, как правило, представляются в виде графиков (так называемых F-N и F-G диаграмм, где N – количество погибших или пострадавших людей, G - величина материального ущерба, F – частота рассматриваемого события).

Наличие F-N и F-G диаграмм по каждой из чрезвычайных ситуаций делает возможным построение интегральных показателей риска для населения регионов.

С использованием результатов комплексного анализа риска, представленных в виде F-N и F-G диаграмм, могут решаться многие практические задачи. Основными из них являются:

– ранжирование потенциально опасных объектов рассматриваемого региона по степени опасности для населения и территории;

– ранжирование чрезвычайных ситуаций различного происхождения (как природного, так и техногенного) между собой по степени опасности для населения и территории. Это позволяет выделить приоритетные направления в области разработки и реализации мероприятий по предупреждению чрезвычайных ситуаций на рассматриваемой территории в целом;

– исследование эффективности различных организационных и технических мероприятий для снижения уровней комплексного риска для населения и территорий;

– определение рациональной величины финансовых и материальных резервов для локализации и ликвидации чрезвычайных ситуаций;

– расчет величин страховых тарифов для страхования от чрезвычайных ситуаций, как для персонала отдельных промышленных предприятий, так и для населения отдельных населенных пунктов и территории региона в целом и др.

В 2000 г. в России была разработана новая технология комплексной оценки риска на основе использования географической информационной системы (ГИС) «Экстремум» [2] .

Для разработки обеспечивающего комплекса ИАС необходимо создать его модель, которая в общем виде имеет следующие составные части:

входные потоки данных $x_1, x_2, ..., x_d$ ; выходные каналы $y_1, y_2, ..., y_c$ и компоненты обеспечивающего комплекса ИАС $K_1, K_2, ..., K_p$ и взаимодействие $\Omega$ компоненты обеспечивающего комплекса ИАС их входных и выходных каналов [3].

Схема взаимодействия $\Omega$ представляет собой $c + p$ функций.

Под задачей разработки обеспечивающего комплекса ИАС будем понимать построение структуры обеспечивающего комплекса U, адаптивной к условиям использования, которая реализует задачи функционального комплекса ИАС - U с заданными ограничениями, т.е. нахождения отображений:

$$\begin{aligned} &1)\quad h: x_1 \times \ldots \times x_d \to x_1' \times \ldots \times x_d';\\ &2)\quad \xi: K_1 \times \ldots \times K_p \to K_1' \times \ldots \times K_p';\\ &3)\quad \eta: K_1' \times \ldots \times K_p' \to K_1 \times \ldots \times K_p;\\ &4)\quad \nu: y_1' \times \ldots \times y_c' \to y_1 \times \ldots \times y_c, \end{aligned} \tag{1}$$

таких, что следующая диаграмма будет коммутативна:

$$\begin{array}{ccccc} x_1 \times \ldots \times x_d \times K_1 \times \ldots \times K_p & & \xrightarrow{\Omega} & & K_1 \times \ldots \times K_p \times y_1 \times \ldots \times y_c; \\ \downarrow h \qquad\quad \downarrow \xi & & & \uparrow \eta \qquad \uparrow \nu & \\ x_1' \times \ldots x_d' \times K_1' \times \ldots \times K_p' & & \xrightarrow{\Omega'} & & K_1' \times \ldots \times K_p' \times y_1' \times \ldots \times y_c'. \end{array}$$

Тогда в соответствии с (1) задачу разработки обеспечивающего комплекса ИАС можно интерпретировать как поиск совокупности компонент (объектов, модулей, классов) $\{Kp'\}$, способов их соединений $\{x'd\}$, $\{y'c\}$ и взаимодействия $\Omega'$ по известному алгоритму, задаваемому перечнем операторов $\{Kp\}$ , способу их взаимодействия $\{\Omega\}$ и соединений друг с другом $\{xd\}$ и $\{yc\}$, обеспечивающих выполнение заданных целевых функций разработки.

К инструментальным средствам разработки обеспечивающего комплекса ИАС может быть отнесена система алгоритмических алгебр (САА) [2], которая по сравнению с другими средствами учитывает требования постановки задачи разработки обеспечивающего комплекса ИАС. САА состоит из пары алгебр: алгебры операторов $H_o$ и алгебры условий $H_u$. Одна из разновидностей САА называется РСА (регулярной схемой алгоритма) и представляет собой систему:

$$H = (z_1, z_2, ..., z_i, ..., z_p, e, \varnothing, \alpha_1, ..., \alpha_j, ..., \alpha_g, 1, 0),$$

где $z_i, i \in [1, p]$ – операторы РСА, причем $z_i \in H_o$; e – тождественный, а $\varnothing$ – пустой (неопределенный) оператор, т.е. единичный и нулевой элементы алгебры $H_o$; $\alpha_1, ..., \alpha_j, ..., \alpha_g$ – условия РСА, $\alpha_j \in H_u$; 1,0 – единичный и нулевой элементы алгебры $H_u$.

На алгебре $H_o$ определены следующие основные операции:

1) $(z_i \bullet z_j)$ – композиция (“умножение”) операторов, заключающаяся в последовательном их применении;

2) $(z_i \vee z_j)$ – дизъюнкция операторов, что эквивалентно алголоподобной конструкции if $\alpha$ then $z_i$ else $z_j$^:

$$\varphi_i : x_1 \times ... x_d \times K_1 \times K_p \to K_i;$$
$$1 \le i \le p;$$
$$\phi_j : x_1 \times ... x_d \times K_1 \times K_p \to y_j;$$
$$1 \le j \le c \quad .$$

3) $\{z_i\} - \alpha$ - итерация, эквивалентная конструкции while $\bar{e}$ do $z_i$;

4) недетерминированная дизъюнкция операторов [3].

На алгебре $H_u$ определена сигнатура булевых операций на наборе 0,1 [3].

Для удовлетворения условию функциональной полноты РСА необходимо дополнение $H_u$ тождественно неопределенным условием $\varphi$. В [2] доказано, что недетерминированная РСА достаточна для структурных программ без использования дополнительных средств.

В конкретной РСА связи между $Z_i \in H_o$ и условием $\alpha \in H_u$ задаются на наборе 0,1, т.е. существует возможность указания наличия (отсутствия) условия активации оператора. Источник условий активации того или иного оператора не рассматривается и его структура и структура активирующих связей считаются определенными.

Для решения задачи синтеза необходимо указание вида связи между операторами РСА, раскрытия структуры управляющего устройства и указание связи его с операторами. Это откроет возможность для формального определения топологии связей между объектами, классами проектируемых программных средств информационной системы.

Разнообразные связи в информационной системе (например, связи клиент-сервер, линии управления, шины обмена данными, адресами и т.п.) можно интерпретировать через множество отношений.

Пусть $r_k, k \in [1, p]$ – отношение между операторами РСА, причем $r_k \in H_r$; $r_e, r_\varnothing$ – соответственно единичный и нулевой элементы алгебры $H_r$.

**Определение.** Систему

$$H = (Z_i, \ldots, Z_p, e, \varnothing, \alpha_i, \ldots, \alpha_g, 1, 0, \varphi, r_i, \ldots, r_k, r_e, r_\varnothing)$$

назовем функционально-структурной схемой алгоритма (ФССА).

Предлагаемая языковая конструкция включает в себя тройку алгебр ($H_o$, $H_u$, $H_r$).

Каждый элемент ФССА будем записывать в виде композиции

$$S_i = (_{\alpha_u r_{u_i}} (\alpha_\vartheta r_{\vartheta_i} Z_i r_{\omega_i} \alpha_\omega)_{r_{f_i} \alpha_f} \vee \varnothing),$$

где $\alpha_\vartheta r_\vartheta, \alpha_\omega r_\omega$ – характеризуют интерфейс входных каналов и согласование передачи информации преобразователю $Z_i$; $r_{\omega_i} \alpha_\omega, r_{f_i} \alpha_f$ – характеризуют интерфейс выходных каналов передачи информации от преобразователя $Z_i$.

Кроме определения в [2], введены определения элементарных входных и выходных цепочек отношений (ЭЦО), композиции ЭЦО (ЭЦО будет активированной, если условия в соответствующих ЭЦО равны и принимают единичное значение). Также введено понятие элементарной структурной цепочки (ЭСЦ), которая имеет вид:

$$S_i = _{\alpha_u r_{u_i}} (\alpha_\vartheta r_{\vartheta_i} Z_i r_{\omega_i} \alpha_\omega)_{r_{f_i} \alpha_f},$$

где $\alpha_u r_u, r_{f_i} \alpha_f$ – интерфейс ЭСЦ, в первую очередь – временное согласование ЭСЦ в алгоритмической структуре ИАС.

Среда разработки ИАС должна иметь единую базу знаний, которая может адаптироваться в соответствии с условиями эксплуатации обеспечивающего комплекса ИАС. Адаптивные, объектно-ориентированные модели, разработанные и отлаженные на первой фазе жизненного цикла разработки обеспечивающего комплекса ИАС, должны использоваться на всех последующих его фазах, облегчая разработку всех составных подсистем обеспечивающего комплекса ИАС, его отладку и тестирование, сопровождение и дальнейшую модификацию.

Адаптивный к условиям использования метод разработки структуры ИАС имеет следующие шаги:

1. С учетом условий функционирования ИАС в условиях ЧПС и на основании предложенного языка ФССА разрабатывается вариант функциональной модели обеспечивающего комплекса ИАС.
2. На том же языке описываются все комплексы обеспечивающего комплекса ИАС.
3. Задаются критерии риска для полученного структурного решения ИАС.
4. На этапе технического предложения решается задача оптимального «покрытия» из базы данных ЭСЦ-ми алгоритмического комплекса, описанного на модифицированном языке функциональных преобразований регулярных схем алгоритмов элементами всех комплексов, которая обеспечивает ИАС. Оптимизация выполняется по критериям риска (шаг 3) и с использованием генетического алгоритма для оптимизации времени реализации задач функционального комплекса в обеспечивающем комплексе ИАС. Время определяется на основании подсчета суммы временных характеристик интерфейса ЭСЦ - $\alpha_u r_u, r_{f_i} \alpha_f$.
5. На этапе конструирования обеспечивающего комплекса ИАС в интеллектуальной среде разработчика принимаются компромиссные решения о распределении подсистем по процессорам и устройствам и устанавливаются основные концепции, которые формируют основу детальной последующей разработки обеспечивающего комплекса ИАС.

**Выводы**

На основании проведенных исследований предложен адаптивный метод разработки обеспечивающего комплекса ИАС, который отличается от существующих более качественной оценкой ситуаций в заданный срок и с минимальными экономическими и социальными рисками в ЧПС, что позволяет получить адаптивную к условиям ЧПС структуру обеспечивающего комплекса ИАС и своевременный прогноз развития ЧПС с минимальным экономическим и социальным риском.

**Список литературы: 1.** *Kuzemin A*. Situation centres in modern sate // International Journal on Information Theories&Applicatios. Bulgaria, 2004. Vol. 11, №1. P. 79-82 **2.***Куземин А.Я., Иванов В.И., Величко А.И.* Об одном подходе к синтезу средств обработки дискретной информации информационно-измерительных систем на микропроцессорах и БИС //АСУ и приборы автоматики. Харьков: ХНУРЭ, 1983. Вып. №68.С.85-91. **3.** *Куземин А.Я.* Объектно-ориентированная технология проектирования программных средств информационной системы // Искусственный интеллект. Донецк : ИИИ, 1999. С. 81-93.

*Поступила в редколлегию 22.08.2007*

**Куземин Александр Яковлевич,** канд. техн. наук, проф. кафедры информатики, начальник инновационно-маркетингового отдела ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел.: 8 (057) 702-15-15, e-mail: kuzy@kture.kharkov.ua.

**Левыкин Виктор Макарович**, д-р техн. наук, проф., зав. кафедрой ИУС ХНУРЭ. Научные интересы: разработка информационно-управляющих систем. Адрес: Украина, 61166 Харьков, пр. Ленина, 14, тел.: 8 (057) 702-15-15, e-mail: ius@kture.kharkov.ua.

УДК 004.5; 004.7; 004.8

*Н.В. ГОЛОВИЙ (ГУСАРЬ), АСЕР ДАЮБ*

# РАЗРАБОТКА МОДЕЛИ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ОБЛАСТИ СЕРВИСНОГО ОБСЛУЖИВАНИЯ АВТОМАТОВ ФИНАНСОВОГО САМООБСЛУЖИВАНИЯ

Предлагается оригинальный метод, базирующийся на исследовании процесса как совокупности ситуаций, представленных в гранулированном виде, разрешает учесть много фактов, их прямых и обратных связей, которые не по силам "ручной" технологии поддержки решений, а также динамично оценить альтернативы принимаемого решения.

## 1. Введение

В последнее время в связи с плотным насыщением рынка банковских услуг автоматами финансового самообслуживания сфера управления, исследования и диагностики таких устройств привлекает к себе все больше внимания. На первый план выходит процесс борьбы между сервисными организациями за доли рынка обслуживания банкоматов, в котором уровень обслуживания и время реакции на различные кризисные ситуации играют решающую роль.

Вывод – актуальность исследования данной области очевидна, а наличие интеллектуальной системы поддержки принятия решений при управлении, эксплуатации и сопровождении автоматов финансового самообслуживания является неоспоримым достоинством.

## 2. Постановка задачи

Необходимо построить модель системы поддержки принятия решений в области сервисного обслуживания банкоматов, которая должна обладать широкими функциональными возможностями, предоставлять значение определенных показателей устройств и решать задачи по планированию, прогнозированию и оптимизации деятельности сервисной организации.

Система должна познавать свое окружение и адаптироваться к нему или изменять его за счет накопленных в процессе функционирования знаний и приобретенных навыков.

## 3. Основные понятия

В основе построения модели системы лежит понятие *ситуации,* как совокупности событий, условий и обстоятельств, в которых протекает исследуемый процесс.

Разнообразие ситуаций создает всю полноту функционирования системы. «…*Ситуация* есть принуждение к принятию решения, свобода же состоит в выборе решения » (N.Hartmann, 1941).

Введем понятие видов ситуаций в деятельности сервисной организации: перспективные (открывающие новые возможности развития)/деструктивные (блокирующие развитие), управляемые/неуправляемые, объективные/субъективные.

Каждая *ситуация* имеет специфическую структуру, т.е. набор устойчивых компонентов, характеризующих исследуемый процесс. Обозначая границы, функции и направленность процесса в определенный промежуток времени, *ситуация* выступает в качестве модели анализа и одновременно служит методом проектирования, позволяя описать некоторую совокупность условий и обстоятельств, характеризующих функционирование системы, а также определить решение проблем путем создания более оптимальных условий.

Характеристика ситуации как модели осуществляется путем выделения и анализа тех ее компонентов, которые: являются относительно устойчивыми и существенными, определяют ее границы; могут быть изменены или усовершенствованы.

**4. Построение модели**

С учетом описанного выше, мы можем представить нашу систему деятельности сервисной организации как совокупность ситуаций $C_t$, характеризующих состояние системы в определенный промежуток времени:

$$Sis = \{C_t\},\ t=1,\ldots,T, \quad (1)$$

T – период времени процесса с фиксированными моментами t, в которые реализуются $C_t$, составляющие модель системы.

Каждая ситуация $C_t$ является ассоциативным отображением множества микроситуаций, описывающих состояние компонентов системы, на ее исход:

$$C = \{X, Y, S, U, F\}; \quad (2)$$

Компоненты, входящие в набор, имеют следующие назначения: X – множества входных параметров или микроситуаций (векторные); Y – множества выходных параметров или исходов (векторные); S, U – множества структур и структурных единиц, их составляющих; F – множество базисных функций, реализуемых в узловых элементах.

Обработка информации в рассматриваемой системе основана на методах дробления (грануляции) и ассоциативно-логической обработки отношений, процессов и данных [1].

Необходимость дробления связана с неопределенностью информации в реальных условиях. Предполагается, что соответствующий каждой микроситуации составной информационный объект (переменная, отображение, образ) может быть декомпозирован на некоторые элементы - гранулы. Каждая гранула является набором элементарных частиц, которые связаны вместе неопределенностью, близостью, подобностью и функциональностью.

Формально объект $O_c$ может быть представлен совокупностью гранул, т.е.:

$$O_c=\text{incl\_g}(G_1,\ldots,G_i,\ldots,G_N)\ ; \quad (3)$$

Incl_g – отношение объединения.

Каждая из гранул обладает определенным набором атрибутов с их значениями:

$$G_i=\text{has\_a}(A_1,\ldots,A_j,\ldots,A_M)\ ; \quad (4)$$

$$A_j=\text{has\_v}(V_1,\ldots,V_q,\ldots,V_Q)\ , \quad (5)$$

$A_j$ – j-й атрибут гранулы $G_i$ ; $V_q$ - q-е значение атрибута $A_j$ ; has_a и *has_v* – отношение принадлежности атрибута объекту и значения атрибута соответственно.

В результате мы получили модель системы, представленную в виде совокупности ситуаций, состоящих на самом нижнем уровне из гранул с определенным набором атрибутов. Такая модель станет основой разрабатываемой системы поддержки принятия решений при эксплуатации автоматов финансового самообслуживания и позволит описать предметную область через совокупность объектов с их свойствами.

**5. Интерпретация модели**

Следующим шагом является выявление связей между составляющими ситуации гранулами.

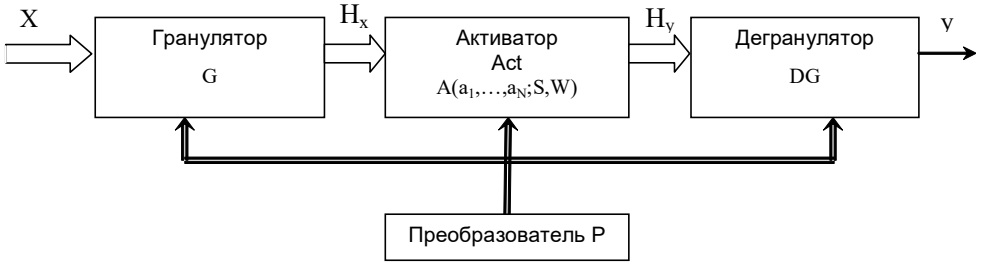
Введем понятие юнита *(Unit)* как минимального обучаемого элемента, способного самостоятельно обрабатывать информацию в гранулированном виде.

Формализованная информационная модель юнита может быть представлена набором множеств

$$M_c = \{X, W_c, H_x, S_c, H_y, BF, y\}, \quad (6)$$

где $X = \{x_0, ..., x_n\}$ – множество входных параметров – микроситуаций; $W_c = \{w_0, ..., w_v\}$ – множество регулируемых весов ($m \geq n$); $H_x = \{h_{x1}, ..., h_{xq}\}$- множество скрытых входных параметров, соответствующих информационным гранулам на входах; $H_y = \{h_{y1}, ..., h_{ye}\}$- множество скрытых выходных параметров, соответствующих информационным гранулам на выходах; $S_c = (s_{c1}, ..., s_{cr})$ – множество связей скрытых входных и выходных информационных гранул, определяющих цепочку преобразований гранул при активизации этой связи; $BF = (bf_1, ..., bf_k)$- множество базисных функций; y – выходной параметр юнита.

Такая информационная модель поддерживается структурно-функциональной моделью юнита, представленной на рисунке [2].




Структурная модель юнита

Здесь G – информационный гранулятор, формирующий множество $H_x$; Act – активатор, состоящий из множества активаторных элементов $a_i$, i=1, ..., N, выполняющих преобразование скрытых информационных параметров и формирование множества $H_y$ в соответствии со связями $S_c$ и весами $W_c$; DG – информационный дегранулятор, формирующий выходной параметр y; P – преобразователь, формирующий множество связей $S_c$ и весов $W_c$ при настройке юнита на отображение $X \rightarrow y$, аппроксимирующее функцию $y = F(X)$.

Юнит является универсальным преобразователем информации (адаптивным аппроксиматором), имеющим n-входов и один выход. Он соответствует в модельном плане биологическому нейрону с его сетью синапсов, через которые организуются связи с другими нейронами [3].

Юниты объединяются в кластер (*Cluster*). Информационная модель кластера может быть формально представлена набором множеств (6), где $X = (X_1 \vee X_2 \vee \ldots \vee X_M)$ - представляет собой "склеенное" множество входов юнитов, входящих в кластер; $W_N$, $H_x$, $H_y$ – множества весов и скрытых параметров кластера, $S_N$ – множество связей кластера, объединяющее связи юнитов, BF – набор базисных функций активаторных элементов, одинаковый для всех юнитов; скалярный выход y заменен на векторный Y – объединенное множество выходных параметров кластера.

В какой-то мере кластер с частичным объединением юнитов можно считать однослойной сетью клеток с входами, параллельно подведенными к каждой клетке (от каждого входа к каждой клетке). Такое представление кластера применяется в нейроинформатике для конструирования слоистых и модульных нейросетей. В данной работе оно использовано для моделирования кластеров, реализующих поведенческие функции (отношения) с несколькими связанными параметрами, зависящими от ряда общих аргументов [1].

Юниты и кластеры являются базовыми модулями, из которых предполагается строить систему поддержки принятия решений (СППР).

**6. Этапы моделирования**

Рассмотрим этапы предлагаемой методики на примере исследуемой предметной области.

**Этап 1** – определение цели и ее подцелей. *Цель* – повышение уровня сервисного обслуживания банкоматов. Подцели: уменьшение времени реакции на возникшую проблему; уменьшение времени, потраченного на ремонт устройства; оптимизация маршрута перемещения сервисного инженера; оптимальное размещение запасных частей на региональных складах; максимально быстрая доставка запасных частей с основного склада; прогнозирование количества необходимых запчастей на будущий период; прогнозирование затрат на командировки сервисным инженерам; построение оптимальных маршрутов доставки запчастей и передвижения инженеров.

**Этап 2** – формирование ситуации, описывающей состояние предметной области. Согласно (2) такую ситуацию можно описать совокупностью микроситуаций типа: **микроситуация 1** – описывает состояние парка устройств; **микроситуация 2** – описывает финансовые затраты предприятия в результате сервисного обслуживания (командировки, премии и т.д.); **микроситуация 3** – описывает характеристики сервисного обслуживания парка банкоматов; **микроситуация 4** – описывает деятельность департамента логистики.

**Этап 3** – гранулирование микроситуаций. Учитывая то, что каждая микроситуация ассоциируется с объектом в некотором состоянии, проведем такие исследования.

Рассмотрим микроситуацию 1. Объектом данной микростуации является парк банкоматов $O_{I,}$ находящийся в состоянии $S_I$. Опишем состояние данного объекта, используя его гранулирование. Таким образом, согласно (3) получаем представление объекта через

совокупность гранул. В нашем случае такой гранулой является единица устройства, например, банкомат. После этого необходимо определить атрибуты каждой гранулы и их возможные значения.

Итак, рассмотрим гранулу $G_{1i}$ –автомат финансового самообслуживания. Выделим атрибуты, описывающие его состояние (4): $\{A_{ik}\}$ – параметры установки банкомата; $\{A_{im}\}$ – параметры эксплуатации банкоматов; $\{A_{in}\}$ – параметры, описывающие состояние узлов и блоков; $\{A_{ip}\}$ – другие параметры.

Аналогичным образом, рассматривая микроситуацию 3, выделим объект $O_{3=}O_1$, находящийся, для этой микроситуации, в состоянии $S_3$ и гранулированный на составляющие $\{G_{3i}\}$, каждая из которых обладает следующими атрибутами: $A_{i1}$ – количество зарегистрированных заявок; $A_{i2}$ – количество просроченных заявок; $A_{i3}$ – количество невыполненных заявок; $A_{i4}$ – среднее время в ремонте; $A_{i5}$ – количество использованных запчастей; $A_{i6}$ – среднее время выполнения заявки.

В результате мы получим вектор входных параметров *X* для рассматриваемой ситуации, как совокупность определенных выше атрибутов, имеющий структуру *S.*

**Этап 4 –** определение комплексной целевой программы (КЦП). На данном этапе производится формирование совокупности мероприятий, называемых в дальнейшем «проектами», объединенных единством главной цели и общими ресурсами.

Сформируем КЦП для исследуемой предметной области: автоматизация процесса регистрации заявок на сервисное обслуживание; оптимизация распределения материальных и человеческих ресурсов; повышение квалификации сотрудников сервисной службы; оптимизация схемы логистики; внедрение средств моментального оповещения; введение дополнительных процедур по профилактике устройств; создание нестандартных решений по защите устройств от несанкционированного вмешательства.

Применим для полученной целевой программы процедуру гранулирования, т.е. представим каждый исход как микроситуацию, состоящую из множества объектов, которые обладают определенным набором атрибутов. В результате, аналогично процедуре с входными параметрами, мы получим вектор выходных параметров *Y*, имеющий структуру *U.*

Таким образом, мы получили отображение множества микроситуаций, описывающих входную информацию на множество микроситуаций, описывающих исходы, т.е.выражение (2).

**Этап 5 –**построение зависимости между входными и выходными параметрами ситуации. На данном этапе необходимо найти множество базисных функций, определяющих отображение множества входных параметров в исходы ситуации. Для построения такой зависимости будем использовать понятие юнитов и кластеров, описанное в п.5 настоящей статьи. В результате получим коэффициенты влияния одних параметров на другие и их структурную зависимость.

**Этап 6 –** вычисление показателей эффективности принятых проектов.

## 7. Заключение

Новизна данного подхода заключается в использовании оригинального метода, который основывается на рассмотрении исследуемого процесса как совокупности ситуаций, представленных в гранулированном виде, что позволяет учесть сотни факторов и их прямых и обратных связей, что не под силу при “ручной” технологии поддержки решений, а также динамически оценить альтернативы принимаемого решения.

СППР, построенные таким образом, являются инструментом для оказания помощи лицу, принимающему решение, в решении таких задач: количественный анализ влияния внешних факторов на выбранную главную или промежуточную цель; определение перспективных направлений выполнения комплексной целевой программы; определение показателей относительной эффективности вариантов решений относительно конкретных действий (проектов), направленных на выполнение.

Таким образом, в результате проведенных исследований были разработаны теоретические и прикладные основы построения информационных технологий для автоматизации функциональных задач управления в области сервисного обслуживания банкоматов и предложена модель такой системы, которая позволит увеличить эффективность работы сервисной службы.

Предлагаемая модель используется для решения не одной, а целой совокупности проблем, возникающих при стратегическом планировании в предметной области и повседневной деятельности по управлению объектами предметной области.

Реализация данной системы предполагается с использованием механизма нейронных сетей. Однако для этого еще предстоит решить несколько задач, а именно, выбор и описание существенных параметров предметной области, выявление взаимосвязи между параметрами, оптимизация, выбор структуры и характеристик нейронной сети для реализации системы.

Несмотря на это, следует отметить, что в дальнейшем могут быть созданы интеллектуальные системы с нервно-системной организацией структуры, функций и поведения, в основе которых будут лежать рассмотренные принципы.

**Список литературы: 1.** *Станкевич Л.А.* Когнитивные нейрологические системы управления. Проблемы нейрокибернетики. Материалы XII Международной конференции по нейрокибернетике. Ростов-на-Дону, 1999. **2.***Ларичев О.И., Петровский А.В.* Системы поддержки принятия решений. Современное состояние и перспективы их развития //Итоги науки и техники. Сер.Техническая кибернетика. Т.21. М. ВИНИТИ, 1987. **3.** *Мушик, Мюллер.* Методы принятия технических решений. М.: Мир, 1992.

*Поступила в редколлегию 28.08.2007*

**Головий (Гусарь) Наталья Владимировна,** аспирантка кафедры Информатики ХНУРЭ. Научные интересы: системный анализ данных. Адрес: Украина, 61166, Харьков, пр.Ленина,14, тел. (057) 702 15 15, e-mail: rica1982@mail.ru.

**Ясер Даюб,** аспирант кафедры Информатики ХНУРЭ. Научные интересы: системный анализ данных. Адрес: Украина, 61166, Харьков, пр.Ленина,14, тел. (057) 702 15 15, e-mail: kuzy@kture.kharkov.ua.

.

---

УДК 004.896, 004.932

*И.В. ГАРЯЧЕВСКАЯ*

# ТЕХНОЛОГИЯ РАЗРАБОТКИ ПО СТЗ, ТЕСТИРОВАНИЯ И АДАПТАЦИИ К УСЛОВИЯМ ЭКСПЛУАТАЦИИ

При создании систем технического зрения (СТЗ) одной из основных задач является выбор методов обработки изображений и их порядка. При этом большое значение имеет стыковка аппаратного (вычислительная техника) и программного (алгоритм обработки изображений) обеспечения. Технология разработки, тестирования и адаптации ПО СТЗ к условиям эксплуатации позволит решить проблему стыковки без этапа натурного испытания.

## Введение

СТЗ с точки зрения ПО – это алгоритм обработки изображения некоторой последовательностью методов. На рис.1 приведены возможные этапы обработки изображений для задачи мониторинга пассажиропотока.

Для каждой конкретной задачи набор этих этапов индивидуален, определение этапов является сложной задачей. Кроме этого, существует еще одна проблема, а именно выбор одного метода, наиболее подходящего для каждого из этапов.

Сроки, за которые производится разработка СТЗ, исчисляются месяцами, требуют больших материальных затрат, и обязательных натурных испытаний. Результаты натурных испытаний могут выявить недостатки разработанного ПО и


Рис. 1. Этапы обработки изображений

потребовать улучшений, а это дополнительные временные и материальные затраты.

Кроме этого, даже незначительные изменения условий эксплуатации могут привести к ошибкам работы СТЗ. Сократить время, требуемое на создание ПО СТЗ, и стоимость разработки может предложенная технология.

**Анализ основных достижений и публикаций**

В настоящее время машинное моделирование – единственный способ анализа и синтеза сложных систем, к которым можно, в частности, отнести и СТЗ. Моделирование используется при проектировании, создании, внедрении, эксплуатации систем, а также на различных уровнях их изучения – начиная от анализа работы элементов и кончая исследованием систем в целом в их взаимодействии с окружающей средой. Ранее уже проводились исследования в области создания технологии разработки СТЗ [1, 2].

*Целью* проводимого исследования является создание технологии разработки ПО СТЗ с этапом адаптации к условиям эксплуатации.

**Методы решения и полученные результаты**

**Первый** шаг технологии – определение структуры СТЗ, т.е. определение необходимых этапов и допустимых методов, реализуется с использованием продукционно-фреймовой модели представления данных.

Комбинация двух известных подходов в один обусловлена достоинствами и недостатками каждой из них.

Так, языком представления знаний целесообразно использовать фреймовую систему, так как фреймовая иерархия базы знаний позволяет знания о методах представить в структурированном виде с сохранением свойства наследования, что является актуальным. Недостатками фреймовой системы является невозможность организовать гибкий механизм логического вывода. А для представления модели предметной области необходима гибкая и простая система вывода. Модель предметной области будет взаимодействовать с методами посредством выводов продукционных правил.

Продукционно-фреймовая модель позволяет сократить объем памяти, необходимый для хранения данных.

Разработанная продукционно-фреймовая модель, проанализировав задачи и функции СТЗ, определит этапы обработки, их последовательность и множество методов для каждого из этапов.

На рис. 2. представлены два алгоритма. Оба включают в себя по одному методу для каждого этапа. Первый алгоритм состоит из метода 1 для первого этапа, метода 1 для второго этапа, метода 1 для третьего этапа. Второй алгоритм состоит из метода 1 для первого этапа, метода 2 для второго этапа, метода 1 для третьего этапа. Таким же образом строятся все остальные алгоритмы.

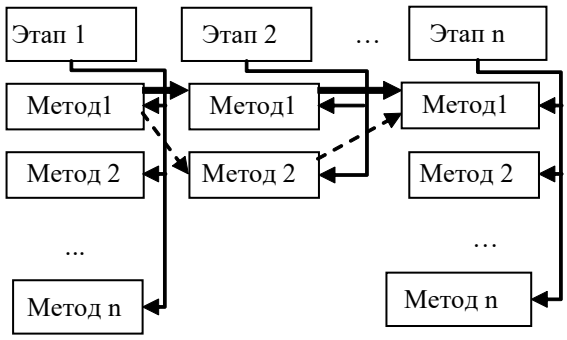



Рис.2. Общий вид формирования алгоритма

Результатом первого шага технологии будет множество алгоритмов, полученных полным перебором из выбранных методов, которые принадлежат выделенному множеству.

**Второй** шаг технологии – сокращение количества полученных алгоритмов.

Необходимо осуществить выбор из множества сгенерированных алгоритмов одного наиболее удачного.

В общем случае следует получить конкретное решение через модель многокритериального оценивания частных критериев, характеризующих альтернативы, и субъективной информации о предпочтении частных критериев:

$$Q \to P(a) \to a^0, \quad P(a) = F\left[w_i, k_i(a)\right].$$

Выбор осуществляется по следующему алгоритму:

1. Определить важность (вес) каждого критерия w. (В зависимости от задач, стоящих перед СТЗ, разработчик определяет, какие критерии более важны).

Так, для СТЗ, работающих в режиме реального времени, скорость обработки является ключевым критерием, а для СТЗ в медицине качество будет гораздо более важным критерием по сравнению со временем.

К основным критериям качества работы алгоритма обработки видеоинформации и, как следствие, всей системы технического зрения можно отнести (Таблица):

– величину отклонения между фактическим и контрольным временем обработки кадра;
– количество шума до и после фильтрации;
– расположение границ должно быть как можно ближе к правильным;

| | k1 | k2 | … | kn | Результирующий показатель |
|---|---|---|---|---|---|
| | w1 | w2 | … | wn | |
| A1 | z11 | z12 | … | z1n | A1’ |
| A2 | z21 | … | … | … | … |
| … | … | … | … | … | … |
| An | zm1 | … | … | zmn | … |

– количество пикселей, выражающих границы объекта, должно быть близким к единице;
– только один результат для одной границы;
– количество ошибок при распознавании и др.

2. Обработать тестовое изображение всеми алгоритмами.
3. Заполнить таблицу значениями по критериям.
4. Удалить те алгоритмы, в которых есть превышение значения хотя бы по одному критерию.

По каждому из критериев существует свое допустимое значение, превышение которого делает невозможным применение алгоритма, в котором использован метод выдавший значение выше допустимого. Например, если при подавлении шума было получено значение 50%, можно точно утверждать, что использовать такой метод нецелесообразно, и как следствие, будет удален весь алгоритм.

По окончанию данного шага алгоритма можно предположить, что любой из оставшихся алгоритмов может быть использован в СТЗ, однако необходимо определить наиболее хороший.

5. В оставшихся алгоритмах нормализовать значения:

$$k_i \in \left\lfloor k_{i_{max}}, k_{i_{min}} \right\rfloor, \quad i = 1,2,...,n \quad ,$$

$$k_{i_{max}} = 1, \quad k_{i_{min}} = 0, \quad 0 \le k_i \le 1 \quad .$$

Каждый из критериев имеет свою систему исчисления, по критерию 1 будут получены значения в миллисекундах, а по критерию 2 – в процентах. Следовательно, необходимо их привести к общему виду. Выбрав по каждому критерию наиболее хорошее значение, приравняем его к 1, а наиболее плохое значение приравняем к 0. Все промежуточные значения будут расположены соответственно в интервале от 1 до 0.

6. Вычислить результирующий показатель алгоритма.

Вычисление для 1 алгоритма:

$$A_1' = z_{11} \cdot w_1 + z_{12} \cdot w_2 + ... + z_{1n} \cdot w_n ,$$

$$A_j' = \sum_{i=1}^{n} z_{ji} \cdot w_i, \quad j \in \left[1, m\right] \quad ,$$

т.е. результирующий показатель – это сумма всех значений, по каждому критерию умноженная на его вес. Веса критериев были расставлены таким образом, чтобы сумма всех весов составляла 1.

7. Упорядочить алгоритмы

Алгоритм, чей результирующий показатель оказался самым большим, является наиболее хорошим для СТЗ. Однако последующие этапы тестирования и адаптации могут выявить недостатки при его работе, в связи с этим все алгоритмы следует упорядочить по убыванию результирующего показателя и при необходимости протестировать второй или третий алгоритм.

**Третий** шаг разработанной технологии отвечает за процесс тестирования.

Выбранный алгоритм дает наиболее хорошие показатели при обработке одного изображения, на котором вычислялись значения по критериям всех алгоритмов, и даже если это изображение было близко к тем, с которыми алгоритм будет работать в реальной системе, по одному изображению невозможно утверждать о стабильности работы алгоритма. В связи с этим этап тестирования включает в себя прогонку алгоритма по последовательности изображений или видеоряда.

Для примера опишем механизм оценивания по 1 критерию. При тестировании определяется разница между фактическим временем и контрольным, и эта разница сравнивается с установленным пределом: $t_i^{фп} - t_i^{кп} = \Delta t_i^{П} \le \Delta t^{П}$.

Расчет общей задержки обработки всего видеоряда: $\Delta T^{П} = \sum_{i=1}^{I_П} (\Delta t_i^{П} > \Delta t),\ i = 1, I_П$.

Среднее время задержки: $\Delta t_{ср}^{П} = \dfrac{\Delta T^{П}}{I_П}$.

Находим математическое ожидание для определения среднего времени обработки всех кадров: $\tilde{m}_x = \dfrac{1}{n}\sum_{i=1}^{n} x_i$.

Рассчитываем дисперсию: $\tilde{D}_x = \dfrac{1}{n-1}\sum_{i=1}^{n}(x_i - \tilde{m}_i)^2$.

Среднеквадратичное отклонение показывает, сколько кадров при обработке вышли за рамки допустимых значений: $\tilde{\sigma}_x = \sqrt{\tilde{D}_x}$.

По окончанию тестирования выдается отчет, в котором содержится график, представленный на рис.3.

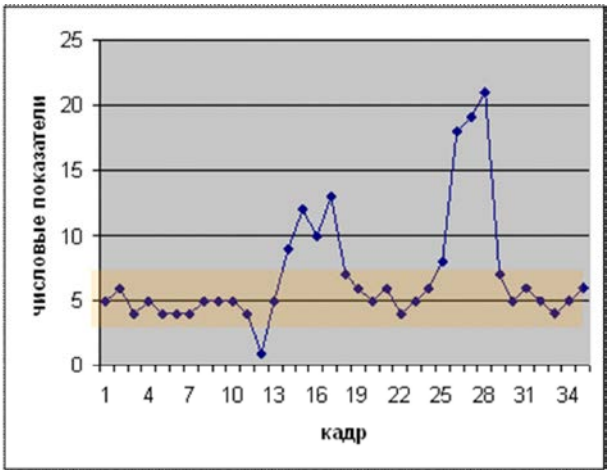



Рис.3. График показателей времени по каждому кадру

По оси *x* – номер кадра; по оси *y* – показатели по критерию (время), также отмечен допустимый предел разброса значений.

Визуальный анализ этого графика позволит определить, при обработке каких кадров алгоритм не справился и превысил допустимое время обработки. Более детальный анализ каждого плохо обработанного кадра отдельно позволит принять решение об изменении алгоритма, замене его другим, либо о пренебрежении данными ошибками.

**Четвертый,** заключительный шаг разработанной технологии позволяет произвести адаптацию ПО СТЗ к условиям эксплуатации.

Под условиями эксплуатации понимают характеристики вычислительной техники, на которой будет работать ПО.

Определение времени, которое будет занимать полная обработка изображения разработанным алгоритмом на предполагаемой, но недоступной ВТ, будет зависеть от трех параметров.

Первый – это разница времени на ВТ разработчика и ВТ реальной системы. Второй – это соотношение размера реального изображения и эталонного, на котором проводилось тестирование. Третий – количество кадров в секунду, требующих обработки.

При разработке системы было определено время работы тестового алгоритма на тестовой платформе:

$$t_d = f(p_d, a_t),$$

где $t_d$ – время работы тестового алгоритма на аппаратной платформе разработчика САПР (d – developer); $p_d$ – характеристики аппаратной платформы разработчика; $a_t$ – тестовый алгоритм (t – test).

Для определения тестового алгоритма на разных платформах $T_{ns} = f(P_{ns}, a_t)$, где $T_{ns}$ – вектор временных характеристик платформ для обучающей выборки нейронной сети; $P_{ns}$ – вектор характеристик аппаратных платформ, используемых для получения обучающей выборки.

Имея входные и выходные значения, можно приступать к обучению нейронной сети:

$$learn(NS, P_{ns}, T_{ns}),$$

где $learn$ – функция обучения нейронной сети. Ее параметры: $NS$ – нейронная сеть.

Настроили нейронную сеть. Проектировщик разрабатывает алгоритм и тестирует его на своей платформе:

$$t_{pr} = f(p_{pr}, a),$$

где $t_{pr}$ – время работы спроектированного пользователем САПР алгоритма на аппаратной платформе проектировщика; $p_{pr}$ – характеристики аппаратной платформы проектировщика; $a$ – разработанный алгоритм.

Время работы тестового алгоритма на аппаратной платформе проектировщика: $t_{prt} = f(p_{pr}, a_t)$.

Рассчитанное нейронной сетью предполагаемое время работы тестового алгоритма на аппаратной платформе проектировщика: $t'_{prt} = run(NS, p_{pr})$.

Далее определяем трудоемкость:

$$d = \frac{t_{pr}}{t'_{pr}},$$

$d$ – трудоемкость (сложность) разработанного проектировщиком алгоритма в сравнении с тестовым алгоритмом.

Предполагаемое время работы тестового алгоритма на целевой (target) аппаратной платформе, рассчитанное нейронной сетью: $t'_t = run(NS, p_t)$.

Время работы спроектированного алгоритма на целевой аппаратной платформе: $t_t = t'_t \cdot d$.

Погрешность определения времени работы тестового алгоритма, вычисляемого нейронной сетью:

$$\Delta' = \frac{(t_{prt} - t'_{prt})^2}{t_{prt}}.$$

Погрешность определения времени работы спроектированного алгоритма на целевой аппаратной платформе: $\Delta = \Delta' \cdot t_t$.

Технология реализована с помощью использования нейронной сети. Структурная схема процесса представлена на рис.4.

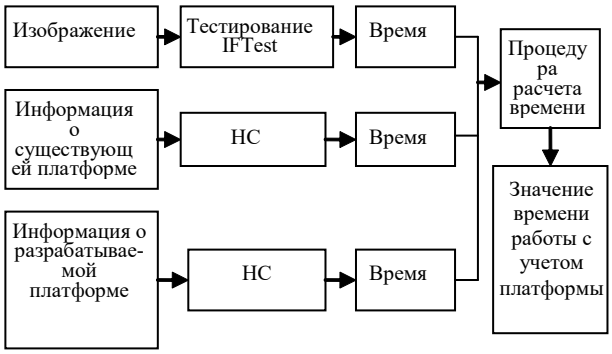



Рис.4. Схема этапа адаптации

Описанный четвертый шаг разработанной технологии позволяет без натурных испытаний определить пригодность разработанного ранее алгоритма обработки изображений в СТЗ на предполагаемой ВТ, что дает возможность увеличить скорость разработки и сократить затраты.

**Выводы по данному исследованию и перспективы дальнейших разработок в этом направлении**

Предложена технология разработки ПО СТЗ, тестирования и адаптации к условиям эксплуатации.

Описанная технология позволяет за короткое время и с минимальными затратами разработать ПО, т.е. алгоритм обработки изображений для СТЗ, провести тестирование и адаптацию его к условиям эксплуатации. Пошаговая структура технологии позволяет вносить изменения в алгоритм на любом шаге, и в случае отрицательных результатов производить изменения как структуры ПО, так и корректировку внутренних параметров методов (пороги, коэффициенты, …).

Разработанная технология базируется на:

– продукционно-фреймовой модели для определения этапов и методов обработки изображений и формирования множества алгоритмов;

– несколько измененных аксиоматических методах принятия решения для выбора наиболее хорошего алгоритма из множества сгенерированных алгоритмов;

– нейросетевых технологиях для определения времени работы алгоритма на недоступной ВТ или определения минимальных требований к ВТ.

**Список литературы: 1.** *Гарячевская И.В., Кузёмин А.Я.* Автоматизация процесса разработки и отладка алгоритмов обработки изображений для СТЗ // Искусственный интеллект. 2004. № 2. С. 269-273. **2.** *Гарячевская И.В., Кузёмин А.Я., Кравцов К.А.* Автоматизированная система для проектирования СТЗ мобильного робота // Искусственный интеллект. 2004. № 2. С. 274-278.

*Поступила в редколлегию 15.09.2007*

**Гарячевская Ирина Васильевна,** аспирантка, м.н.с. кафедры Информатики ХНУРЭ. Научные интересы: построение СТЗ, обработка изображений. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-14-38, e-mail: irisha@imagefilterer.com.ua.

УДК 001:002

*А.С. ШКИЛЬ, С.В. ЧУМАЧЕНКО, С.В. НАПРАСНИК, А.В. БАБИЧ, Е.В.ГАРКУША*

# ПРИНЦИПЫ ПОСТРОЕНИЯ СЕАНСА ТЕСТИРОВАНИЯ В КОМПЬЮТЕРНОЙ СИСТЕМЕ ТЕСТИРОВАНИЯ ЗНАНИЙ OPENTEST2

Рассматриваются вопросы формирования частично адаптированного сеанса тестирования в компьютерной системе тестирования знаний OpenTEST2. Сеанс тестирования формируется как репрезентативная выборка из структурированной базы тестовых заданий с использованием методов дискретной оптимизации. Даются рекомендации относительно формирования структуры базы данных тестовых заданий системы OpenTEST2.

## Введение

В Харьковском национальном университете радиоэлектроники (ХНУРЭ) разработана и в течение нескольких лет находится в эксплуатации компьютерная система тестирования знаний студентов OpenTEST, а с 2007 года – вторая версия этой системы OpenTEST2. В ней реализуется частично адаптированный алгоритм проведения контроля знаний со случайной выборкой заданий из тестовой базы данных на основе модели учебного материала. Адаптивность состоит в том, что тематическая структура набора тестовых заданий для каждого тестируемого, а также их качественные и количественные характеристики зависят от модели учебного материала.

Тестирование в системе OpenTEST2 осуществляется путем организации сеанса тестирования для каждого тестируемого. Сеанс тестирования представляет собой репрезентативную выборку из генеральной совокупности теста на основе его содержательной структуры, максимально покрывающей модель предметной области.

С точки зрения теории педагогических измерений сеанс тестирования – это фрагмент педагогического теста как системы заданий возрастающей трудности, специфической формы, которая позволяет качественно и эффективно измерить уровень и структуру подготовленности испытуемых. Составной единицей педагогического теста является тестовое задание, отвечающее требованиям к заданиям в тестовой форме и статистическим требованиям известной трудности, дифференцирующей способности (вариации баллов), и положительной корреляции с результатами по тесту в целом [1].

Сеанс характеризуется длиной (количеством N тестовых заданий в сеансе), временем, отводимым на сеанс, и количеством попыток прохождения теста в рамках одного сеанса. Хотя задания в сеансе нумеруются в порядке возрастания трудности, тестируемый может отвечать на них в произвольном порядке.

Формой тестовых заданий в системе OpenTEST2 являются вопросы закрытого типа (типа «выбор одного из нескольких», «выбор нескольких из нескольких», «соответствие») и открытого типа (свободный ввод короткого ответа) с произвольным назначением весомозначности тестовых заданий в баллах в рамках одного теста [2].

## Трудность и дифференцирующая способность тестовых заданий

В педагогических измерениях для количественной оценки содержания теста разными авторами используются понятия трудности, сложности и весомозначности тестовых заданий.

1. Весомозначность тестового задания обычно характеризуется баллом за правильный ответ и отражает субъективный взгляд автора теста на относительное влияние конкретного тестового задания на суммарную оценку за ответы на все предложенные тестовые задания.

2. Сложность тестового задания характеризуется количеством умственных или вычислительных операций, необходимых для получения правильного ответа. Эта мера широко используется в учебном процессе при нормировании времени, необходимого на усвоение определенного количества учебного материала.

3. Трудность тестового задания – это эмпирическая оценка отношения тестируемых к конкретному тестовому заданию путем правильных или неправильных ответов на него. В качестве меры трудности тестового задания принято использовать частоту (вероятность) получения правильного ответа на i-е тестовое задание ($p_i$) :

$$p_i = \frac{R_i}{NN_i} ,$$

где $R_i$ – количество правильных ответов на *i*-е тестовое задание; $NN_i$– количество участий i-го тестового задания в испытаниях.

Следует отметить, что $p_i$ на самом деле характеризует легкость тестовых заданий, поэтому между истинной трудностью тестового задания и величиной $p_i$ существует обратно пропорциональная зависимость, т.е. чем меньше значение $p_i$ , тем труднее тестовое задание.

При достаточно большой выборке испытуемых $p_i$ можно считать вероятностью получения правильного ответа,. $p_i$ характеризуется положительным числом в интервале от 0 до 1. В системах компьютерного тестирования знаний [3] при анализе трудности тестовых заданий принято выделять три уровня трудности:

– легкие ( $0,6 < p_i \le 0,8$ );

– средние ( $0,4 < p_i \le 0,6$ );

– трудные ( $0,2 < p_i \le 0,4$ ).

Задания, у которых $p_i < 0,2$ (очень сложные) и $p_i > 0,8$ (очень легкие), не рекомендуется использовать в качестве тестовых, если они не несут какой-либо специальной нагрузки. Необходимо иметь в виду, что трудность, как эмпирическая мера количественной оценки тестовых заданий, требует предварительной апробации на достаточно представительной типичной выборке обучаемых. И только после этого тестовые задания могут быть отнесены к определенному уровню трудности.

В качестве меры дифференцирующей способности i–го тестового задания (вариации тестовых баллов) при дихотомической оценке правильных и неправильных ответов принято использовать его дисперсию $D_i = p_i * q_i$, где $q_i = 1 - p_i$. Таким образом, наилучшей дифференцирующей способностью обладают тестовые задания среднего уровня трудности, а максимальная дисперсия будет при $p_i = 0,5$.

**Модель тестовой базы данных**

В компьютерной системе тестирования знаний OpenTEST2 принята четырехуровневая иерархическая модель предметной области и соответствующая иерархическая структуризация тестовых заданий (вопросов):

«КАТЕГОРИЯ ТЕСТА» -> « ИМЯ ТЕСТА» -> «ИМЯ ТЕМЫ»-> «ВОПРОСЫ ТЕМЫ»-> «КАТЕГОРИЯ ТЕСТА»

– это организационный уровень иерархии, который формируется для каждой конкретной инсталляции OpenTEST2 администратором системы. Система OpenTEST2 адаптирована для использования в учебных учреждениях со сложной структурой, поэтому организационное понятие «категория» связано с организацией учебного процесса в вузе. Под «категорией» обычно понимают кафедру или факультет (деканат). Такая категория позволяет пользователям значительно сократить время поиска тестов в базе данных системы OpenTEST2.

“ИМЯ ТЕСТА” обычно соответствует названию учебной дисциплины или ее части. Доступ к тесту (при организации сеанса тестирования) осуществляется по категории теста и имени теста.

“ИМЯ ТЕМЫ” обычно соответствует структурной единице учебной дисциплины, определенной в ее рабочей программе, и используется при формировании сеанса тестирования из тестовой базы данных.

“ВОПРОСЫ ТЕМЫ” – это тестовые задания (вопросы) разных форм, содержащие текстовые, графические и мультимедийные компоненты.

Три нижних уровня иерархии модели предметной области образуют содержательную структуру теста.

В соответствии с моделью предметной области и принятой системой градаций трудностей тестовых заданий, в системе OpenTEST2 формируется тестовая база данных, каждый тест в которой состоит из L тем, а каждая тема содержит K подразделов (по количеству уровней трудности тестовых заданий). Каждый подраздел содержит тестовые задания (вопросы), трудность которых находится в пределах, заданных для этого уровня трудности. На рис. 1 показана структура базы данных для теста, содержащего 10 тем по три уровня трудности в каждой теме. На данном рисунке $t_{i,j}$ обозначает количество тестовых заданий соответствующего уровня трудности в каждой теме. Исходя из рекомендаций по общей длине теста целесообразно, чтобы соблюдалось условие $t_{i,j} > 10$.

| Тестовые задания | ТЕМЫ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Уровни трудности | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 (легкие) | $t_{1,1}$ | $t_{1,2}$ | $t_{1,3}$ | $t_{1,4}$ | $t_{1,5}$ | $t_{1,6}$ | $t_{1,7}$ | $t_{1,8}$ | $t_{1,9}$ | $t_{1,10}$ |
| 2 (средние) | $t_{2,1}$ | $t_{2,2}$ | $t_{2,3}$ | $t_{2,4}$ | $t_{2,5}$ | $t_{2,6}$ | $t_{2,7}$ | $t_{2,8}$ | $t_{2,9}$ | $t_{2,10}$ |
| 3 (трудные) | $t_{3,1}$ | $t_{3,2}$ | $t_{3,3}$ | $t_{3,4}$ | $t_{3,5}$ | $t_{3,6}$ | $t_{3,7}$ | $t_{3,8}$ | $t_{3,9}$ | $t_{3,10}$ |

Рис. 1. Структура теста в базе данных в системе OpenTEST2

**Покрытие учебного материала тестовыми заданиями**

Для численной оценки степени покрытия учебного материала тестовыми заданиями определим $C_{i,j}$ – коэффициент покрытия учебного материала j-й темы содержательной структуры теста, где $i = \overline{1,K}$, $j = \overline{1,L}$. Он является чисто качественной характеристикой и отражает субъективную точку зрения автора теста на степень покрытия содержательной структуры j-й темы теста заданиями разных уровней трудности. Коэффициент $C_{i,j}$ задается автором теста для каждого уровня трудности темы или для всего теста целиком. В дальнейшем для простоты изложения будем считать $C_{i,j}$ равными для всех j тем теста, таким образом: $C = (c_1, c_2, \ldots c_K)$.

Определим, что предметная область покрывается полностью (на 100%), если в сеанс тестирования из каждой темы попадает хотя бы по одному заданию каждого уровня трудности. На практике это часто недостижимо, потому что длина сеанса является ограниченной и условие $N \geq K * L$ трудновыполнимо. Но, с другой стороны, правильный ответ на задание 3-го уровня трудности предполагает наличие у студентов какого-то минимума знаний учебного материала, покрываемого заданиями 1-го и 2-го уровней трудности. Например, автор теста может принять, что задания первого уровня покрывают 30% учебного материала ($c_1$=0,3) данной темы или теста в целом, второго – 40% ($c_2$=0,4), третьего – 50% ($c_3$=0,5) Любые два уровня в сумме не покрывают 100% учебного материала, а все три уровня в сумме покрывают 100%, но это не арифметическая сумма, а учет взаимного «перекрытия» учебного материала различных уровней трудности.

Определим пересечение («перекрытие») уровней трудности $E$ сумму частных коэффициентов пересечения $e_i$ для каждого уровня трудности:

$$E = (e_1, e_2, \ldots e_K)\ ,\ E = \sum_{i=1}^{K} e_i\ ,\ E = \sum_{i=1}^{K} c_i - 1\ ,$$

где K – количество уровней трудности тестовых заданий.

Значения отдельных $e_i$ назначаются экспертом по аналогии $c_i$. Значение $E < 0$ говорит о неправильном выборе автором (экспертом) теста значений $c_i$. Если $E = 0$, то «перекрытие» между заданиями разного уровня трудности отсутствует, а это не совсем точно отражает содержательную структуру учебного материала (от простого к сложному).

Дополнительно определим групповые коэффициенты пересечения нескольких уровней трудности учебного материала для одной темы:

$$c_{i,j} = \sum_{i=1}^{i,j} c_i - \sum_{i=1}^{i,j} e_i\ .$$

Например, для $c_1$=0,3, $c_2$=0,4, $c_3$=0,5 E = (0,3 + 0,4 + 0,5) – 1 = 0,2 .
При этом можно определить $e_1$= 0,06, $e_2$= 0,07, $e_3$= 0,07 .
Следовательно, групповые коэффициенты покрытия вычисляются:
$c_{1,2}$=(0,3 +0,4) – (0,06 + 0,07) = 0,57 , $c_{1,3}$=(0,3 +0,5) – (0.06 + 0.07) = 0.67,
$c_{2,3}$=(0,4 +0,5) – (0,07 + 0,07) = 0,76 , $c_{1,2,3}$= 1 (по определению).

**Алгоритм формирования сеанса тестирования**

Исходя из сказанного выше, в сеанс тестирования из тестовой базы данных должны включаться задания разных уровней трудности с максимальной дифференцирующей способностью и максимальным покрытием всех разделов учебного материала.

Формальная постановка задачи: построить выборку, содержащую N тестовых заданий с максимальными значениями дисперсии $D_i$ и коэффициентов покрытия учебного материала $C_j$, где $i = \overline{1,N}$, $j = \overline{1,L}$.

Ограничения: количество тем в тесте L, количество уровней сложности в каждой теме K, из каждого уровня сложности любой темы в сеанс тестирования выбирается одно тестовое задание, при условии $K * L \le N$ .

Система OpenTEST2 построена таким образом, что все параметры сеанса тестирования и тестовой базы данных, за исключением количества уровней трудности K, задаются пользователем произвольно. Поэтому за базовый параметр при формировании сеанса тестирования выберем количество уровней трудности тестовых заданий.

Для решения поставленной задачи будем использовать методы дискретной оптимизации. Определим множество вариантов сеансов тестирования как K-мерное пространство $M(m_1, m_2, ..., m_K)$ с ограничениями $m_1 + m_2 + ... + m_K \le N$ и $0 \le m_i \le L$, где $m_i$ – количество тестовых заданий соответствующего уровня трудности [4].

Определим количество элементов пространства M первоначально без учета верхнего ограничения на $m_i$, т.е. только с учетом условия $0 \le m_i$. Из практики решения комбинаторных задач [5] известно, что количество элементов такого пространства будет равно числу целых неотрицательных решений уравнения $x_1 + x_2 + ... + x_K = N$.

Если имеются целые неотрицательные числа $x_1, x_2, ..., x_K$ такие, что $x_1 + x_2 + ... + x_K = N$, то из них можно составить сочетания из $K$ элементов по $N$, взяв $x_1$ элементов первого типа, $x_2$ элементов второго типа, …, $x_K$ – K-го типа. Наоборот, имея сочетание из K элементов по N, получим решение уравнения $x_1 + x_2 + ... + x_K = N$ в целых неотрицательных числах, где $x_1$ элементов первого типа, $x_2$ элементов второго типа, …, $x_K$ - $K$-го типа. Следовательно, между множеством всех сочетаний из $K$ элементов по $N$ с повторениями и множеством всех целых неотрицательных решений уравнения $x_1 + x_2 + ... + x_K = N$ устанавливается взаимно-однозначное соответствие. Поэтому число решений равно $f_K^N = C_{N+K-1}^N$ или $C_{N+K-1}^{K-1}$ (по свойству симметрии биномиальных коэффициентов $C_n^k = C_n^{n-k}$).

Рассмотрим проекцию трехмерного пространства $M(m_1, m_2, m_3)$ на плоскость $(m_1, m_2)$, при этом учитывая, что $m_3 = N - (m_1 + m_2)$. Если на данную проекцию наложить ограничение $m_i \le L$ , то получим число элементов пространства M, равное :

$$Z = C_{N+K-1}^{K-1} - K \cdot C_{N-L+K-2}^{K-1} + K \cdot C_{N-2(L+1)+K-1}^{K-1} . \qquad (1)$$

Вывод этой формулы достаточно сложен, а на алгоритм формирования сеанса практически не влияет, поэтому ввиду ограниченности объема статьи может быть опущен.

С точки зрения учета содержательной структуры теста, уровней трудности тестовых заданий и реальной длины сеанса тестирования в качестве примера рассмотрим структуру сеанса при K=3, L=10, N=20. При этом количество элементов пространства $M(m_1, m_2, m_3)$ будет:

$$Z = C_{20-10+3-2}^{3-1} - 3 \cdot C_{20-10+3-2}^{3-1} + 3 \cdot C_{20-22+3-1}^{3-1} = 121 - 3 \cdot C_{11}^2 + 3 \cdot C_1^2 = 121 - 3 \cdot \frac{11 \cdot 10}{2} + 0 = 231 - 165 = 66.$$

Если число тем в тесте $L < \frac{N}{K}$, то сеанс, в котором из каждого уровня трудности в тест попадает только одно задание, не существует. В этом случае из каждой темы в сеанс попадает больше, чем K заданий, но этот вариант в данной статье не рассматривается. Отметим, что для $N = 20$ минимальное число тем в тесте не может быть меньше 7.

На рис. 2 проекция пространства M на плоскость $(m_1,\ m_2)$ без верхних ограничений на $m_i$ показана серым цветом без штриховки, а с ограничениями $m_i \le 10$ – заштрихованная область. Справа от этой области показаны координаты трех различных элементов пространства M.

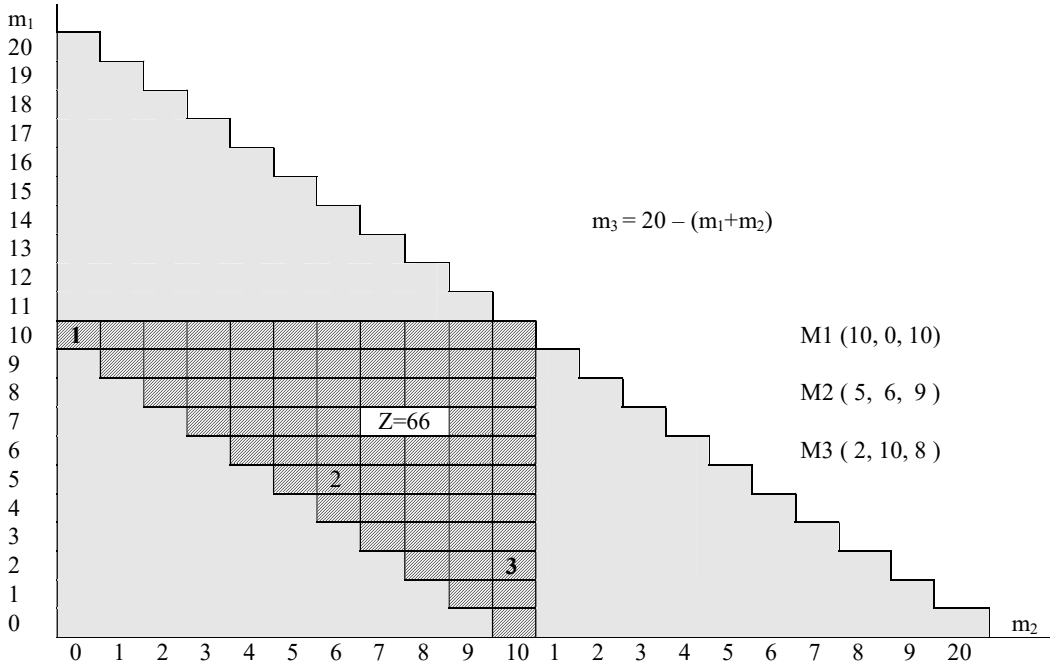



Рис. 2. Модель трехмерного пространства $M(m_1, m_2, m_3)$

Путем несложных расчетов можно рассчитать число элементов $Z$ для любой длины сеанса. Результаты для $N$ от 10 до 20 приведены в таблице.

| N | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Z | 66 | 75 | 82 | 87 | 90 | 91 | 90 | 87 | 82 | 75 | 66 |

Представим табличную модель сеанса тестирования (рис.3) с учетом, что $k_{i,j} = \{0, 1\}$. Для упрощения изложения в качестве первого примера будем рассматривать середины диапазонов уровней трудности тестовых заданий, для которых $p_1 = 0{,}7$, $p_2 = 0{,}5$, $p_3 = 0{,}3$, хотя в базе данных присутствуют реальные коэффициенты трудности.

| | Коэффициенты | | | Темы | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,7 | 0,3 | 0,06 | $k_{1,1}$ | $k_{1,2}$ | $k_{1,3}$ | $k_{1,4}$ | $k_{1,5}$ | $k_{1,6}$ | $k_{1,7}$ | $k_{1,8}$ | $k_{1,9}$ | $k_{1,10}$ |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | $k_{2,1}$ | $k_{2,2}$ | $k_{2,3}$ | $k_{2,4}$ | $k_{2,5}$ | $k_{2,6}$ | $k_{2,7}$ | $k_{2,8}$ | $k_{2,9}$ | $k_{2,10}$ |
| 3 ($m_3$) | 0,3 | 0,5 | 0,07 | $k_{3,1}$ | $k_{3,2}$ | $k_{3,3}$ | $k_{3,4}$ | $k_{3,5}$ | $k_{3,6}$ | $k_{3,7}$ | $k_{3,8}$ | $k_{3,9}$ | $k_{3,10}$ |

Рис. 3. Модель сеанса тестирования в системе OpenTEST2

Данная модель строится на основе координат элемента пространства $M(m_1, m_2, m_3)$. В i-ю строку, соответствующую уровню сложности i, записывается число единиц ($k_{i,j} = 1$),

равное значению координаты $m_i$. Для остальных $(L - m_i)$ элементов i-й строки $k_{i,j} = 0$. На рис. 4 показана модель сеанса тестирования для точки 2 на рис. 2, для которой M2 (5, 6, 9).

| | Коэффициенты | | | ТЕМЫ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,7 | 0,3 | 0,06 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 3 ($m_3$) | 0,3 | 0,5 | 0,07 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Рис. 4. Модель сеанса тестирования для M2 (5, 6, 9)

В данной работе весомозначность всех тем в тесте считается одинаковой, а способ распределения $k_{i,j} = 1$ по темам принят произвольным (случайным).

Из подраздела темы, соответствующего определенному уровню трудности, задания в сеанс тестирования выбираются по случайному алгоритму. Рекомендуется, чтобы в каждом подразделе было не менее 10 тестовых заданий.

Введем оценку достоверности репрезентативной выборки из тестовой базы данных при формировании сеанса тестирования – коэффициент S. Примем, что среди множества вариантов формирования сеансов тестирования максимальной достоверностью, с точки зрения дифференцирующей способности тестовых заданий и максимального покрытия учебного материала, обладает тот вариант, у которого значение S максимально.

Коэффициент S вычисляется как сумма для всех $L$ тем теста произведений средней дисперсии коэффициентов трудности тестовых заданий всех $K$ уровней трудности на коэффициент покрытия учебного материала для каждого уровня трудности темы :

$$S = \sum_{j=1}^{L} D_j C_j ,$$

$$D_j = \frac{\sum_{i=1}^{K} k_{i,j} p_i q_i}{\sum_{i=1}^{K} k_{i,j}} = \frac{k_{1,j} p_1 q_1 + k_{2j} p_2 q_2 + k_{3,j} p_3 q_3}{k_{1,j} + k_{2j} + k_{3,j}} ,$$

$$C_j = k_{1,j}(1 - k_{2,j})(1 - k_{3,j})c_1 + k_{2,j}(1 - k_{1,j})(1 - k_{3,j})c_2 + k_{3,j}(1 - k_{1,j})(1 - k_{2,j})c_3 +$$
$$+ k_{1,j}k_{2,j}(1 - k_{3,j})(c_1 - e_1 + c_2 - e_2) + k_{1,j}k_{3,j}(1 - k_{2,j})(c_1 - e_1 + c_3 - e_3) +$$
$$+ k_{2,j}k_{3,j}(1 - k_{1,j})(c_2 - e_2 + c_3 - e_3) + k_{1,j}k_{2,j}k_{3,j} ,$$

где L – количество тем в тесте; K – количество уровней трудности тестовых заданий; $c_j$, $e_i$ – коэффициенты покрытия и пересечения учебного материала для каждого уровня трудности.

Отметим, что $S_{max}$ => 0,25*L => 2,5, при условии $p_1 = p_2 = p_3 = 0,5$.

Вычисление значений коэффициента S выполняется для всех вариантов сеанса тестирования (элементов пространства M), количество которых приведено в таблице. Так как количество вариантов в общем случае меньше 100, то использовать какие-либо специальные алгоритмы дискретной оптимизации нецелесообразно, достаточно использовать полный перебор вариантов для нахождения максимальных значений коэффициента S.

На рис. 5–8 показаны результаты расчета достоверности 10 сеансов тестирования для разных вариантов p и c с максимальными величинами S для значений K=3, L=10, N=20.

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,7 | 0,3 | 0,06 | 0 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 4 | 3 |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | 10 | 9 | 10 | 8 | 10 | 7 | 9 | 10 | 6 | 8 |
| 3 ($m_3$) | 0,3 | 0,5 | 0,07 | 10 | 10 | 9 | 10 | 8 | 10 | 9 | 7 | 10 | 9 |
| | S | | | 1.748 | 1.726 | 1.721 | 1.705 | 1.695 | 1.684 | 1.67 | 1.669 | 1.662 | 1.649 |

Рис. 5. Структура сеанса тестирования (вариант 1) по уровням трудности

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,8 | 0,3 | 0,06 | 0 | 1 | 1 | 2 | 3 | 2 | 4 | 2 | 5 | 3 |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | 10 | 10 | 9 | 10 | 10 | 8 | 10 | 9 | 10 | 9 |
| 3 ($m_3$) | 0,2 | 0,5 | 0,07 | 10 | 9 | 10 | 8 | 7 | 10 | 6 | 9 | 5 | 8 |
| | S | | | 1.746 | 1.742 | 1.741 | 1.738 | 1.738 | 1.735 | 1.733 | 1.731 | 1.729 | 1.728 |

Рис. 6. Структура сеанса тестирования (вариант 2) по уровням трудности

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,7 | 0,4 | 0,15 | 5 | 4 | 5 | 4 | 3 | 3 | 2 | 2 | 1 | 1 |
| 2 ($m_3$) | 0,5 | 0,5 | 0,15 | 5 | 6 | 10 | 10 | 7 | 10 | 8 | 10 | 9 | 10 |
| 3 ($m_3$) | 0,3 | 0,6 | 0,2 | 10 | 10 | 5 | 6 | 10 | 7 | 10 | 8 | 10 | 9 |
| | S | | | 1.746 | 1.742 | 1.741 | 1.738 | 1.738 | 1.735 | 1.733 | 1.731 | 1.729 | 1.728 |

Рис. 7. Структура сеанса тестирования (вариант 3) по уровням трудности

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,8 | 0,4 | 0,15 | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 6 | 2 | 5 |
| 2 ($m_3$) | 0,5 | 0,5 | 0,15 | 10 | 10 | 10 | 10 | 10 | 10 | 9 | 10 | 8 | 9 |
| 3 ($m_3$) | 0,2 | 0,6 | 0,2 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 4 | 10 | 6 |
| | S | | | 1.575 | 1.567 | 1.56 | 1.552 | 1.545 | 1.537 | 1.516 | 1.506 | 1.494 | 1.478 |

Рис. 8. Структура сеанса тестирования (вариант 4) по уровням трудности

На рис. 9,10 показаны результаты расчетов S для K=3, L=10, N=15.

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,7 | 0,3 | 0,06 | 0 | 1 | 0 | 0 | 2 | 0 | 1 | 1 | 0 | 0 |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | 5 | 4 | 10 | 6 | 3 | 9 | 10 | 5 | 7 | 8 |
| 3 ($m_3$) | 0,3 | 0,5 | 0,07 | 10 | 10 | 5 | 9 | 10 | 6 | 4 | 9 | 8 | 7 |
| | S | | | 1.399 | 1.377 | 1.374 | 1.363 | 1.356 | 1.348 | 1.347 | 1.342 | 1.328 | 1.323 |

Рис. 9. Структура сеанса тестирования (вариант 5) по уровням трудности

| | Коэффициенты | | | Структура сеанса (задания по уровням трудности) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Уровни | p | c | e | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 ($m_1$) | 0,8 | 0,3 | 0,06 | 0 | 1 | 2 | 0 | 1 | 3 | 2 | 0 | 0 | 0 |
| 2 ($m_3$) | 0,5 | 0,4 | 0,07 | 10 | 10 | 10 | 9 | 9 | 10 | 9 | 8 | 5 | 6 |
| 3 ($m_3$) | 0,2 | 0,5 | 0,07 | 5 | 4 | 3 | 6 | 5 | 2 | 4 | 7 | 10 | 9 |
| | S | | | 1.279 | 1.257 | 1.235 | 1.234 | 1.213 | 1.196 | 1.191 | 1.19 | 1.179 | 1.174 |

Рис. 10. Структура сеанса тестирования (вариант 6) по уровням трудности

Анализ приведенных результатов позволяет сделать три основных вывода :

1) наивысшую достоверность имеют сеансы тестирования, у которых хотя бы по одному из уровней трудности заполнены все темы;

2) уменьшение дисперсии трудностей тестовых заданий (увеличение разброса значений $p_i$) смещает структуру сеанса в область заданий среднего уровня трудности;

3) увеличение значений коэффициентов покрытия учебного материала смещает структуру сеанса в область заданий повышенного уровня трудности.

Таким образом, можно сформулировать алгоритм формирования частично адаптивного сеанса тестирования.

1. На основании значений K, L, N находится элемент пространства M (или несколько элементов) с максимальным значением достоверности.

2. На основании координат выбранной точки пространства M формируется модель сеанса тестирования (см. рис. 4).

3. На основании указанной модели из тестовой базы данных выбираются задания из соответствующих тем и уровней трудности.

Отметим, что в каждой теме могут быть свои соотношения числовых значений $p_i$ и $c_i$, но это не влияет на общий алгоритм формирования сеанса тестирования.

В системе OpenTEST2 преподаватель может выбрать упрощенный неадаптивный алгоритм формирования сеанса тестирования (если количественные оценки уровней трудности всех тем фиксированы, и автор теста не задал коэффициентов покрытия учебного материала).

1. В сеанс тестирования из всех тем выбираются задания средней трудности с максимальной дисперсией.
2. Если после этого сеанс не заполнен, то в него из всех тем выбираются трудные задания.
3. Если и после этого сеанс не заполнен, то он дополняется легкими заданиями.

Данный алгоритм формирования сеанса также рекомендуется применять для тестов неизвестного происхождения, которые импортируются в систему OpenTEST2 из внешних источников.

В заключение отметим, что весомозначность отдельных тем в тесте и способ распределения $k_{i,j} = 1$ по темам существенно влияют на структуру сеанса тестирования. Но это предмет дальнейших исследований.

**Выводы**

Анализ расчетов достоверности для сеансов тестирования с разными параметрами показал, что при формировании сеанса многое зависит от соотношения параметров длины сеанса (N) и количества тем в тесте (L) при фиксированном количестве уровней трудности тестовых заданий.

При $N > KL$ сеанса, формируемого по принципу «из каждого подраздела темы в сеанс попадает по одному заданию» не существует, а преподавателю, проводящему тестирование, следует задуматься о соотношении длины сеанса и тематической разбивки учебного материала.

При $N < L$ длина сеанса такова, что задания из некоторых тем вообще не попадают в сеанс. Если эта ситуация преподавателя устраивает, то все в порядке, а если нет, то следует увеличить длину сеанса хотя бы до числа тем в тесте.

При $L \le N \le KL$ длина сеанса считается оптимальной. Кроме того, данное соотношение может помочь автору теста в тематической разбивке учебного материала при заданных временных и количественных ограничениях на длину сеанса тестирования.

**Список литературы: 1.** *Аванесов В.С.* Композиция тестовых заданий. Учебная книга, 3 изд. доп. М.: Центр тестирования, 2002. 240 с. **2.** *Шкиль А.С., Чумаченко С.В., Напрасник С.В., Гаркуша Е.В.* Построение тестовых заданий в системе компьютерного тестирования знаний в OpenTEST2 // АСУ и приборы автоматики. 2006. Вып. 137. С.21-32. **3.** *Карпенко Д.С., Карпенко О.М., Шлихунова Е.Н.* Система автоматического повышения качества тестовых заданий и мониторинг процесса усвоения знаний // Телекоммуникации и информатизация образования. 2001. № 1. С. 42-57. **4.***Сергиенко И.В., Каспшицкая М.Ф.* Модели и методы решения на ЭВМ комбинаторных задач оптимизации. Киев: Наукова думка, 1981.288 с. **5.** *Ежов И.И., Скороход А.В., Ядренко М.И.* Элементы комбинаторики. М.: Наука, 1977. 80 с.

*Поступила в редколлегию 07.12.2006*

**Шкиль Александр Сергеевич,** канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и диагностика цифровых устройств, создание электронных учебников, технологии дистанционного образования. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

**Чумаченко Светлана Викторовна,** канд. физ.-мат. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: технологии тестирования знаний. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: ri@kture.kharkov.ua.

**Напрасник Сергей Владимирович,** инженер тестового центра ХНУРЭ. Научные интересы: системное администрирование сетей Windows и UNIX, проектирование и разработка Internet/Intranet приложений. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

**Бабич Анна Витальевна,** канд. техн. наук, старший преподаватель кафедры АПВТ ХНУРЭ. Научные интересы: сетевые технологии, технологии дистанционного образования. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

**Гаркуша Елена Владимировна,** преподаватель кафедры автоматических систем управления войсками факультета “Телекоммуникации” Полтавского военного института связи.

УДК 658.5

*В.В. СЕМЕНЕЦ, В.Г. ИВАНОВ*

# К ВОПРОСУ ОБ АНАЛИЗЕ СТРУКТУРЫ ЗАДАЧИ ПРОЕКТИРОВАНИЯ ТОПОЛОГИИ МИКРОЭЛЕКТРОННЫХ УСТРОЙСТВ

Рассматриваются задачи проектирования топологии микропроцессорных электронных устройств с помощью экспертных систем. Предлагается алгоритм трассировки, использующий дерево решений с утверждениями и доказательствами.

Современной конструктивной базой создания электронной аппаратуры являются большие, сверхбольшие интегральные схемы, многослойные печатные платы и многокристальные микросборки. Проектирование микроэлектронных устройств (МЭУ) имеет значительный недостаток - проектирование топологии, связанное со значительными временными затратами. Под проектированием топологии МЭУ понимают определение точного расположения электронных элементов на кристалле или подложке и реализацию всех соединений в соответствии с электрической схемой так, чтобы удовлетворялись технологические требования к электрическим связям. Большинство задач топологического проектирования относятся к классу трудно решаемых задач, время решения которых растет экспоненциально с увеличением размерности задач.

Основная задача канальной трассировки – выбор наименьшей ширины канала, достаточной для размещения в нем всех соединений, и назначения соединений на магистрали. Некоторые алгоритмы решения данной задачи нами были рассмотрены в [1]. Проанализировав литературу [2 – 4] и полученные нами результаты, мы пришли к выводу о необходимости дальнейшего исследования структур системы ограничений задачи трассировки многокристальных микросборок.

**1. Алгоритм трассировки, использующий дерево решений**

Шаг 1. Определение кратчайшего остова графа.

Шаг 2. Если локальная степень каждой вершины остова меньше или равна двум, идти к шагу 14. В противном случае идти к шагу 3.

Шаг 3. Определение ребер, которые входят в вершины остового дерева с локальной степенью больше двух (формирование списка $A_s$, $\overline{s = i, M}$).

Шаг 4. Примем $i = 1$.

Шаг 5. Ребру с номером i присвоим бесконечный вес.

Шаг 6. Определение кратчайшего остова графа.

Шаг 7. Если локальная степень каждой вершины остова меньше или равна двум, идти к шагу 9. В противном случае – к шагу 8.

Шаг 8. Формирование списка $B_s$, $\overline{s = i, N}$. (В список $B_s$ включаются ребра остового дерева, которые инцидентны вершинам с локальной степенью больше двух).

Шаг 9. Примем $i = i + 1$.

Шаг 10. Если $i \le M$, то идти к шагу 5, в противном случае – к шагу 11.

Шаг 11. Если список $B_s$ пуст, то идти к шагу 14, в противном случае – к шагу 12.

Шаг 12. Перезапись списка $B_s$ в список $A_s$. $B_s = 0$.

Шаг 13. Идти к шагу 4.

Шаг 14. Конец.

**2. Алгоритм определения кратчайшего остова** [5]

Шаг 1. Пусть $T_s = \{v_s\}$, где $v_s$ – произвольно выбранная вершина, $D_s = \varnothing$ ($D_s$ является множеством ребер, входящих в кратчайший остов).

Шаг 2. Для каждой вершины $v_i \in T_s$ найти вершину $a_j \in T_s$, такую что

$$c(a_j, v_j) = \min_{v_i \in T_S} [c(v_i, v_j)] = B_j,$$

и присвоить вершине $v_j$ пометки $[\alpha_j, \beta_j]$. Если такой вершины $\alpha_j$ нет, т.е. при $\Gamma(v_j) \bigcap T_s = \varnothing$, приписать вершине $v_j$ пометку $[0, \infty]$.

Шаг 3. Выбрать такую вершину $v_{j*}$, что $B_{j*} = \min_{v_j \in T_S} [B_j]$.

Обновить данные $T_S = T_S \bigcup \{v_{j*}\}, D_S = D_S \bigcup \{\alpha_{j*}, v_{j*}\}$.

Если $[T_s] = P$, то остановиться. Ребра в $D_s$ образуют кратчайший остов.

Если $[T_s] \neq P$, то перейти к шагу 4.

Шаг 4. Для всех $v_j \in T_s$, таких, что $v_j \in \Gamma(v_{j*})$, обновить пометки следующим образом.

Если $\beta_j > c(v_{j*}, v_j)$, то положить $\beta_j > c(v_{j*}, v_j)$, $\alpha_j = v_{j*}$ и вернуться к шагу 3.

Если $\beta_j \leq c(v_{j*}, v_j)$, то перейти к шагу 3. Задача коммивояжера с произвольной асимметричной матрицей весов может быть решена за время $T \approx kp^3$.

Рассмотрим две группы правых цепей, у которых существуют вершины с локальной степенью, равной нулю. Из свойств 1–5 следует, что возможно шесть различных вариантов (рис. 1 – 6).

Рассмотрим две группы цепей (рис. 1).

Определим разности

$$\begin{gathered}
\Delta_1 = k - P_{a_1}, \\
\Delta_2 = P_{b_1} - P_{a_2}, \\
\ldots\ldots\ldots\ldots\ldots \\
\ldots\ldots\ldots\ldots\ldots \\
\Delta_i = P_{b_{i-1}} - P_{a_i}, \\
\ldots\ldots\ldots\ldots\ldots \\
\ldots\ldots\ldots\ldots\ldots \\
\Delta_d = \begin{cases} P_{b_{d-1}} - P_{a_d}, & \text{при } d = l+1, \\ P_{b_d}, & \text{при } d = l \end{cases},
\end{gathered} \qquad (1)$$

$$g = \sum_{S=1}^{i} \Delta_S \quad i = 1,2,\ldots,d,$$

где k – количество вершин, локальная степень которых равна нулю.

**Утверждение 1.** Если $g \geq 0$ $i = 1,2,\ldots,d$, то для трассировки двух групп цепей необходимо и достаточно i треков, где $l = \max\{l_1, l_2\}$.

Доказательство. *Необходимость*. Для трассировки каждой группы необходимо $l_i$ треков (длина дуги в графе вертикальных ограничений i-й группы). Тогда очевидно $l = \max\{l_1, l_2\}$.

*Достаточность*. Рассмотрим крайний случай, когда $g_i = 0$, $I = 1,2,\ldots,d$. К вершинам первой группы присвоим пометки от 1 до k, так как $\Delta_1 = 0$, то k – вершинам второй группы присвоим пометки от 1 до k (вершина $b_1$ получит пометку k). Тогда $P_{b_1}$ вершинам первой группы дадим пометки от $k+1$ до $k + P_{b_1}$ (вершина $a_2$ получит пометку $k + P_{b_1}$). Теперь $P_{a_2}$ вершин второй группы получат пометки от k до $k + P_{a_2} - 1$. На основании того, что $g_i = 0$ можно присвоить пометки всем вершинам второй группы. Мы рассматривали наихудший случай. При $g_i > 0$, $I = 1,2,\ldots,d$ пометки будут расставлены аналогичным путем.

Утверждение 1 доказано.

**Следствие 1.** В том случае, когда для некоторого значения k $g_k < 0$, необходимо расставить пометки для $g_1, g_2, ..., g_{k-1} \geq 0$, а для оставшегося графа расставить пометки согласно выражению

$$\min\{\hat{i}_1, \hat{i}_2\} + 1. \tag{2}$$

Для двух групп правых цепей (рис. 2) определим

$$\begin{gathered}
\Delta_1 = P_{a_1} - P_{b_2} + k, \\
\Delta_2 = P_{a_2} - P_{b_2}, \\
\ldots\ldots\ldots\ldots\ldots \\
\ldots\ldots\ldots\ldots\ldots \\
\Delta_i = P_{a_i} - P_{b_i}, \\
\Delta_d = \begin{cases} P_{a_d} - P_{b_d}, & \text{при } d = l, \\ P_{a_d}, & \text{при } d = l+1 \end{cases}, \\
g_i = \sum_{S=1}^{i} \Delta_S \, i = \overline{1,d},
\end{gathered} \tag{3}$$

где k – количество вершин, с локальной степенью равной нулю.

**Утверждение 2.** Если $g_i > 0$, $i = i,d$, то для трассировки двух групп правых цепей необходимо и достаточно i – треков, где $l = \max\{l_1, l_2\}$.

Доказательство. *Необходимость*. Аналогично утверждению 1.

*Достаточность*. Пусть $\Delta_1 = 0$ $k + P_{a_1}$ вершинам первой и второй группы присвоим пометки от 1 до $k + P_{a_1}$. Возникает противоречие, так как вершины $b_1$ и $a_2$ получили одинаковые пометки. Для разрешения этого противоречия необходимо одной из вершин дать пометку, равную $k + P_a + 1$, что эквивалентно увеличению длины пути $i_1$ или $i_2$. Этого противоречия не будет, если $\Delta_1 > 0$. Вершина $a_2$ получит пометку меньшую, чем $b_1$. Тогда для подграфа, состоящего из вершин, образующих подмножества $\tilde{A} = (a_2, a_3, ..., a_d)$ и $B = (b_1, b_2, ..., b_m)$, справедливо утверждение 1. Для $g_i \geq 0$, $i = 2,3,...,d$ необходимо, чтобы пометка вершины $b_1$ была больше пометки вершины $a_2$. Это условие выполнено. Тогда с учетом $\Delta_1 > 0$ мы имеем $g_i > 0$, $i = i,d$ и $l = \max\{l_1, l_2\}$.

Утверждение 2 доказано.

**Следствие 2**. В том случае, когда для некоторого значения k $g_k \leq 0$, необходимо расставить пометки для $g_1, g_2, ..., g_{k-1} > 0$, а для оставшегося расставить пометки согласно выражению (2).

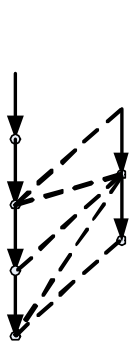
Для групп цепей, показанных на рис. 3 и 4, справедливы утверждения 1 и 2, соответственно, если выполним нумерацию треков снизу вверх. При этом ориентированные ребра поменяют направления, и графы по своей структуре будут соответствовать графам, изображенным на рис. 1 и 2.

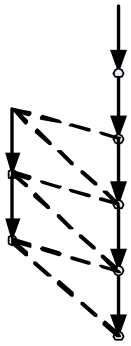
Для групп цепей, изображенных на рис. 5 и 6, необходимо использовать различные комбинации утверждений 1 и 2.

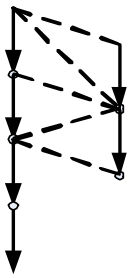
Если рассмотреть левые группы цепей, рассмотренных в работе [1] (на рис. 1), для которых выполняются условия $x_1^i < x_2^i$ $y_1^i < y_2^i$ $x_2^i < x_1^{i+1}$, то для них справедливы все приведенные выше утверждения.

Рассмотрим две группы цепей, одна из которых левая, а другая правая (считаем, что локальная степень любой вершины в ОГВГО отлична от нуля).

ОГВГО приведен на рис. 7.







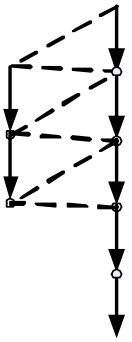


Рис. 1 Рис. 2 Рис. 3 Рис. 4

**Утверждение 3.** Если в ОГВГО, который описывает ограничения на расположение правой и левой группы цепей, не существует вершин с локальной степенью, равной нулю, то трассировка двух групп цепей возможна на $i+1$ треках, где $l = \max\{l_1, l_2\}$.

Доказательство. Для определенности будем считать, что ориентированные ребра первой группы направлены вниз, а второй группы – вверх. Если начальные вершины каждой группы не связаны ребром, то присвоим им пометки с номером один. Аналогично дадим пометки для последующих вершин. На определенном этапе найдутся две вершины, которые связаны неориентированным ребром и обе претендуют на пометку с одинаковым номером $\alpha$. Пусть $P_i$ – локальная степень этих вершин без учета помеченных вершин ($i$ – номер группы); $l_i$ – длина простого ориентированного пути i-й группы цепей в ОГВГО. Возможны следующие случаи:

а) $\tilde{P}_1 = 1, \tilde{P}_2 = 1, l_1 < l_2$,

б) $\tilde{P}_1 = i, \tilde{P}_2 = i, l_1 = l_2$,

в) $\tilde{P}_1 = i, \tilde{P}_2 = i, l_1 > l_2$,

г) $\tilde{P}_1 > i, \tilde{P}_2 = i, l_1 < l_2$,

д) $\tilde{P}_1 > i, \tilde{P}_2 = 1, l_1 = l_2$,

е) $\tilde{P}_1 > i, \tilde{P}_2 = i, l_1 > l_2$,

ж) $\tilde{P}_1 = i, \tilde{P}_2 > i, l_1 < l_2$,

з) $\tilde{P}_1 = i, \tilde{P}_2 > i, l_1 = l_2$,

и) $\tilde{P}_1 = i, \tilde{P}_2 > i, l_1 > l_2$,

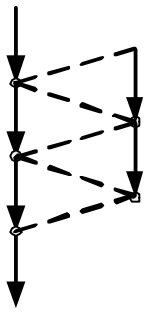
к) $\tilde{P}_1 > i, \tilde{P}_2 > i, l_1 > l_2$,

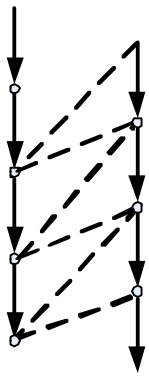
л) $\tilde{P}_1 > i, \tilde{P}_2 > i, l_1 = l_2$,

м) $\tilde{P}_1 > i, \tilde{P}_2 > i, l_1 < l_2$.

После разрешения данного конфликта при дальнейшем распределении пометок хотя и будут встречаться вершины, связанные ребром, однако одна вершина будет иметь пометку меньше $\alpha$, вторая – больше $\alpha$.

Различные варианты ОГВО для правой и левой групп цепей изображены на рис. 5-7.





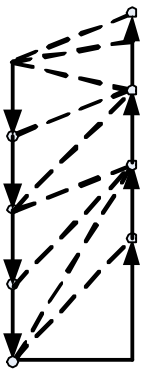


Рис. 5 Рис. 6 Рис. 7

Случай а).

Вершине первой группы присвоим пометку с номером $\alpha+1$, а вершине второй группы – пометку с номером $\alpha$. Тогда $l=\max\{l_1+l,l_2\}=l_2+1$.

Случай б).

Расставим пометки как для случая а), тогда

$l=\max\{l_1+l,l_2\}=l_1+l$.

Случай в).

Вершине второй группы присвоим пометку с номером $\alpha+1$, а вершине второй группы – пометку с номером $\alpha$. Тогда $l=\max\{l,l_2+1\}=l_1$.

Случай г).

Если $l_2-l_1\geq\widetilde{P}_1$, то вершине первой группы присвоим пометку с номером $\alpha+\widetilde{P}_1$, а вершине второй группы – пометку с номером $\alpha$, тогда

$l=\max\{l_1+\widetilde{P}_1,l_2\}=l_2$.

Если $l_2-l_1<P_1$, то вершине первой группы присвоим пометку с номером $\alpha$, а вершине второй группы – $\alpha+1$, тогда

$l=\max\{l_1,l_2+l\}=l_2+1$.

Случай д).

В этом случае справедлива вторая часть случая г) и поэтому $l=\max\{l_1,l_2+l\}=l_2+1$.

Случай е).

Вершине первой группы дадим пометку с номером $\alpha$, а вершине второй группы – $\alpha+1$, тогда $l=\max\{l_1,l_2+1\}=l_1$.

Случай ж).

Вершине второй группы присвоим пометку с номером б, а вершине первой группы – $\alpha+1$, тогда

$l=\max\{l_1,+1,l_2\}=l_2$.

Случай з).

Аналогично случаю ж).

Случай и).

Если $l_1-l_2\geq\widetilde{P}_2$, то вершине первой группы присвоим пометку с номером $\alpha$, а вершине второй группы – $\alpha+\widetilde{P}_2$, тогда

$l=\max\{l_1,l_2+\widetilde{P}_2\}=l_1$.

Если $l_1-l_2<\widetilde{P}_2$, то вершине первой группы присвоим пометку с номером $\alpha+1$, а вершине второй группы – $\alpha$. Поэтому I=max $\{I_1+1,I_2\}=I_1+1$

Случай к).

Если $l_1-l_2\geq\widetilde{P}_2$, то вершине первой группы присвоим пометку с номером $\alpha$, а вершине второй группы – $\alpha+\widetilde{P}_2$, тогда

$l=\max\{l_1,l_2+\widetilde{P}_2\}=l_1$.

Если $l_1-l_2<\widetilde{P}_2$, то вершинам первой группы присвоим пометку с номером $\alpha$, $\alpha+2$, $\alpha+3$ и т. д., а вершинам второй группы – $\alpha+1$, $\alpha+2$, $\alpha+3$ и т. д. Поэтому $l=\max\{l_1,+1,l_2+1\}=l_1+1$.

Случай л).

Он соответствует второй части случая к).

Случай м). Если $l_2-l_1\geq\widetilde{P}_1$, то вершине первой группы присвоим пометку с номером $\alpha+\widetilde{P}_1$, а вершине второй группы – $\alpha$. Поэтому $l=\max\{l_1+\widetilde{P}_1,l_2\}=l_2$.

Если $l_2 - l_1 < \widetilde{P}_1$, то вершинам первой группы присвоим пометки с номером $\alpha$, $\alpha+2$, $\alpha+3$ и т. д., а вершинам второй группы – $\alpha+1$, $\alpha+2$, $\alpha+3$ и т. д.

Поэтому $l = \max\{l_1, +1, l_2+1\} = l_2+1$. Утверждение 3 доказано.

**Следствие 3.** Если в ОГВГО существуют вершины с локальной степенью, равной нулю, то трассировка двух групп цепей возможна на 1, либо на $I+1$ треках ($l = \max\{l_1, l_2\}$).

Приведенные выше утверждения позволяют уменьшить область оптимизации для произвольных классов цепей. Уменьшение области оптимизации повышает вероятность попадания в область допустимых решений. Область оптимизации будет уменьшаться за счет отсечения вариантов трассировки, которые не удовлетворяют системе ограничений:

$$\Omega' : x_i > x_j, i = 1,2,...,N, \quad j : v_j \in \sigma_v^*[v_i], \tag{4}$$

$$x_i > x_j, i = 1,2,...,N, \quad j : v_j \in \sigma_v[v_i], \tag{5}$$

$$x_i - x_j \neq 0, i = 1,2,...,N, \quad j : v_j^* \in \sigma_H[v_i],$$

а область допустимых решений будет оставаться неизменной. Таким образом, если мы найдем хотя бы одну точку из области $\Omega$, которая удовлетворяет всем ограничениям, то тем самым решим задачу 100% трассировки, а затем необходимо минимизировать функцию $F(x) = \sum_{i=1}^{n} I_i(x_i)$. Допустимое множество решений задачи [1] не изменится при замене графа $s_v$ любой его степенью. При этом в графе вертикальных ограничений появляются дополнительные ребра. Ограничения (4) и (5) запишутся в виде:

$$x_i > x_j, i = 1,2,...,N, j : v_j \in \sigma_v^{*N}[v_i], \tag{6}$$

$$x_i < x_j, \ i = 1,2,...,N, j : v_j \in \sigma_v^{N}[v_i]. \tag{7}$$

Определим протяженность $I(v_i, v_j)$ по аналогии с работой [2] как путь наибольшей длины из вершины $v_i$ в $v_j$, тогда ограничения (6) – (7) можно еще более усилить, переписав в виде:

$$x_i \geq x_j, \ +e(v_j, v_i), \ i = 1,2,...,N, \quad j : v_j \in \sigma_v^{N}[v_i], \tag{8}$$

$$x_i \leq x_j, \ +I(v_i, v_j), \ i = 1,2,...,N, \quad j : v_j \in \sigma_v^{N}[v_i]. \tag{9}$$

Определим число протяженности для вершин $v_i$ как $e^-(v_i)$ – максимальный простой путь с конечной вершиной в $v_i$, и $e^+(v_i)$ —максимальный простой путь с начальной вершиной в $v_i$ в графе $\sigma_v$, тогда область $\Omega$ можно уменьшить, определив множества $A_i$ следующим образом:

$$A_i^{'} = \{e^-(v_i)+1, \ e^-(v_i)+2, \ ..., M - e^+(v_i)\}, \tag{10}$$

или, что то же самое, записать явные ограничения на значения переменных:

$$e^-(v_i)+1 \leq x_i \leq M - e^+(v_i).$$

Таким образом, задача [1] может быть записана в виде:

$$X^* = \arg \min_{x \in \Omega' \subset \Omega} F(X), \tag{11}$$

$$F(x) = \sum_{i=1}^{N} I_i(x_i), \tag{12}$$

$$\Omega' : e^-(v_i)+1 \leq x_i \leq M - e^+(v_i), \tag{13}$$

$$\Omega' : \ x_i \geq x_j + e(v_j, v_i), \ i = 1,2,..., N, : v_j \in G_v^{*N}[v_i], \tag{14}$$

$$x_i \leq x_j - e(v_i, v_j), \ i = 1,2,..., N, : v_j \in G_v^{N}[v_j] \ , \tag{15}$$

$$x_i - x_j \neq 0, \ i = 1,2,..., N, \ j : v \in G_H[v_i]. \tag{16}$$

Дополнительной информацией о задаче является то, что множество вершин V разбито на m подмножеств (m вертикалей), обладающих определенными свойствами. Таким образом, имеется m подмножеств: $V_i \leq V, \ i = 1,2..., m$.

Свойства подмножеств:

1) соответствующие подмножества (подграфы) графа горизонтальных ограничений $G_H$ являются полными:

$$G_H[v_j]V_i = V_i \setminus v_j \forall v_j \in V_i, \ i = 1,2,..., m \ ; \tag{17}$$

2) «непрерывность» цепей

$$v_j \in V_{i_1}, \ v_j \in V_{i_2} (i_1 < i_2) \Rightarrow v_j \in V_i, \ i_1 < i < i_2 ; \tag{18}$$

3) «соседние» подмножества могут отличаться не более чем двумя элементами:

$$\mathrm{Card}(V_{i_1} / V_{i_1} V_{i_2})) \leq 2, \ \forall i_1, i_2 : |i_1 - i_2| = i ; \tag{19}$$

4) если подмножество отличается от «соседнего» двумя элементами, между ними должно существовать ребро в графе вертикальных ограничений:

$$\mathrm{Card}(V_{i_1} / V_{i_2}) = 2 \Rightarrow v_i \in H_v[v_i] \ \text{либо} \ v_j \in H_v[v_i] ; \tag{20}$$

$$\forall i_1, i_2 : |i_1 - i_2| = 1, \ \{v_i, v_j\} = V_{i_1} / V_{i_2} .$$

**Утверждение 4.** Если для некоторого подмножества вершин одной вершин мощность множества допустимых значений переменных равна мощности данного подмножества вершин и при исключении из него некоторой вершины мощность множества допустимых значений уменьшается на единицу, то для любого допустимого решения исходной задачи [1] значение переменной, соответствующей данной вершине, должно быть равно данному исключаемому значению. Другими словами, формально:

$$\exists 1 \leq k \leq m, \ V_k' \leq V_k : \mathrm{Card} \underset{j:V_j \in V_k}{\{UA_j'\}} = \mathrm{Card} V_k' \ \text{и}$$

$$\exists V_s \in V_k' : \mathrm{Card} \underset{j:V_j \in (V_k' / V_s)}{\{A_s' / UA_j'\}} = 1 \Rightarrow x_s = * \underset{j:V_j \in (V_k' / V_s)}{A_s' / UA_j} .$$

Доказательство. Исключим из $A_s'$ указанное значение: $A_s' = A_s' / \{A_s' / UA_j'\}$, $j : V_j \in (V_k' / V_s)$. Тогда $\mathrm{Card} \underset{j:V_j \in V_k)}{\{UA_j'\}} < \mathrm{Card} V_k'$ и, следовательно, $\exists V_p, V_q \in V_k', p \neq q$, $p \neq q : x_p^* \equiv x_q^*$. Но, согласно (17), $V_p \in G_H[V_q]$ и, следовательно, согласно (18) $x_p^* \neq x_q$. Приходим к противоречию. Утверждение 4 доказано.

**Следствие 4.** Если для некоторой вертикали мощность соответствующего подмножества вершин равна M и существует единственная вершина с нулевым положительным либо отрицательным числом протяженности, то значение соответствующей переменной в любом допустимом решении задачи должно равняться 1 в случае нулевого положительного числа протяженности.

**Следствие 5.** Для любой вертикали мощность подмножества вершин не может превышать M.

**Утверждение 5.** Если мощность подмножеств вершин для двух соседних вертикалей равна M и подмножества отличаются друг от друга одной вершиной, то в любом допустимом решении задачи значения переменных, соответствующие этим различным для подмножества вершинам, должны быть равны. Формально:

$$\exists l \le k \le m : \mathrm{Card}V_k = \mathrm{Card}M_{k+1} = M,\ \ \mathrm{Card}\{V_k / V_{k+1}\} = 1,$$
$$v_i = V_k / V_{k+1},\ \ v_j = V_{k+1} / V_k \Rightarrow x_j^* = x_j^*.$$

Доказательство:

$$x_i^*, x_j^* \in \underset{s:V_s \in V_k \cap V_{k+1}}{I_M / x_s^*} \quad \mathrm{Card}\left\{ \underset{s:V_s \in V_k \cap V_{k+1}}{I_M / Ux_s^*} \right\} = 1 \Rightarrow x_i^* = x_j^*.$$

Утверждение 5 доказано.

**Утверждение 6.** Если мощность подмножеств вершин для двух соседних вертикалей равна M и подмножества отличаются друг от друга двумя вершинами, то в любом допустимом решении задачи значения переменных**,** соответствующие этим различным для подмножеств вершинам, должны быть попарно равны, причем состав пар однозначно определяется направлением ребер в графе вертикальных ограничений**.**

Формально: $\exists\ 1 \le k \le m : \mathrm{Card}\, V_k = \mathrm{Card}\, V_{k+1} = M, \mathrm{Card}\left\{V_k / V_{k+1}\right\} = 2,$

$$\left\{v_{i1}, v_{i2}\right\} = V_k / V_{k+1}\ , v_{i_1} \in \sigma_v\left[v_{i_2}\right], \left\{v_{j_2}, v_{j_2}\right\} = V_{k+1} / V_k\ ,$$
$$v_{j1} \in \sigma_v\left[v_{j2}\right] \Rightarrow x_{i_1}^* = x_{j_1}^*,\ x_{i_2}^* = x_{j_2}^*, x_{i_1}^* > x_{i_2}^*\ .$$

Доказательство:

$$x_{i_1}^*, x_{i_2}^*,\ \ x_{j1}^*, x_{j2}^* \in \underset{s:V_s \in V_k \cap V_{k+1}}{I_M / x_s^*} \quad \mathrm{Card}\left\{ \underset{s:V_s \in V_k \cap V_{k+1}}{I_M / x_s^*} \right\} = 2\ .$$

Согласно (20) и (14), (15) $x_{i1}^* \ne x_{i2}^*$, $x_{j1}^* \ne x_{j2}^*$. Если $v_{i1} \in \sigma_v\left[v_{i2}\right]$ и $v_{j1} \in \sigma_v[v_{j2}]$, то $x_{i1}^* > x_{i2}^*$, $x_{j1}^* > x_{j2}^*$, следовательно, $x_{i1}^* = x_{j1}^*$, $x_{j2}^* = x_{j2}^*$.

Утверждение 6 доказано.

**Следствие 6.** Используя утверждения 4 и 5, можно провести коррекцию соответствующих множеств $A_i$. В терминах условий утверждения 6 данная коррекция запишется: $\max\{e^-(v_i), e^-(v_j)\} + 1 \le x_i, x_j \le M - \max\{e^+(v_i), e^+(v_j)\}$. В терминах условий утверждения 6 данная коррекция запишется:

$$\max\{e^-(v_{i_1}), e^-(v_{j1})\} + 1 \le x_{i_1},\ x_{j1} \le M - \max\{e^+(v_{i_1}), e^+(v_{j1})\},$$

$$\max\{e^-(v_{i_2}), e^-(v_{j2})\} + 1 \le x_{i_2},\ x_{j2} \le M - \max\{e^+(v_{i_2}), e^+(v_{j2})\}.$$

**Утверждение 7.** Если для двух вершин ОГВГО выполняется условие $e^+(v_i) = e^+(v_j)$ и они связаны неориентированным ребром, то либо $x_i \ge x_j + 1$, либо $x_j \ge x_i + 1$.

Доказательство. Пусть простые пути от вершин $v_i$ и $v_j$ не пересекаются. Тогда вершинам с одинаковым значением $e^+(v_k)$ можно присвоить одинаковые метки, если они не связаны ребром. В противном случае им присваиваются различные метки.

Следствие 7. Рассмотрим два простые пути в ГВО, у которых только первая и последняя вершины общие. Пусть длина первого пути больше второго и равна M. Возьмем предпоследнюю вершину $v_j$ второго пути и найдем непомеченную вершину первого пути, не связанную с данной вершиной ребром и для которой справедливо

$$\min_{k = \overline{I,M}} \left| e^+(v_j) - e^+(v_k) \right|.$$

Присвоим обеим вершинам одинаковые пометки. Выполним аналогичные операции для остальных вершин второго пути. Количество непомеченных вершин второго пути показывает, насколько необходимо увеличить количество треков M, чтобы осуществить 100 % трассировку двух цепей.

**Утверждение 8.** Если для двух вершин $v_i$ и $v_j$ ОГВГО, не связанных ребром, выполняются условия

$$(\sigma_H[v_i] \cup \sigma_H[v_j]) \cap V_s = V_s,\ \mathrm{Card}\, V_s = M,\ s = \overline{i,m},$$

то этим вершинам нельзя присваивать одинаковые метки.

Доказательство. Так как $\mathrm{Card}\, V_S = M$, то это говорит о том, что существует вертикаль в канале шириной M, через которую проходит M цепей, образующих полный подграф в графе горизонтальных ограничений. Этим цепям необходимо и достаточно M пометок. Цепи, соответствующие вершинам $v_i$ и $v_j$, не входят в множество $V_s$. Если мы дадим вершинам $v_i$ и $v_j$ одинаковую метку, то это соответствует тому, что в множестве $V_s$ вводится дополнительная вершина. Отсюда следует, что M пометок уже недостаточно для раскраски множества $V_s$.

**Выводы**

1. Показано, что важна структура задачи трассировки, т.е. показаны возможности ее упрощения и сведения к менее сложным задачам.

2. Исследована структура системы ограничений для задачи трассировки многокристальных микросборок. Показано, что от вида ограничений можно выбрать подходящий алгоритм для решения задачи трассировки, либо значительно уменьшить область оптимизации задачи проектирования топологии.

3. Данная работа доказывает необходимость создания общих (единых) методологических основ для проектирования и организации оптимального функционирования САПР конструктивных узлов и устройств электронной аппаратуры широкого функционального назначения на базе методов синтеза проектных и конструкторских решений, использующих решающие правила.

**Научная новизна.** Обоснована необходимость научного исследования структуры задач для более эффективного их решения в области проектирования и разработки микроэлектронных устройств на базе методов логического синтеза, необходимость создания методологических основ для проектирования и организации оптимального функционирования САПР.

**Практическое значение.** Рассмотренные методы предназначаются для разработки многофункциональных устройств различного назначения. Разработанные методы оптимизации можно применить при проектировании систем автоматизированного проектирования.

**Список литературы: 1.** *Семенец В.В., Иванов В.Г.* Анализ структуры задачи проектирования топологии микроэлектронных устройств / Радиоэлектроника и информатика. 2007. № 3. С. 112–117. **2.** *Оре О.* Теория графов. М.: Наука, 1980. 336 с. **3.** *Кристофидес Н.* Теория графов. Алгоритмический подход / Пер. с англ. М.: Мир, 1978. 432 с. **4.** *Селютин В.А.* Автоматизированное проектирование топологии БИС. М.: Радио и связь, 1983. 112 с. **5.** *Шалтянис В.Р.* Анализ структур задач оптимизации. Вильнюс: Институт математики и кибернетики АИ Литовской ССР, 1989. 120 с.

*Поступила в редколлегию 06.08.2008*

**Семенец Валерий Васильевич,** д-р техн. наук, профессор, первый проректор ХНУРЭ. Научные интересы: САПР, логический синтез. Увлечения и хобби: футбол, бильярд, рыбалка. Адрес: Украина, Харьков, 61166, пр. Ленина, 14, тел. 702-13-64.

**Иванов Виталий Геннадьевич,** соискатель каф. БМЭ ХНУРЭ. Научные интересы: САПР, логический синтез. Увлечения и хобби: футбол, охота. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-64.

УДК 658.52.011.56

*М.В.КОСТИКОВА, В.А.ПЬЯНИДА*

# АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ ТЕОРИИ РАСПИСАНИЙ НА ОСНОВЕ ПРОГНОЗА. ЧАСТЬ 2

Настоящая статья является прямым продолжением работы [1]. В ней исследуется алгоритм решения задачи, сформулированной в указанной работе, приводятся его характеристики, полученные экспериментальным путем.

## 1. Алгоритм

В основу предлагаемого алгоритма положена универсальная процедура последовательного формирования допустимых порядков построения ациклических графов $G$ и вычисления длин их критических путей $L$. В режиме исчерпывающего поиска процедура позволяет перечислить все ациклические графы $G$, вычислить длины их критических путей $L$ и выделить экстремальный граф $G^*$ со значением пути $L^*$. Процедура также позволяет ограничиться построением ряда или одного ациклического графа $G$ и, таким образом, довольствоваться приближенным решением задачи. Действие процедуры продемонстрируем на примере задачи, технологический граф $G_T$ которой изображен на рис. 1.

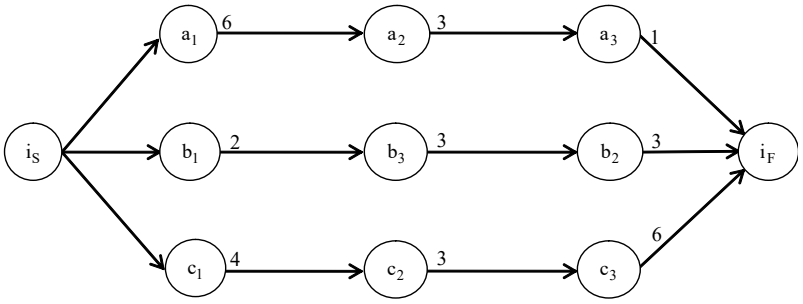



Рис. 1. Технологический граф $G_T$

В алгебраической форме исходные величины представим в виде двух матриц: $S = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_3 & b_2 \\ c_1 & c_2 & c_3 \end{bmatrix}$, $T = \begin{bmatrix} 6 & 3 & 1 \\ 2 & 3 & 3 \\ 4 & 3 & 6 \end{bmatrix}$. Номера строк этих матриц соответствуют номерам обрабатываемых предметов: 1 – a, 2 – b, 3 – c. Номера столбцов отображают естественные порядки выполнения операций предметов: 1, 2, 3. Элементы матрицы $S$ задают имена операций и машин, на которых они выполняются согласно технологиям изготовления предметов. Элементы матрицы $T$ указывают длительности выполнения операций каждого предмета соответствующей машиной. Результаты решения задачи также будем представлять в виде двух матриц: $P = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ b_3 & a_3 & c_3 \end{bmatrix}$, $U = \begin{bmatrix} U_{a1} & U_{b1} & U_{c1} \\ U_{a2} & U_{b2} & U_{c2} \\ U_{b3} & U_{a3} & U_{c3} \end{bmatrix}$ и величины $L$, которую в дальнейшем станем называть длиной расписания. Номера строк матриц $P, U$ определяют номера машин 1, 2, 3. Номера столбцов – естественные последовательности выполнения операций на машинах, найденные в результате решения задачи. Элементы матрицы $P$ указывают предметы в последовательностях их обработки на машинах. Элементы матрицы $U$ задают моменты начала обработки предметов каждой машиной. Величина $L$ представляет собой полное время обработки всех предметов.

Следуя [2], определим понятие множества ожидающих операций $O_k$, $k = 1, 2, ..., \le n$, которые согласно технологическим последовательностям $I_i, i = 1, 2, ..., n$ могут выполняться машинами $q$ из множества $Q$ в данный момент времени $t$. Тогда для момента $t = 0$ это операции $a_1$, $b_1$, $c_1$ – вершины графа $G_T$, в которые заходят только дуги $V_S$. В матрице $S$ – это элементы первого столбца.

Обозначим для $t = 0$ моменты начала ожидающих операций $U_{iq} = 0$, $i = 1, 2, ..., n$, и моменты начала возможной загрузки машин $U_q = 0$, $q \in Q$. По существу это времена завершения фиктивной операции, обозначенной вершиной $i_S$.

Выберем из множества ожидающих операций $\{a_1, b_1, c_1\}$ для выполнения любую операцию, например, $a_1$. Тем самым мы скажем, что вначале на машине 1 будет выполняться операция $a_1$ и, таким образом, она предшествует операциям $b_1$, $c_1$. На технологическом графе $G_T$ (рис. 1) этот выбор отметим дугами порядка, исходящими из вершины $a_1$ и направленными к вершинам $b_1$ и $c_1$. Эти действия проиллюстрированы на рис. 2.

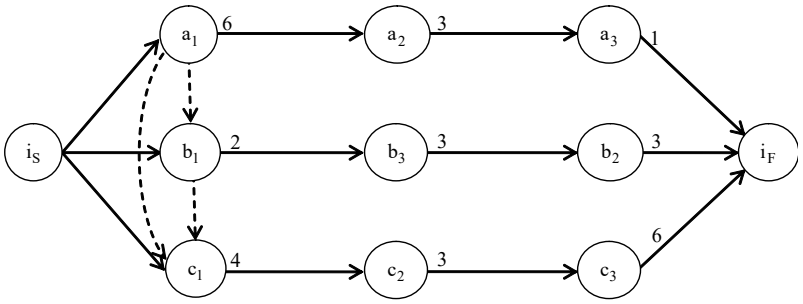



Рис. 2. Выбор выполнения операции $a_1$

Далее вычислим времена возможного начала $U^S_{a1}$ и завершения $U^F_{a1}$ операции $a_1$. Учитывая, что в каждую вершину графа $G$, исключая вершину $i_F$, может заходить не более двух дуг, одна из которых технологическая, а другая – дуга порядка, в общем случае, для получения времени начала выбранной операции следует использовать выражение $U^S_{iq} = \max(U_{iq}, U_q)$, где $U_{iq}$ – момент завершения операции, предшествующей по технологии рассматриваемой операции $iq$, а $U_q$ – момент освобождения машины $q \in Q$, на которой должна выполняться рассматриваемая операция. Время завершения операции вычисляется так: $U^F_{iq} = U^S_{iq} + t_{iq} = \max(U_{iq}, U_q) + t_{iq}$.

На этом основании для операции $a_1$ имеем: $U^S_{a1} = \max(U_{iS}, U_1) = \max(0, 0) = 0$, $U^F_{a1} = 0 + t_{a1} = 0 + 6 = 6$.

В результате получаем, что машина 1 освобождается в момент времени $U_1 = 6$, и операция $a_2$, следующая по технологии за операцией $a_1$, может начаться в момент $U_{a2} = 6$.

Выбранную операцию $a_1$ заносим в матрицу $P$ на первое место первой строки, а момент ее начала $U^S_{a1}$ – в матрицу $U$.

Теперь после выбора операции $a_1$ множество ожидающих операций $O_k$ составляют операции $a_2$, $b_1$, $c_1$. Из них для выполнения можно выбрать любую. Выберем операцию $b_1$ и отметим это дугой порядка, направленной к вершине $c_1$ (см. рис. 2). Тем самым мы скажем, что на машине 1 вслед за операцией $a_1$ будет выполняться операция $b_1$.

Вычислим временные параметры, характеризующие выбор этой операции. Момент ее начала $U^S_{b1} = \max(U_{iS}, U_1) = \max(0, 6) = 6$, и момент завершения $U^F_{b1} = U^S_{b1} + t_{b1} = 6 + 2 = 8$.

Таким образом, время начала операции $b_3$, следующей по технологии за операцией $b_1$, равно $U_{b3} = 8$, а время освобождения машины $U_1 = 8$.

Операцию $b_1$ заносим в матрицу $P$ вслед за операцией $a_1$, а в матрицу $U$ помещаем момент ее начала $U^S_{b1} = 6$.

Очередное множество ожидающих операций $O_k$ образуют операции $a_2$, $b_3$, $c_1$. Выберем для выполнения операцию $c_1$. Тем самым на машине 1 устанавливаем последовательность выполнения операций $a_1$, $b_1$, $c_1$.

Момент начала выполнения операции $c_1$ определяем как $U^S_{c1} = \max(U_{iS}, U_1) = \max(0, 8) = 8$. Момент ее завершения $U^F_{c1} = U^S_{c1} + t_{c1} = 8 + 4 = 12$. В результате получаем, что время начала операции $c_2$, следующей по технологии за операцией $c_1$, $U_{c2} = 12$, а время освобождения машины $U_1 = 12$.

Операцию $c_1$ заносим в матрицу $P$ вслед за операцией $b_1$, а в матрицу $U$ помещаем время ее начала $U^S_{c1} = 8$.

Теперь множество ожидающих операций образуют операции $a_2$, $b_3$, $c_2$. Для них известны технологические времена начала операций $U_{a2} = 6$, $U_{b3} = 8$, $U_{c2} = 12$ и возможные моменты загрузки машин $U_2 = 0$, $U_3 = 0$. Поэтому описанные выше действия выбора операций могут быть продолжены. В связи с тем, что множество операций конечно и согласно процедуре ни одна из них не выбирается более одного раза, выбор операций будет завершен тогда, когда будет выбрана последняя операция и множество $O_k$ окажется пустым.

Предположим, что $O_k = \varnothing$. Тогда полное время обработки всех $n$ предметов может быть найдено так: $L = \max\limits_{i \in I}(U^F_{iq}) = \max\limits_{q \in Q}(U_q)$. Для рассматриваемого примера $L = \max(U^F_{a3}, U^F_{b2}, U^F_{c3}) = \max(U_1, U_2, U_3)$. Заметим, что при вычислении момента начала операции $c_1$ мы учитываем лишь одно значение $U_1 = 8$, найденное как сумма весов дуг порядка $(a_1, b_1) + (b_1, c_1) = 6 + 2 = 8$. Длина дуги $(a_1, c_1)$ транзитивного замыкания была отброшена, так как ее вес меньше $U_1 = 8$. При построении графа $G$ и вычислении длины его критического пути при помощи описанной процедуры дуги транзитивного замыкания всегда могут быть отброшены,так как сумма весов дуг включает все такие дуги: $(a_1, b_1) = 6$ и $(a_1, c_1) = 6$.

Описанная процедура позволяет построить один ациклический граф $G$, вычислить ранние моменты начала выполнения операций и длину $L$ критического пути графа. Иными словами, определить порядки выполнения операций на машинах, составить расписание их работы и найти его длину.

Согласно графу $G_T$ один из вариантов решения будет таким: $P = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ b_3 & a_3 & c_3 \end{bmatrix}$,

$U = \begin{bmatrix} 0 & 6 & 8 \\ 6 & 11 & 14 \\ 8 & 11 & 17 \end{bmatrix}$, $L = \max(U_{c1} + t_{c1}, U_{c2} + t_{c2}, U_{c3} + t_{c3}) = \max(8 + 4, 14 + 3, 17 + 6) = 23$.

Возможность построения других ациклических графов порождается возможностью выбора из множеств ожидающих операций иных элементов. Например, на первом шаге процедуры из множества ожидающих операций $\{a_1, b_1, c_1\}$ могла быть выбрана не операция $a_1$, а операция $b_1$ или $c_1$. Это привело бы к получению других множеств ожидающих операций $\{a_1, b_3, c_1\}$ или $\{a_1, b_1, c_2\}$, которые предоставили бы возможность очередного формирования иных множеств ожидающих операций.

Наглядную картину вариантов выбора операций дает граф типа дерева, первые три и последний ярусы которого изображены на рис. 3.

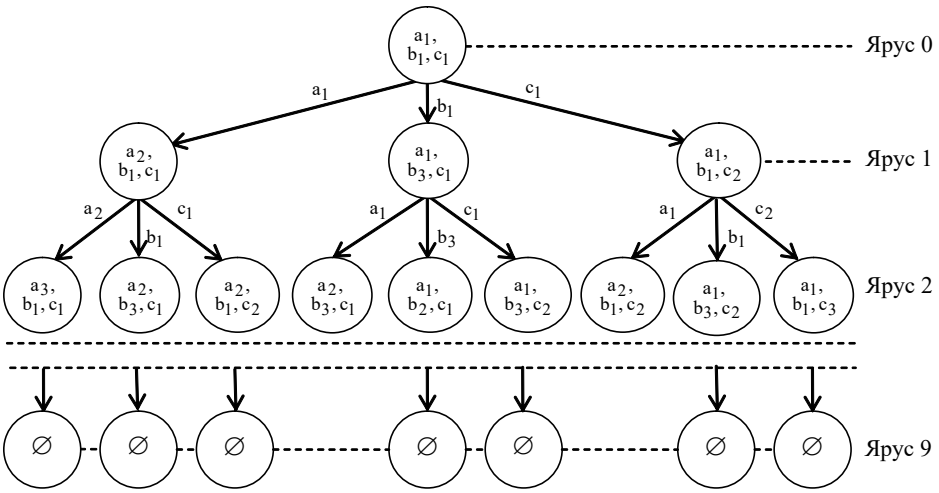



Рис. 3. Дерево вариантов выбора операций

Корневой вершиной этого дерева является исходное множество ожидающих операций $\{a_1, b_1, c_1\}$. Остальные вершины – множества ожидающих операций, порождаемые выбором операций из непосредственно предшествующих множеств. Дуги, исходящие из вершин дерева, идентифицируют выбираемую операцию и заходят в вершины, порождаемые ее выбором. Дерево содержит $n \cdot m + 1$ ярусов, определяемых количеством операций и исходным множеством ожидающих операций. Последний ярус дерева – его листья – представляют собой пустые множества, фиксирующие то обстоятельство, что та или иная последовательность выбора операций завершена. Пути от корня дерева к некоторому его листу, как последовательность дуг, определяют порядки выполнения операций на машинах. Среди этих путей имеется и тот путь (оптимальный), который порождается графом $G^*$ с наименьшим значением критического пути $L^*$.

Таким образом, задача состоит в том, как, реализуя описанную выше процедуру построения ациклического графа $G$ путем выбора операций из множеств ожидающих, каждый раз выбор осуществлять так, чтобы он приводил к построению оптимального пути, определить порядки выполнения операций на машинах, составить расписание их работы и найти его длину.

Некоторые положительные результаты решения этой задачи получены. Однако в целом проблема является открытой.

Интуитивно ясно, что расписание тем короче, чем меньше суммарные простои машин. Поэтому выбор операций должен быть таким, чтобы минимизировать эти простои. Эта идея была положена в основу составления компактных и незадерживающих расписаний [2].

Компактные расписания характеризуются тем, что если расписание составлено, то никакие перестановки выполнения операций на машинах не допускают уменьшения его длины, согласно [2], не допускают сдвига начала операций влево. Незадерживающие расписания образуют подмножество компактных расписаний. Они характеризуются тем, что не допускают задержек в выполнении операций: как только машина освободилась, она сразу же загружается выполнением «ждущей» операции. Множеству компактных расписаний всегда принадлежит расписание с длиной $L^*$, множество незадерживающих расписаний в общем случае этим свойством не обладает [2].

В рамках описанной выше процедуры построения ациклического графа $G$ компактное расписание составляется следующим образом: 1) в множестве ожидающих операций $O_k$ найти ту операцию, которая может завершиться на некоторой машине $q \in Q$ раньше остальных операций $O_k$ и зафиксировать момент ее завершения $U_F$; 2) выделить такие операции $I_k^q \subseteq O_k$, которые можно начать выполнять на машине $q \in Q$ строго раньше момента $U_F$, т.е. для которых $U_{iq} < U_F$; 3) выбрать любую операцию из множества $I_k^q$.

Таким образом, компактные расписания порождаются выбором операций из множеств $I_k^q$, $k = 1, 2, ..., \le n$, $q \in Q$. Экспериментально установлено, что в среднем примерно половина множеств $I_k^q$ имеет единичную мощность [4]. Это свидетельствует о том, что дерево поиска компактных расписаний содержит существенно меньше ветвей, чем общее дерево (рис. 3). Тем не менее, перебор вариантов для поиска лучшего расписания в задачах реальной размерности и для компактных расписаний практически непригоден. По-прежнему не решенной является проблема выбора «лучшей» операции из множеств $I_k^q$, $k = 1, 2, ..., \le n$, $q \in Q$.

Из набора различных способов выбора операций, известных под названием приоритетных правил [2], мы предлагаем выбор на основании прогноза длины расписания.

В множестве $I_k^q$ произвольно выберем операцию $iq \in I_k^q$. По этой операции сформируем очередное множество $I_{k+1}^q$ с самым ранним временем завершения некоторой операции $U_{F_1}$. Затем выберем очередную операцию $iq \in I_k^q$ и для нее сформируем очередное множество $I_{k+1}^q$ с самым ранним временем завершения некоторой операции $U_{F_2}$. Этот процесс закончим выбором последней $p$-й операции из множества $I_k^q$ и определением значения $U_{F_p}$. Операцией, которую следует окончательно выбрать для построения расписания, будем считать ту из них, для которой $U_F = \min\{U_{F_1}, ..., U_{F_p}\}$.

Назовем описанный процесс выбора операции одношаговым прогнозом частичной длины расписания $L$. Возможен также двухшаговый, трехшаговый и т.д. прогнозы, которые осуществляются не на одну операцию вперед, а на две, три и т.д. операции. Например, двухшаговый прогноз может быть осуществлен так. Выберем из множества $I_k^q$ первую операцию $iq \in I_k^q$. Сформируем далее множество $I_{k+1}^q$. Для каждой операции из $I_{k+1}^q$ выделим множества $I_{k+2}^q$, вычислим значения $U_F^{k+1}$ для каждой операции из $I_{k+2}^q$ и определим операцию $iq \in I_{k+1}^q$, порождающую наименьшее значение $U_F$. Тем самым получим последовательность операций $iq \in I_k^q$, $iq \in I_{k+1}^q$. Далее перейдем к выполнению описанных действий для очередной операции из множества $I_k^q$. Прогноз закончим выбором последовательности операций $iq \in I_k^q$, $iq \in I_{k+1}^q$, порождающей наименьшее значение $U_F$.

Продемонстрируем ряд этапов одношагового прогноза на задаче (см. рис. 1).

Множество ожидающих операций $\{a_1, b_1, c_1\}$ является одновременно и множеством $I_1^1$, порождающим компактные расписания, так как $U_F = 2$, а $U_{a1} = 0$, $U_{b1} = 0$, $U_{c1} = 0$ строго меньше $U_F$.

Шаг 1. Выберем операцию $a_1 \in I_1^1 = \{a_1, b_1, c_1\}$ и образуем множество ожидающих операций $O_2^a = \{a_2, b_1, c_1\}$, для которых $U_{a2} = 6$, $U_{b1} = 0$, $U_{c1} = 0$, $U_1 = 6$. Найдем $U_F$ для элементов множества $O_2^a$:

$$U_F^{a2} = \max(U_{a2}, U_2) + t_{a2} = \max(6, 0) + 3 = 9,$$

$$U_F^{b1} = \max(U_{b1}, U_1) + t_{b1} = \max(0, 6) + 2 = 8,$$

$$U_F^{c1} = \max(U_{c1}, U_1) + t_{c1} = \max(0, 6) + 4 = 10.$$

Получаем, что самое раннее время завершения операции $U_F^a = \min\{U_F^{a2}, U_F^{b1}, U_F^{c1}\} = \min(9,8,10) = 8$ определяется операцией $b_1$, выполняемой на машине 1.

Шаг 2. Выберем операцию $b_1 \in I_1^1 = \{a_1, b_1, c_1\}$ и образуем множество ожидающих операций $O_2^b = \{a_1, b_3, c_1\}$, для которых $U_{a1} = 0$, $U_{b3} = 2$, $U_{c1} = 0$, $U_1 = 2$. Найдем $U_F$ для элементов множества $O_2^b$:

$$U_F^{a1} = \max(U_{a1}, U_1) + t_{a1} = \max(0,2) + 6 = 8,$$

$$U_F^{b3} = \max(u_{b3}, U_3) + t_{b3} = \max(2,0) + 3 = 5,$$

$$U_F^{c1} = \max(U_{c1}, U_1) + t_{c1} = \max(0,2) + 4 = 6.$$

Таким образом, самое раннее время завершения операции $U_F^b = \min\{U_F^{a1}, U_F^{b3}, U_F^{c1}\} = \min(8,5,6) = 5$ определяется операцией $b_3$, которая выполняется на машине 3.

Шаг 3. Выберем операцию $c_1 \in I_1^1 = \{a_1, b_1, c_1\}$ и образуем множество ожидающих операций $O_2^c = \{a_1, b_1, c_2\}$, для которых $U_{a1} = 0$, $U_{b1} = 2$, $U_{c2} = 4$, $U_1 = 4$. Найдем $U_F$ для элементов множества $O_2^c$:

$$U_F^{a1} = \max(U_{a1}, U_1) + t_{a1} = \max(0,4) + 6 = 10,$$

$$U_F^{b1} = \max(U_{b1}, U_1) + t_{b1} = \max(0,4) + 2 = 6,$$

$$U_F^{c2} = \max(U_{c1}, U_2) + t_{c2} = \max(4,0) + 3 = 7.$$

В результате, $U_F^c$ для $O_2^c$ равно $\min\{U_F^{a1}, U_F^{b1}, U_F^{c2}\} = \min\{10,6,7\} = 6$. Оно определяется машиной 1.

На основании изложенного получаем, что наименьшее значение $U_F$, полученное при выполнении одношагового прогноза, равно $\min\{U_F^a, U_F^b, U_F^c\} = \min\{8,5,6\} = 5$. Это значение можно получить, если планировать к выполнению на машине 1 операцию $b_1$. Далее описанные действия следует повторять до тех пор, пока не будут выбраны все операции. В заключение отметим, что в тех случаях, когда в результате прогноза будут получены одинаковые $U_F$ для разных работ, следует выбрать ту операцию, которая на соответствующей машине завершается раньше всех других операций.

**2. Экспериментальные характеристики алгоритма**

Экспериментальные характеристики алгоритма представлены статистиками его относительной погрешности, полученными обработкой наблюдений результатов решения ряда наборов «случайных» задач разного размера $N = n \cdot m$ и разных отношений $\frac{n}{m}$.

«Случайными» задачами мы называем такие задачи, в которых матрицы $S, T$ получены при помощи случайных механизмов. Конкретно «случайная» матрица $S$ формируется путем задания случайных перестановок элементов ее строк, что порождает случайные технологические последовательности выполнения операций предметов. Матрица $T$ создается генерацией ее элементов как случайных целых чисел, равномерно распределенных на интервале $(0,100)$.

Для сравнительной оценки статистик погрешностей исследованы два алгоритма, составляющие компактные расписания: с одношаговым прогнозом и при помощи одного из лучших приоритетных правил выбора операций из множеств $I_k^q - \mathrm{NOPNR}$ [2]. Алгоритмы

запрограммированы на языке Турбо Паскаль 7.0. Программы реализованы в среде Delhi на ПК IBM PC.

В целях получения правдоподобных оценок погрешностей алгоритмов при равных значениях $n, m$ использовался следующий ряд размеров задач: 5x5, 10x10, 15x15, 20x20, 25x25, 30x30, 35x35, 40x40. Таким образом, задача самого малого размера – это 25 операций, самого большого – 1600 операций. По каждому размеру генерировалось 100 «случайных» задач. Иными словами, на языке статистики объем выборки составлял 100 элементов. Относительная погрешность алгоритмов при решении каждой задачи вычислялась по отношению к нижней границе $L_N = \max(L_T, L_P)$ [1] в процентах согласно выражению $\Delta = \frac{L - L_N}{L_N} \cdot 100$, гдеL – найденная длина расписания.

В связи с этим получаемые значения погрешностей оказывались несколько завышенными по отношению к точным их значениям.

При разных значениях $\frac{n}{m}$ использовался ряд задач возрастающего отношения $\frac{n}{m}$: $40x40,\ 57x28,\ 80x20,\ 100x16,\ 114x14,\ 126x13,\ 160x10,\ 180x9$.

Для каждой выборки вычислялись следующие статистики погрешности: среднее арифметическое $\Delta_S$, стандартное отклонение $S_\Delta$, максимальное отклонение $\Delta_{max}$, среднее арифметическое отклонение решения задачи от верхней границы $\Delta_S^V$, которое определялось путем усреднения относительного отклонения $\Delta^V = \frac{L_T + L_P - L}{L_T + L_P} \cdot 100$ каждой из 100 задач. Статистики получены средствами пакета Statgraphics.

Результаты экспериментов сведены в три таблицы. В первой из них приведены статистические данные для наборов задач отношения $\frac{n}{m} = 1$, т.е $n = m$, во второй – данные для наборов задач возрастающего отношения $\frac{n}{m} \approx 1, 2, 4, 6, 8, 10, 16, 20$ и $N = \text{const} \approx 1600$, в третьей – минимальные превышения верхней границы над решением.

Таблица 1

| Размер задачи | Алгоритм с прогнозом | | | | Алгоритм с правилом NOPNR | | | |
|---|---|---|---|---|---|---|---|---|
| | $\Delta_S$ | $S_\Delta$ | $\Delta_{max}$ | $\Delta_S^V$ | $\Delta_S$ | $S_\Delta$ | $\Delta_{max}$ | $\Delta_S^V$ |
| 5x5 | 32,75 | 17,58 | 45,24 | 30,24 | 37,16 | 16,28 | 85,11 | 27,95 |
| 10x10 | 40,64 | 11,78 | 80,68 | 26,88 | 47,10 | 12,29 | 81,96 | 23,51 |
| 15x15 | 47,99 | 10,47 | 79,32 | 23,89 | 50,55 | 10,13 | 86,02 | 22,56 |
| 20x20 | 51,61 | 10,07 | 75,49 | 21,94 | 53,60 | 8,85 | 74,77 | 20,91 |
| 25x25 | 53,62 | 8,27 | 73,24 | 21,36 | 57,48 | 7,67 | 78,65 | 19,38 |
| 30x30 | 55,31 | 7,37 | 83,26 | 20,58 | 59,98 | 7,02 | 75,33 | 18,19 |
| 35x35 | 57,23 | 7,21 | 71,40 | 19,73 | 61,29 | 6,80 | 75,33 | 17,65 |
| 40x40 | 58,09 | 6,87 | 79,66 | 19,39 | 63,10 | 6,33 | 76,32 | 16,84 |

Из табл. 1, 2 следует, что средняя погрешность алгоритмов $\Delta_S$ при увеличении размера задачи $N$ и $\frac{n}{m} = 1 = \text{const}$ растет, а при $N = \text{const}$ и увеличении $\frac{n}{m}$ падает. Тем самым экспериментально подтверждаются теоретические выводы о поведении погрешности алгоритмов при изменении параметров $N$, $\frac{n}{m}$, полученные ранее в работе [1].

Таблица 2

| Параметры n , m задачи | Алгоритм с прогнозом | | | | Алгоритм с правилом NOPNR | | | |
|---|---|---|---|---|---|---|---|---|
| | $\Delta_S$ | $S_\Delta$ | $\Delta_{max}$ | $\Delta_S^V$ | $\Delta_S$ | $S_\Delta$ | $\Delta_{max}$ | $\Delta_S^V$ |
| 40x40 | 59,09 | 6,87 | 79,62 | 19,39 | 63,10 | 6,33 | 76,32 | 16,84 |
| 57x28 | 27,93 | 5,06 | 42,64 | 16,34 | 28,78 | 6,10 | 46,34 | 15,78 |
| 80x20 | 10,78 | 3,89 | 19,56 | 14,19 | 15,65 | 4,18 | 24,80 | 10,41 |
| 100x16 | 6,03 | 3,51 | 20,35 | 11,25 | 13,65 | 3,67 | 24,37 | 4,87 |
| 114x14 | 3,74 | 3,10 | 13,50 | 10,09 | 12,54 | 2,70 | 20,19 | 2,47 |
| 126x13 | 2,13 | 2,52 | 12,38 | 9,66 | 12,16 | 2,68 | 18,11 | 0,80 |
| 160x10 | 0,53 | 1,08 | 6,90 | 7,13 | 11,36 | 2,14 | 16,96 | -2,87 |
| 180x9 | 0,26 | 0,53 | 3,65 | 6,01 | 10,71 | 1,96 | 16,07 | -3,79 |

Таблица 3

| Размер задачи / Правило | 5x5 | 10x10 | 15x15 | 20x20 | 25x25 | 30x30 | 35x35 | 40x40 |
|---|---|---|---|---|---|---|---|---|
| NOPNR | 4,63 | 4,80 | 5,66 | 12,58 | 9,53 | 9,91 | 10,72 | 9,30 |
| Прогноз | 4,95 | 5,48 | 9,88 | 10,46 | 7,91 | 7,29 | 13,62 | 8,69 |

Минимальные превышения верхней границы длины расписания над решением (табл. 3), в качестве которой была взята величина $L_T + L_P$, свидетельствуют о том, что из 800 задач размера от $N = 25$ до $N = 1600$, $\frac{n}{m} = 1$ не встретилось задачи, для которой бы эта граница была превзойдена. Это свидетельствует о том, что данная граница для случая $\frac{n}{m} = 1$ является статистически устойчивой, хотя, безусловно, на практике могут встретиться задачи, когда она будет превышена.

Не выполняет принятая величина $L_T + L_P$ роль верхней границы, например, для задач размера $160x10$, $180x9$, что видно из табл. 2.

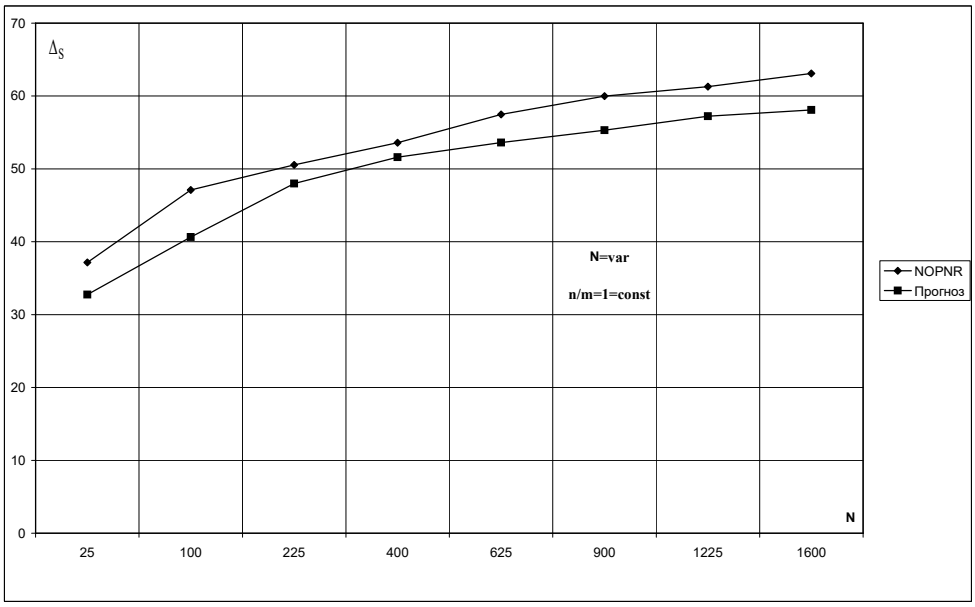



Рис. 4. Диаграммы рассеяния погрешностей $\Delta_S$ для случаев $N \to \infty$, $\frac{n}{m} = 1 = \text{const}$

Законы изменения погрешностей алгоритма наглядно видны из диаграмм рассеяния, построенных на основании табл. 1, 2 и приведенных на рис. 4, 5. Как видно из рис. 4, при $\frac{n}{m} = 1 = \text{const}$ и росте размера задачи $N$ рост средней погрешности $\Delta_S$ алгоритма происходит согласно степенной функции $\Delta_S = a \cdot N^b$ с положительным значением показателя $b < 1$. При этом с увеличением $N$ темпы роста средней погрешности алгоритма замедляются.

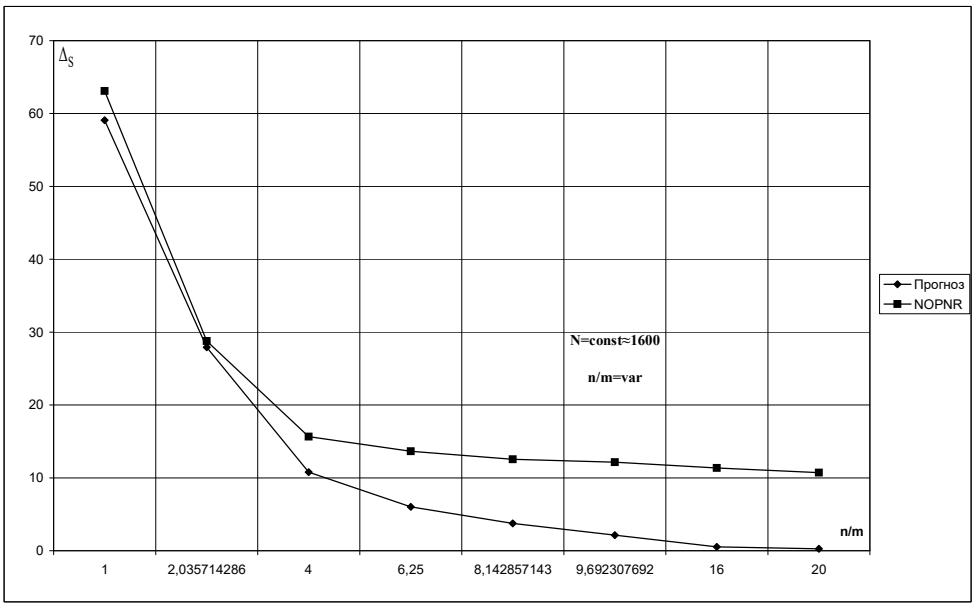



Рис. 5. Диаграммы рассеяния погрешностей $\Delta_S$ для случая $N \to \infty$, $\frac{n}{m} = \text{const} \approx 1600$

Аналитическая зависимость $\Delta_S = a \cdot N^b$ методом регрессионного анализа приближена эмпирическими кривыми $\Delta_S = 22{,}8 \cdot N^{0{,}134}$ (правило прогноза), $\Delta_S = 24{,}87 \cdot N^{0{,}128}$ (правило $NOPNR$).

Для случая $N = \text{const}$, $\frac{n}{m} = \text{var}$ согласно рис. 5 уменьшение погрешностей $\Delta_S$ с ростом отношения $\frac{n}{m}$ происходит по закону гиперболы $\Delta_S = a \cdot \left(\frac{n}{m}\right)^{-b}$. Эмпирические зависимости для правил прогноза и $NOPNR$, полученные методом регрессии, такие: $\Delta_S = 110{,}7 \cdot \left(\frac{n}{m}\right)^{-1{,}8446}$, $\Delta_S = 47{,}71 \cdot \left(\frac{n}{m}\right)^{-0{,}7568}$. При этом характерно, что при стремлении отношения $\frac{n}{m} \to \infty$ для правила прогноза средняя погрешность $\Delta_S$ асимптотически стремится к нулю, а для правила $NOPNR$ это явление не наблюдается. Например, при $\frac{n}{m} = 20$ погрешность $\Delta_S$ (прогноз) равна 0,26%, а погрешность $\Delta_S$ ($NOPNR$) равна 10,71%, т.е. она примерно в 40 раз выше.

Что касается других статистик алгоритмов, представленных в табл. 1, 2, то для обоих правил выбора операций стандартные отклонения $S_\Delta$ с ростом размера задачи $N$ и отношения $\frac{n}{m}$ имеют тенденцию к уменьшению, что свидетельствует о более интенсивной концентрации разброса случайных погрешностей $\Delta$ вокруг среднего значения и, таким образом, более «кучному» решению задач. С ростом отношения $\frac{n}{m}$ для обоих правил алгоритма наблюдается также интенсивное снижение максимального выброса погрешностей $\Delta_{max}$, что свидетельствует о более «точном» решении задач с большим отношением $\frac{n}{m}$.

Таким образом, экспериментально показано, что предложенный новый алгоритм решения задачи теории расписаний с правилом выбора операций на основании прогноза текущей длины расписания по величине погрешности решения превосходит алгоритм с лучшим правилом выбора операций $NOPNR$. На этом основании его можно рекомендовать к практическому использованию в разработках программного обеспечения систем компьютерного составления производственных расписаний.

**Список литературы: 1.** *Костикова М.В., Пьянида В.А.* Алгоритмы решения задач теории расписаний на основе прогноза. Часть 1 // АСУ и приборы автоматики. 2007. **2.** *Конвей Р.В., Максвелл В.П., Миллер Л.В.* Теория расписаний. М.: Наука, 1975. 360 с. **3.** *Канцедал С.А.* Вычислительные алгоритмы решения задач теории расписаний // Изв. АН СССР. Техническая кибернетика. 1982. – №3. С. 42–51. **4.** *Канцедал С.А.* Эффективные алгоритмы упорядочения работ в многостадийных производственных системах дискретного типа. Автореф. дисс. д-ра техн. наук. Харьков: Институт проблем машиностроения АН УССР. 1991. 32 с.

*Поступила в редколлегию 07.09.2007*

**Костикова Марина Владимировна**, канд. техн. наук, доцент кафедры информатики Харьковского национального автомобильно-дорожного университета. Научные интересы: математическое моделирование, теория расписаний и ее применение. Адрес: Украина, 61002, Харьков, ул. Петровского, 25, тел. 707-37-74.

**Пьянида Виктор Александрович**, преподаватель кафедры прикладной математики и информатики Западнодонбасского института экономики и управления. Научные интересы: теория расписаний в моделировании производственных и управленческих процессов. Адрес: Украина, 51400, Павлоград, ул. Парковая, 1а, тел. 311-95.

---

УДК 519.713:681.326

*В.И. ХАХАНОВ, М.А.КАМИНСКАЯ, С.А.ЗАЙЧЕНКО*

# ВЕРИФИКАЦИЯ ЦИФРОВЫХ УСТРОЙСТВ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ АНАЛИЗА ТЕСТОПРИГОДНОСТИ И АССЕРЦИОННЫХ БИБЛИОТЕК

Предлагается метод анализа тестопригодности для операционных и управляющих автоматов, представленных на системном уровне, на основе использования ассерционных библиотек. Описывается методология выбора контрольных точек в устройстве для дальнейшего внедрения ассерций в описание устройства.

## 1. Введение

В связи с интенсивным развитием цифровых устройств и ростом их сложности верификация цифровых устройств становится все более актуальной темой. Актуальность определяется необходимостью повышения быстродействия средств моделирования, улучшения

качества теста и уменьшения его размерности для цифровых систем на кристаллах, имеющих миллионы вентилей. Высокие затраты, обусловленные трудоемкостью верификации функционально и структурно сложных схем, могут достигать 70% от общего времени разработки проекта.

До настоящего времени в связи с надежностью используемых программ синтеза и методик проектирования, самым низким уровнем детализации цифровых устройств для обработки был уровень регистровых передач [1]. Основными элементами для обработки на этом уровне являются регистры и соединения между ними. Проектирование на уровне регистровых передач подобно программированию, например на С. Однако есть небольшие отличия. Например, такие моменты как синхронизация, timing, concurrency, должны быть рассмотрены на этапе программирования. Одним из решений проблемы верификации является использование ассерций [2] – OpenVera Assertions (OVA), PSL/Sugar, Open Verification Libraries (OVL) и C/C++/SystemC™ конструкции как в программном моделировании, так и аппаратно в виде синтезируемых конструкций. Языком описания устройства могут служить такие языки как VHDL, System Verilog, C.

Существует ряд стандартов использования ассерций, таких как стандарты AVM [1], OVM [3], VVM.

Ассерции моделируются отдельно от остального HDL кода. Результат работы ассерций можно увидеть в программе просмотра функционального покрытия (Functional Coverage viewer).

Внедрение ассерций для сложных цифровых устройств – трудоемкий и времязатратный процесс. В настоящее время нет определенного метода выбора критических мест в устройстве для введения ассерций – проектировщик интуитивно внедряет ассерции в программный код. Именно поэтому был разработан метод анализа цифровых устройств на системном уровне описания с дальнейшей выборкой проблемных мест в коде для внедрения ассерций.

Цель работы: существенное уменьшение времени верификации, синтеза тестов и/или повышение степени покрытия неисправностей для заданных входных наборов путем модификации структуры цифрового автомата на основе анализа его тестопригодности.

Объект исследования – цифровой автомат, представленный в виде ориентированного графа.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Задание модели цифрового автомата.
2. Определение модели неисправностей.
3. Оценка тестопригодности графа автомата.
4. Разработка алгоритма модификации графа автомата.
5. Верификация и тестирование метода анализа тестопригодности на реальных цифровых автоматах.

## 2. Структура операционного устройства

По принципу академика В.М. Глушкова любое устройство обработки цифровой информации можно разделить на операционный и управляющий блоки. Такой подход упрощает проектирование, а также облегчает понимание процесса функционирования вычислительного устройства.

Операционный автомат может быть представлен системным уровнем (досинтезное представление устройства, его внутренняя структурная модель), уровнем регистрровых передач, а также вентильным уровнем. Для проведения анализа тестопригодности операционный автомат может быть представлен совокупностью регистров, сумматоров и других узлов, производящих прием из внешней среды и хранение кодов, их преобразование и выдачу результатов работы во внешнюю среду, а также выдачу в управляющий блок и внешнюю среду оповещающих сигналов $U = \{u_1, u_2, ..., u_n\}$, входных-выходных сигналов, модулей памяти, набором счетчиков, триггеров, блоков АЛУ и взаимосвязями между этими компонентами.

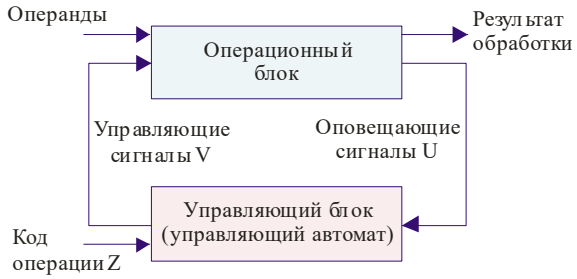



Рис.1. Операционный и управляющий блоки цифрового устройства

Управляющий автомат вырабатывает распределенную во времени последовательность управляющих сигналов $V = \{v_1, v_2, ..., v_m\}$ порождающих в операционном блоке нужную последовательность микроопераций.

Последовательность управляющих сигналов определяется сигналами Z кода операции, поступающими в управляющий блок извне, и сигналами V, зависящими от операндов и промежуточных результатов преобразований.

Декомпозиция цифрового вычислительного устройства представлена на рис. 1.

Пример содержательного графа переходов, синтезируемого в среде Active-HDL [4] автомата со всеми типами операционных вершин, представлен на рис. 2.

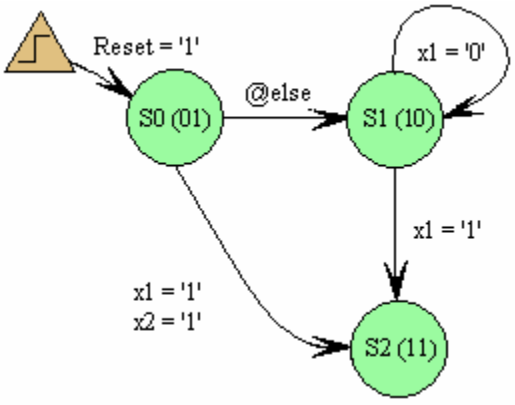



Рис. 2. Граф переходов автомата в среде Active-HDL

В табл. 1 задано кубическое покрытие данного графа в двухтактном алфавите $A^2$.

Для представления двухтактного алфавита используются следующие обозначения: $A^2 = \{Q = 00, E = 01, H = 10, J = 11, O = \{Q,H\}, I = \{E,J\}, A = \{Q,E\}, B = \{H,J\}, S = \{Q,J\}, P = \{E,H\}, C = \{E,H,J\}, F = \{Q,H,J\}, L = \{Q,E,J\}, V = \{Q,E,H\}, Y = \{Q,E,H,J\}, U\}$ – двухтактный алфавит описания состояний/переходов автоматных переменных [4]. Символ U – пустое множество $\varnothing$.

### 3. Определение модели неисправностей

Граф переходов конечного управляющего автомата представляет собой поведенческую модель цифрового устройства. Данное обстоятельство предполагает введение адекватных моделей неисправностей, соответствующих данному уровню. Не затрагивая физическую сущность вводимых далее дефектов, рассмотрим основные неисправности для каждого уровня описания объекта.

*Определение 1*. Модель неисправности вентильного $F^G$, функционального $F^F$, алгоритмического или поведенческого $F^B$ уровней описания цифрового автомата называется импликативной,

Таблица 1

| Вход | Переход | Выход |
|---|---|---|
| Reset = ‘1’ | 00 – 01 | QE |
| Reset = ‘0’ | 01 – 10 | EH |
| x1 = ‘0’ | 10 – 10 | JQ |
| x1 = ‘1’ | 10 – 11 | JE |
| x1 = ‘1’, x2 = ‘1’ | 01 – 11 | EJ |

если она определяется посредством элементарных термов, формирующих его функцию с выполнением условий:

$$M^G \Rightarrow F^G = (L_j \in L, A = \{0,1\});\ M^F \Rightarrow F^F = (C_{ij} \in C, A^2);$$

$$M^B \Rightarrow F^B = (T_{ij} \in T, V = \{V_1, V_2, ..., V_j, ..., V_n\});$$

где $M = \{M^G, M^F, M^B\}$ – модели вентильного, функционального, алгоритмического уровней описания цифрового автомата; $L_j \in L$ – переменная или линия, принимающая двоичное значение из алфавита существенных символов A={0,1}; $C_{ij} \in C$ – координата кубического покрытия, определенная в существенных символах двухтактного алфавита $A^2$; $T_{ij} \in T$ – множество переходов автомата, определенное на векторе вершин $V = \{V_1, V_2, ..., V_j, ..., V_n\}$ графа.

*Определение 2*. Графу переходов цифрового автомата соответствует обобщенная модель неисправностей $F = (F^G, F^F, F^B)$, где $F^G$ – модель неисправности вентильного, $F^F$ – функционального, $F^B$ – алгоритмического или поведенческого уровней описания цифрового автомата. Это связано с тем, что помимо поведенческого уровня представления самого графа, в модели автомата существуют предикатные функции возбуждения, которые могут быть описаны на вентильном и/или функциональном уровнях.

*Определение 3*. Импликативная неисправность графа автомата есть переход $T_{ir} \in T$, который выполняется вместо $T_{ij} \in T$ при условии, что в графе существуют переходы $\{T_{ij}, T_{ir}\} \in T$.

Импликативная модель дефектов $F = (F^G, F^F, F^B)$ не может увеличить пространство состояний конечного автомата S, и находится с последним в отношении $S^F \subseteq S$ [4].

Таким образом, представленная модель дефектов графа автомата может быть использована для проверки неправильных переходов в цифровом изделии.

Одной из распространенных форм исходного описания специализированного цифрового вычислительного устройства является граф переходов автомата. Достоинства упомянутой формы заключаются в наглядности и технологичности представления функций, которые просто преобразуются в табличную форму внутреннего описания автомата в виде таблиц или кубических покрытий. Поэтому такой способ представления проекта имеется во всех системах проектирования ведущих фирм мира. Однако под такую реализацию цифрового устройства практически отсутствуют средства синтеза тестов для верификации и диагностирования проекта. Аналогами тест-генераторов для граф-схем могут служить системы: State-CAD фирмы Visual Software Solutions, Escalade фирмы Mentor Graphics, TetraMax фирмы Synopsys. Поэтому проблема генерации проверяющих тестов для цифровых автоматов, описанных в виде графов переходов и реализуемых на кристаллах ПЛИС, является весьма актуальной для EDA (Electronic Design Automation) рынка.

Другое решение связано с использованием методов анализа тестопригодности и дальнейшей модификацией управляющего либо операционного блока, представленного в виде графа автомата.

Метод анализа тестопригодности основан на топологическом анализе ориентированного графа и его модификации путем разделения тестового и функционального режимов работы в целях улучшения тестопригодности и упрощения решения задач тестирования [5].

**4. Метод TGA – Testability Graph Analysis**

Анализ тестопригодности разрабатываемой модели системы необходимо проводить на всех стадиях проектирования. Методы анализа тестопригодности разрабатывались для различных уровней описания цифрового изделия, в зависимости от тенденций развития

средств проектирования. Первые публикации в этой области принадлежат Рутману [6], Стефенсону [7] и Грасону [8]. Наиболее известные методы анализа тестопригодности на вентильном уровне – это CAMELOT, метод Питерсона, SCOAP, TADATPG [9]. Другие исследования ученых в этой области рассмотрены в работах [10-15].

При этом самый адекватный анализ соответствует наиболее точной модели, которая определяется вентильным уровнем описания, поскольку структура устройства здесь представлена максимально детализированно. Тем не менее, анализ тестопригодности на более высоких уровнях описания, где модель проекта отражает лишь структуру взаимосвязанных компонентов, имеет место, поскольку здесь трудоемкость процедуры анализа минимальна, но оценки тестопригодности и последующая модификация проекта на основе технологий граничного сканирования могут существенно повлиять на стоимость диагностического обеспечения и обслуживания (временные и материальные затраты на синтез тестов, моделирование неисправностей и диагностирование дефектов для каждой стадии проектирования).

На системном уровне устройство представлено в виде совокупности взаимосвязанных компонентов (взаимосвязанных функциональностей устройства – в этом случае для каждого функционального блока устанавливается ассерция и системой диагностирования определяется, в каком именно функциональном блоке существует неисправность. Далее к рассмотрению и дальнейшему тестированию подлежит тот функциональный блок, в котором произошел сбой) или операционного и управляющего автоматов (граф-схема алгоритма). Далее рассматриваются модели проекта системного уровня, представленные в виде граф-схем.

Здесь основная сложность их тестирования заключается в том, что в FSM должны быть проверены все труднодостижимые состояния; тупиковые ситуации или коллизии двух потоков данных; места локализации ветвлений и обратных связей в коде (if, case, loop). Относительно управляющих автоматов можно выделить следующее правило: Каждый FSM должен содержать ассерции, которые проверяют кодирование состояний и переходов [2]. Примеры использования ассеций для верификации FSM:

assert_bits, assert_next_state, assert_value, assert_cycle_sequence, assert_code_distance, assert_next, assert_transition, assert_one_hot, assert_one_cold, assert_zero_one_hot.

Предлагаемый в работе метод заключается в вычислении значений достижимостей вершин, формирующих оценку тестопригодности. Проводится процедура анализа на графе, который строится по размеченной ГСА автомата Мили, Мура. Автомат Мура выбран по причине возможности наблюдения большего количества внутренних состояний, чем в ГСА автомата Мили.

Достижимость (accessibility) – равно управляемость начальной вершины $A(a_1)=1$ или 100%. Значение управляемости каждой последующей вершины зависит от достижимости предыдущей вершины и коэффициента достижимости. Достижимость может принимать относительное значение, лежащее в интервале [0;1]. Значение $A(a_i)=0$ имеет вершина, которая не достижима ни по какому пути в графе. Практически значения достижимости большинства вершин находятся в границах интервала [0;1]:

$$A(a_j)=\sum_{i=1}^{m} A(a_{i-1})\times\frac{1}{n}\times w_{ij},$$

где n – количество возможных наборов, с помощью которых можно попасть в вершину $a_i$, $w_{ij}$ – вес дуги; $w_{ij}=\frac{1}{k_i}$; k – число исходящих дуг из вершины, m – количество управляющих вершин (входящих дуг).

На рис. 3 представлена отмеченная граф-схема алгоритма автомата Мура.

Достижимость всех вершин ($a_i, i=\overline{1,8}$) данного графа представлена следующими оценками:

$A(a_1 = 1)$; $A(a_2 = 1)$;
$A(a_3) = 0{,}2863533467$; $A(a_4) = 0{,}2863533467$;
$A(a_5) = 0{,}035794183$; $A(a_6) = 0{,}008948545$;
$A(a_7) = 0{,}0363545$; $A(a_8) = 0{,}045301$;

Тестопригодность схемы вычисляется по формуле:

$$A_{total} = \frac{1}{m} \sum_{i=1}^{m} A(a_i),$$

где m – количество вершин.

Исходная тестопригодность схемы (до модификации) равна $A_{total} = 0{,}333455$.

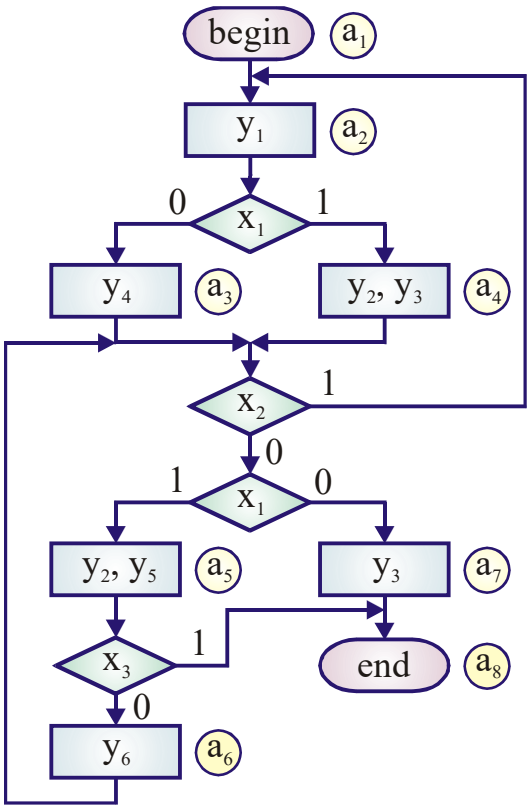



Рис. 3. Отмеченная ГСА автомата Мура

## 5. Стратегия выбора контрольных точек и введения ассерций в код устройства

Стратегия выбора точек для модификации графа состоит в следующем: выбираются 3% линий с минимальными значениями достижимости A. При этом к выбранным линиям добавляются другие, имеющие оценки, равные максимальному из 3% выбранных, если таковые имеются. Обычно линий с одинаковыми значениями показателей сравнительно мало – это особенность метода анализа тестопригодности. 3%-ная квота линий была выбрана из накладываемых ограничений на количество внешних дополнительных контактов в устройстве – не более 5%.

Далее с полученным множеством пересекается множество контрольных точек в коде, выбранных на основе общепринятых правил установки ассерций в коде. В [2] представлен набор правил, предположений и рекомендаций по использованию ассерций. Всего таких правил порядка сорока. Например, для устройств АЛУ ассерции должны проверять правильность выполнения операций, переполнение разрядной сетки и др. Каждый модуль памяти, регистры FIFO, LIFO должны быть проверены ассерциями.

Формула, по которой в коде устройства могут быть выбраны контрольные точки для внедрения ассерций, выглядит следующим образом:

$$Z = \{Y_{TY}\} \cap \{A_{Rules}\} / \{Z_{TY}\},$$

где $\{Y_{TY}\}$ – множество точек, выбранных по методу анализа тестопригодности; $\{A_{Rules}\}$ – множество контрольных точек в устройстве, выбранных на основе правил использования ассерций в коде устройства; $\{Z_{TY}\}$ – множество точек в устройстве, которые были выбраны по методу анализа тестопригодности, но не могут быть использованы в качестве ассерций.

Модификация структуры графа. При выполнении процедуры обхода всех вершин графа и выбора минимального пути для построения теста возникает проблема достижимости состояний. Из-за наличия дополнительных внутренних переменных в функции возбуждения полученный путь может не существовать для содержательного графа. В этом случае необходима модификация пути. Предложенная стратегия модификации состоит в разделении режимов тестирования и нормального функционирования схемы. Для этого на каждую выбранную по стратегии вершину в графе ставится условная вершина, которая предполагает 100% управляемость выбранной вершины графа. Ожидается, что такой подход к модификации устройства позволит при небольших аппаратурных затратах существенно повысить тестопригодность разрабатываемого устройства.

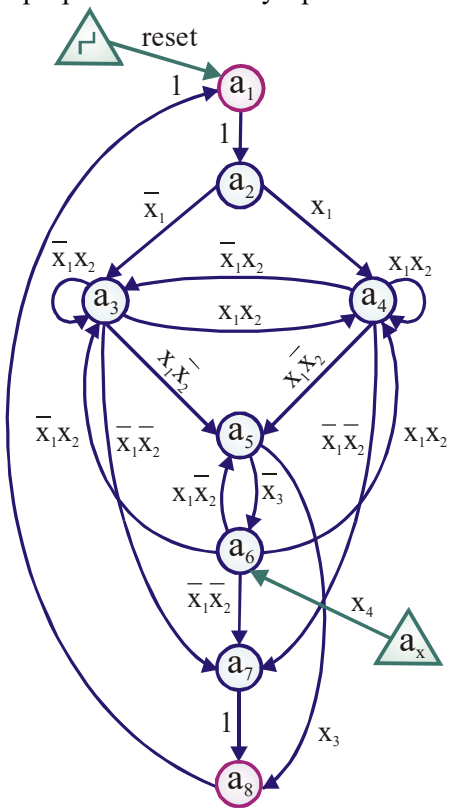



Рис. 4. Модифицированный граф автомата Мура

На примере (рис. 4) видно, что наихудший показатель достижимости имеет вершина $a_6$. Поэтому в структуру графа, представленного на рис. 4, встраивается вершина $a_X$, повышающая управляемость узла $a_6$.

При пересчете показателей достижимости при наличии управляющей вершины получаются значения:

$A(a_1 = 1);$ $A(a_2 = 1);$
$A(a_3) = 0{,}2863533467;$ $A(a_4) = 0{,}2863533467;$
$A(a_5) = 0{,}035794183;$ $A(a_6) = 0{,}05894;$
$A(a_7) = 0{,}039462;$ $A(a_8) = 0{,}0484105;$
$A(a_X) = 1.$

Общая тестопригодность схемы после модификации равна $A_{total} = 0{,}41725$.

Аналогичный анализ может быть проведен как на системном уровне, так и на уровне регистровых передач и на вентильном уровне.

**6. Экспериментальные исследования.**

Вычисление показателей достижимости и применение стратегии модификации автомата производилось на примерах FSM, представленных в табл. 2. В качестве управляемой вершины выбиралась вершина с наименьшим показателем достижимости.

Таблица 2. Результаты анализа тестопригодности

| FSM | $A_{BM}$ | $A_{AM}$ | Differ | Increase |
|---|---|---|---|---|
| CA_M | 0,23306 | 0,3133 | 0,08 | 23,9% |
| Multiplier | 0,44271 | 0,5508 | 0,11 | 25% |
| CSK_M | 0,38360 | 0,5232 | 0,14 | 36,8% |
| MPA_Ma | 0,24702 | 0,4001 | 0,16 | 64,7% |
| Traf_light | 0,33346 | 0,4173 | 0,08 | 23,9% |

Здесь $A_{BM}$ и $A_{AM}$ – показатели достижимости до и после модификации графа.

Введение одной дополнительной вершины в структуру графа позволило повысить качество покрытия неисправностей для вершин графа в среднем на 15–30%, что иллюстрируется данными, представленными в табл.3. Здесь отсутствует оценка качества теста для примера Multiplier, поскольку покрытие неисправностей до модификации уже было равно 100%.

Таблица 3. Результаты анализа покрытия дефектов

| FSM | $Q_{BM}$, % | $Q_{AM}$, % | Increase, % |
|---|---|---|---|
| CA_M | 50,00 | 62,5 | 12,5 |
| CSK_M | 71,43 | 85,71 | 14,28 |
| MPA_M | 37,5 | 75,0 | 37,5 |
| Traf_light | 62,5 | 87,5 | 25 |

**7. Заключение**

Научная новизна: разработан метод расчета показателей тестопригодности, а также стратегии выбора точек для модификации устройства и способ модификации содержательного графа FSM в целях увеличения его тестопригодности и повышения качества покрытия неисправностей тестом.

*Практическая значимость* и преимущества:

1) Разработанный метод анализа тестопригодности позволяет при небольших аппаратурных затратах (до 20%) повысить качество покрытия (на 10-30%) дефектов уже существующего теста.

2) Уменьшает времени синтеза тестов путем введения дополнительных переменных при разделении режимов тестирования и функционирования.

3) Метод позволяет проводить анализ устройства на самых ранних стадиях проектирования и повышать Yield Ratio – выход годных изделий.

Недостатком предложенного метода можно считать то, что анализ на вентильном уровне дает более точные показатели тестопригодности для дальнейшей модификации схемы, чем на более высоких уровнях описания устройства. Также в схему вводится избыточность в виде дополнительных вершин и дуг, что приводит к появлению дополнительных переменных.

**Список литературы: 1.** *Advanced* Verification Methodology Cookbook, Version 2.0, Mark Glasser, Adam Rose, Tom Fitzpatrick, Dave Rich, Harry Foster, July 24, 2006. **2.** *Verification* Methodology Manualfor System Verilog, Janick Bergeron, Eduard Cerny, Alan Hunter, Andrew Nightingale, 2006 Synopsys, Inc. and ARM Limited. 528 p. 3.OVM Class Reference, Version 1.0.1, February 2008. 286 p. **4.** *Хаханов В.И., Ковалев Е.В., Ханько В.В., Масуд М.Д. Мехеди.* Система генерации тестов для проектирования цифровых автоматов в среде Active-HDL // АСУ и приборы автоматики. Харьков. 2000. Вып. 111. С. 15-22. **5.** *Кривуля Г.Ф., Хаханов В.И., Ковалев Е.В.* Проектирование тестов для цифровых устройств на основе FPGA, CPLD // Информационно-управляющие системы на железно-дорожном транспорте. 2000. № 4. С. 120-121. **6.** *Rutman R. A.* Fault Detection Test Generation for Sequential Logic Heuristic Tree Search." IEEE Computer Repository Paper No. R-72-187. 1972. **7.** Grason J., "TMEAS - A Testability Measurement Program," Proc. 16th Design Automation Conf. P. 156-161, June, 1979. **8.** J. Grason and A. W. Nagel, "Digital Test Generation and Design for Testability." Journal Digital Systems, Vol.5, No. 4. P. 319-359, 1981. **9.** *Кулак Э.Н., Каминская М.А.* Эвристический метод анализа тестопригодности для тестирования цифровых схем детерминированным тестом // Радиоэлектроника и информатика. 2005. № 3. С. 113-119. **10.** *Parker K.P., McCluskey E.J.* Probabilistic Treatment of General Combinational Networks // IEEE Trans. on Computers. vol. C-24. no. 6. 1975. P. 668–670. **11.** Grason J. TMEAS - A Testability Measurement Program // Proc. 16th

Design Automation Conf. 1979. P. 156-161. **12.** *Larsson E., Peng Z.* A Behavioral-Level Testability Enhancement Technique // IEEE European Test Workshop. Constance.– Germany. 1999. **13.** *Larsson E., Peng Z.* Testability Analysis of Behavioral-Level VHDL Specifications // IEEE European Test Workshop.– 1998. **14.** Flottes M. L., Pires R., Rouzeyre B. Analyzing Testability from Behavioral to RT Level // Proc. European Design&Test Conf. 1997. P.158-165. **15.***Zdenek Kotasek, Richard Ruzicka, Josef Strnadel, Jan Hlavicka.* Interactive Tool for Behavioral Level Testability Analysis // Proceedings of the IEEE ETW2001.- Stockholm.- SE. 2001. P. 117-119.

*Поступила в редколлегию 20.09.2007*

**Хаханов Владимир Иванович**, декан факультета КИУ, доктор технических наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование цифровых систем. Увлечения: футбол, горные лыжи, путешествия. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua

**Каминская Марина Александровна,** аспирантка кафедры АПВТ ХНУРЭ, инженер по тестированию компании ИП «Интспеи-Украина». Научные интересы: тестопригодное проектирование, моделирование и верификация цифровых систем на кристаллах. Увлечения: литература, музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 7021326. E-mail: maryna4329@kture.kharkov.ua.

**Зайченко Сергей Александрович,** аспирант кафедры АПВТ ХНУРЭ, начальник отдела разработки компании Aldec-Kharkov Ltd. Научные интересы: системы автоматизированного проектирования, моделирования и верификации цифровых систем на кристаллах. Увлечения: литература, музыка, футбол. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (097)-367-62-93. E-mail: Sergei.Zaychenko@aldec.com.

---

УДК 004.652

*В.В. ЕВСЕЕВ, Ю.В. ШОВКОПЛЯС, Н.В. САМОЙЛЕНКО*

# МОДЕЛИРОВАНИЕ РБД С ПРОИЗВОЛЬНОЙ ТОПОЛОГИЕЙ

Рассматривается задача моделирования распределенных баз данных (РБД) с произвольной топологией, а также его аналитические и имитационные методы. Описываются результаты, которые подверждают эффективность использования данных методов в зависимости от различных исходных данных. Даются рекомендации по дальнейшему использованию предложенных методов моделирования.

## Введение

В настоящее время в связи с использованием РБД в различных технических, экономических и социальных системах, имеющих разные топологические структуры, топология самих РБД может также существенно варьироваться. Можно выделить следующие основные топологические структуры РБД: полносвязную, ячеистую, шинную, структуру «звезда» и кольцевую [1]. Таким образом, актуальной является задача формализации процесса моделирования распределенных баз данных независимо от их топологической структуры в реальных условиях функционирования, т.е. в условиях, предполагающих отказы оборудования, линий связи, случайный характер возникновения запросов пользователей.

## 1. Постановка задачи

Для различных топологических структур РБД принято разрабатывать различные математические модели, отличающиеся как необходимой исходной информацией об РБД, так и методами моделирования [2]. В данном исследовании предполагается разработка моделей РБД с произвольной топологией. Это обусловлено тем, что такой подход позволяет создать достаточно универсальные модели, описывающие широкий класс РБД.

Модели функционирования РБД предназначены для количественного описания зависимостей между параметрами, определяющими объемно-временные характеристики потоков входной информации, основные качественные характеристики РБД [3], эффективность преобразования входных потоков информации в выходные. В настоящее время применяются в основном имитационные и аналитические методы моделирования РБД [4].

Имитационные методы обычно представляются в виде специального алгоритма, воспроизводящего элементарные составляющие процесса функционирования объекта с сохранением их логической последовательности и последовательности протекания во времени. Основным достоинством имитационных методов является возможность описания с их помощью РБД практически любой сложности. Аналитическая модель представляется в виде совокупности математических формул и выкладок. Среди достоинств аналитических методов можно выделить их высокую точность и то, что при достаточно высокой сложности моделируемой РБД их использование позволяет существенно снизить аппаратные и временные затраты на моделирование [5].

Для получения основных показателей эффективности функционирования РБД по ряду причин целесообразно применять теоретико-вероятностные методы, в частности методы теории массового обслуживания [6].

Цель работы – провести сравнение имитационных и аналитических методов моделирования РБД с произвольной топологией, используя для описания РБД методы теории массового обслуживания.

Задачи: построить математические модели РБД с произвольной топологией, применяя имитационное и аналитическое моделирование и рассматривая РБД в виде системы массового обслуживания; разработать программное средство реализации полученных моделей и провести сравнительный анализ используемых математических моделей с помощью разработанного программного средства.

**2. Разработка аналитической модели**

Если узлы распределенной базы данных интерпретировать как обслуживающие приборы, а пользовательские запросы – как обслуживаемые заявки, то РБД можно представить в виде многолинейной многофазной (сетевой) системы массового обслуживания. Известно, что теория многофазных СМО в настоящее время развита лишь фрагментарно. В связи с этим при решении практических задач соответствующую многофазную СМО в большинстве случаев рассматривают как совокупность независимых однофазных систем, как это сделано, например, в работе [3]. Исходя из этого утверждения, в разрабатываемой многофазной системе массового обслуживания все операционные характеристики будут рассчитываться в отдельности для каждой из тех однофазных систем, на которые можно декомпозировать исходную многофазовую систему, а затем для всей системы будут вычисляться усредненные значения этих характеристик, являющихся, по сути, показателями эффективности всей многофазной системы.

В качестве основных показателей эффективности функционирования СМО будем рассматривать [6]: $L_S$ - среднее число заявок на обслуживание, находящихся в системе; $L_q$ - среднее число заявок на обслуживание, находящихся в очереди; $W_S$-средняя продолжительность пребывания заявки на обслуживание в системе; $W_q$-средняя продолжительность пребывания заявки на обслуживание в очереди. Для однофазных систем между $L_S$ и $W_S$ (как и между $L_q$ и $W_q$) существует строгая взаимосвязь [7], так что, зная числовые значения одной из этих величин, можно легко найти значение другой величины. В частности, если частота поступлений в систему заявок на обслуживание (интенсивность поступления требований) равняется $\lambda$, то имеем:

$$L_S = \lambda W_S, L_q = \lambda W_q.$$

Если средняя скорость обслуживания заявок равняется $\mu$, то справедливы следующие соотношения:

$$W_S = W_q + 1/\mu;\ L_S = L_q + \lambda/\mu;$$

Пусть $\rho = \lambda/\mu < 1$, тогда справедливы следующие зависимости [6]:

$$L_S = \rho/(1-\rho), W_S = L_S/\lambda = 1/[\mu(1-\rho)]; \quad (1)$$

$$L_q = L_S - \lambda/\mu = \rho^2/(1-\rho), W_q = L_q/\lambda = \rho/[\mu(1-\rho)]. \quad (2)$$

Отсюда очевидно, что для определения основных операционных характеристик СМО (или показателей качества функционирования РБД) достаточно информации лишь о частоте поступлений заявок на обслуживание в систему, скорости их обслуживания и о структуре РБД [8]. Как правило, эти исходные данные задаются исследователем при моделировании.

Известно, что для описания непрерывных стохастических моделей в виде типовых математических схем используют Q-схемы [9] (рис. 1).

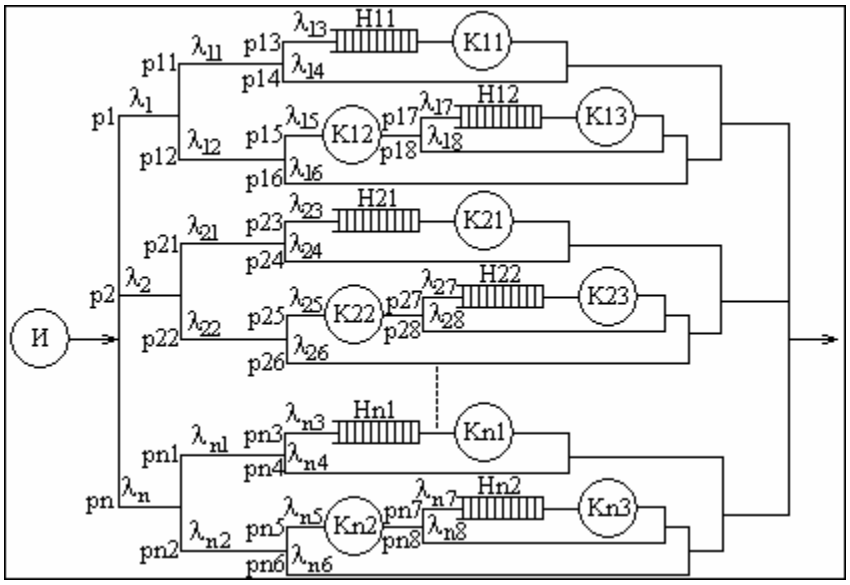



Рис. 1. Q-схема модели

Составляющие Q-схемы: И – источник заявок; К – каналы обслуживания.

Пусть Т – количество узлов РБД; $f_T$ – количество оптимальных маршрутов между Т-м узлом-источником и всеми остальными узлами; $V_{fT}$ – количество линий связи на пути следования запроса из Т-го узла-источника в удаленный узел; $K_{n1}$ (n=1,..,T) – процесс обслуживания запроса в n-м узле-источнике; $K_{n2}$ (n=1,..,T) – процесс передачи запроса по линиям связи от n-го узла; $K_{n3}$ (n=1,..,T) – процесс обслуживания в удаленном узле запроса, возникшего в n-м узле; Н – накопители (очереди заявок); $P_n$ (n=1,..,T) – вероятности возникновения запроса на n-м узле ( $\sum_{n=1}^{T} P_n = 1$ ). Вводится допущение о том, что вероятности возникновения запросов на каждом из узлов равны, т.е. $P_n = \frac{1}{K}, n = \overline{1, T;}$ $P_{n1}$ (n=1,..,T) – вероятность того, что запрос будет удовлетворен в узле-источнике; $P_{n2}$ (n=1,..,T) – вероятность того, что запрос не будет удовлетворен в узле-источнике; $P_{n3}$ (n=1,..,T) – вероятность безотказной работы n-го узла-источника; $P_{n4}$ (n=1,..,T) – вероятность выхода из строя n-го узла-источника при поступлении на него запроса (задается при моделировании); $P_{n5}$ (n=1,..,T) – средняя вероятность безотказной работы всех линий связи на маршруте движения запроса из n-го узла-источника до удаленного узла; $P_{n6}$ (n=1,..,T) – средняя вероятность потери запроса из-за отказа линии связи при передаче его из n-го узла-источника в удаленный узел, содержащий необходимый файл; $P_{n7}$ (n=1,..,T) – вероятность безотказной работы n-го удаленного узла; $P_{n8}$ (n=1,..,T) – вероятность выхода из строя n-го удаленного узла (задается при моделировании); $\lambda_n$ (n=1,..,T) – интенсивность входного потока заявок в СМО, описывающей работу n-го узла; $\lambda_{n1}$ (n=1,..,T) – интенсивность входного потока заявок в СМО, описывающей работу n-го узла-источника; $\lambda_{n2}$ (n=1,..,T) – интенсивность потока заявок, возникающего в результате передачи запроса из n-го узла-

источника в удаленный узел; $\lambda_{n3}$ (n=1,..,T) – интенсивность входного потока заявок в СМО, описывающей обработку запросов в n-м узле-источнике; $\lambda_{n4}$ (n=1,..,T) – интенсивность потока заявок, возникающего в результате отказов оборудования n-го узла-источника; $\lambda_{n5}$ (n=1,..,T) – интенсивность входного потока заявок в СМО, описывающей передачу запросов по линиям связи к удаленному узлу, в который был переслан запрос из n-го узла-источника; $\lambda_{n6}$ (n=1,..,T) – интенсивность потока заявок, возникающего в результате выхода из строя линий связи, по которым осуществляется передача запроса к удаленному узлу; $\lambda_{n7}$ (n=1,..,T) – интенсивность входного потока заявок в СМО, описывающей обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника; $\lambda_{n8}$ (n=1,..,T) – интенсивность потока заявок, возникающего в результате выходов из строя удаленного узла, в который был переслан запрос из n-го узла-источника.

Пусть $L_n$ – количество файлов в каждом конкретном узле, $M$ – количество всех видов файлов, которые могут располагаться в узлах РБД.

Тогда справедливы следующие соотношения:

$$P_{n1} = \frac{L_n}{M}, n = \overline{1,T};\ P_{n2} = 1 - \frac{L_n}{M} = 1 - P_{n1}, n = \overline{1,T};\ P_{n3} = 1 - P_{n4}, n = \overline{1,T};$$

$$P_{n5} = 1 - P_{n6} = 1 - \frac{1}{T}\sum_{m=1}^{T}\left[\frac{1}{f_T}\sum_{j=1}^{f_T}\left(1 - \prod_{i=1}^{v_{fj}}(1 - P_{\text{отк.i}})\right)\right],\ n = \overline{1,T};$$

$$P_{n6} = \frac{1}{T}\sum_{m=1}^{T}\left[\frac{1}{f_T}\sum_{j=1}^{f_T}\left(1 - \prod_{i=1}^{v_{fj}}(1 - P_{\text{отк.i}})\right)\right],\ n = \overline{1,T};\ P_{n7} = 1 - P_{n8}, n = \overline{1,T};$$

где $P_{\text{отк.i}}$ (i=1,.., $V_{fT}$) – вероятность выхода из строя i-й линии связи f-го маршрута из узла T (вероятности выхода из строя каждой из линий связи задаются при моделировании).

Зная интенсивность входного потока заявок $\lambda$, рассчитаем интенсивности входных потоков заявок для каждой из однофазных СМО, на которые можно декомпозировать исходную многофазную СМО, изображенную на рис. 1.

Интенсивность входного потока заявок в СМО, описывающей работу n-го узла, можно рассчитать по формуле: $\lambda_n = \lambda P_n, n = \overline{1,T};$ интенсивности входных потоков частных СМО $\lambda_{n1} \ldots \lambda_{n8}$ могут быть рассчитаны по следующей формуле: $\lambda_{nj} = \lambda_n P_{nj}, n = \overline{1,T}; j = \overline{1,8};$ где j – номер соответствующей однофазной СМО, на которые разбивается многофазная система.

Пусть $\mu_{n1}$ (n=1,..,T) – скорость обслуживания заявок в каналах, имитирующих обработку запросов в n-м узле-источнике (в данной работе принимается допущение о равном объеме всех запросов). Эта величина задается при моделировании;

$\mu_{n3}$ (n=1,..,T) – скорость обслуживания заявок в каналах, имитирующих обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника. Данная величина задается при моделировании.

Пусть $V_ç$ – установленный объем запросов (Kb); $\mu_{n2}$ (n=1,..,T) – скорость передачи запросов по линиям связи до удаленного узла, содержащего необходимый файл. Эта величина может быть вычислена по формуле:

$$\mu_{n2} = \frac{1}{\sum_{i=1}^{m}\frac{V_ç}{S_i}}, n = \overline{1,T},$$

где $S_i$ – скорость передачи информации (Kb/s) по i-й линии связи маршрута между n-м узлом-источником и удаленным узлом, содержащим необходимый файл (данная величина задается при моделировании).

Имея информацию о частотах поступлений заявок на обслуживание в однофазные системы $\lambda$ и о скоростях обслуживания заявок в этих системах, рассчитаем операционные характеристики каждой из однофазных СМО, а затем операционные характеристики всей рассматриваемой многофазной СМО.

Пусть $L_{Sn1}$ – среднее число заявок на обслуживание, находящихся в системе, описывающей обработку запросов в n-м узле-источнике; $W_{Sn1}$ – средняя продолжительность пребывания заявки на обслуживание в системе, описывающей обработку запросов в n-м узле-источнике.

Тогда по формуле (1) рассчитаем значения этих величин:

$$\rho_{n1} = \lambda_{n3} / \mu_{n1} = \lambda_{n1} P_{n3} / \mu_{n1} = \lambda_n P_{n1} P_{n3} / \mu_{n1} =$$

$$= \lambda P_n P_{n1} P_{n3} / \mu_{n1} = \lambda \frac{1}{K} \frac{L_n}{M} (1 - P_{n4}) / \mu_{n1}, n = \overline{1, T};$$

$$L_{Sn1} = \rho_{n1} / (1 - \rho_{n1}), n = \overline{1, T}; \quad W_{Sn1} = 1 / [\mu_{n1} (1 - \rho_{n1})], n = \overline{1, T}.$$

Пусть $L_{qn1}$ – среднее число заявок на обслуживание, находящихся в очереди, для системы, описывающей обработку запросов в n-м узле-источнике; $W_{qn1}$ – средняя продолжительность пребывания заявки на обслуживание в очереди, для системы, описывающей обработку запросов в n-м узле-источнике.

Тогда по формуле (2) рассчитаем значения этих величин:

$$L_{qn1} = L_{Sn1} - \lambda_{n3} / \mu_{n1} = \rho_{n1}^2 / (1 - \rho_{n1}); \; W_{qn1} = L_{qn1} / \lambda_{n3} = \rho_{n1} / [\mu_{n1} (1 - \rho_{n1})], \quad n = \overline{1, T}.$$

Пусть $L_{Sn3}$ – среднее число заявок на обслуживание, находящихся в системе, описывающей обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника; $W_{Sn3}$ – средняя продолжительность пребывания заявки на обслуживание в системе, описывающей обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника.

Тогда по формуле (1) рассчитаем значения этих величин:

$$\rho_{n3} = \lambda_{n7} / \mu_{n3} = \lambda_{n5} P_{n7} / \mu_{n3} = \lambda_{n2} P_{n5} P_{n7} / \mu_{n3} = \lambda_n P_{n2} P_{n5} P_{n7} / \mu_{n1} \quad , n = \overline{1, T};$$

$$L_{Sn3} = \rho_{n3} / (1 - \rho_{n3}), n = \overline{1, T}; \quad W_{Sn3} = 1 / [\mu_{n3} (1 - \rho_{n3})], n = \overline{1, T}.$$

Пусть $L_{qn3}$ – среднее число заявок на обслуживание, находящихся в очереди, для системы, описывающей обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника; $W_{qn3}$ – средняя продолжительность пребывания заявки на обслуживание в очереди для системы, описывающей обработку запросов в удаленном узле, в который был переслан запрос из n-го узла-источника.

Тогда по формуле (2) рассчитаем значения этих величин:

$$L_{qn3} = L_{Sn} - \lambda_{n7} / \mu_{n3} = \rho_{n3}^2 / (1 - \rho_{n3}); \; W_{qn3} = L_{qn3} / \lambda_{n7} = \rho_{n3} / [\mu_{n3} (1 - \rho_{n3})], \quad n = \overline{1, T}.$$

Пусть $L_{Sn2}$ – среднее число заявок на обслуживание, находящихся в системе, описывающей передачу запросов по линиям связи между n-м узлом-источником и удаленным узлом, содержащим необходимый файл; $W_{Sn2}$ – средняя продолжительность пребывания заявки на обслуживание в системе, описывающей передачу запросов по линиям связи между n-м узлом-источником и удаленным узлом, содержащим необходимый файл.

Тогда по формуле (1) рассчитаем значения этих величин:

$$\rho_{n2} = \lambda_{n5} / \mu_{n2} = \lambda_{n2} P_{n5} / \mu_{n2} = \lambda_n P_{n2} P_{n5} / \mu_{n2} = \lambda P_n P_{n2} P_{n5} / \mu_{n2} \; , n = \overline{1, T};$$

$$L_{Sn2} = \rho_{n2} / (1 - \rho_{n2}), n = \overline{1, T}; \; W_{Sn2} = 1 / [\mu_{n2} (1 - \rho_{n2})], n = \overline{1, T}.$$

Рассчитаем характеристики качества для всей системы:

1) среднее число заявок на обслуживание, находящихся в системе:

$$L_s^{\text{сист}} = \frac{L_{s1}^{\text{ср.}} + L_{s2}^{\text{ср.}} + L_{s3}^{\text{ср.}}}{3},$$

где $L_s^{\text{сист}}$ – среднее число заявок на обслуживание, находящихся в рассматриваемой многофазной системе;

$L_{s1}^{\text{ср.}}$ можно вычислить по формуле:

$$L_{s1}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{sn1}}{T} = \frac{\sum_{n=1}^{K} \rho_{n1}/(1-\rho_{n1})}{T},$$

здесь $\rho_{n1} = \lambda_{n3}/\mu_{n1} = \lambda_{n1}P_{n3}/\mu_{n1} = \lambda_n P_{n1} P_{n3}/\mu_{n1} = \lambda P_n P_{n1} P_{n3}/\mu_{n1}$, $n = \overline{1,T};$

$L_{s2}^{\text{ср.}}$ вычисляем по формуле:

$$L_{s2}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{sn2}}{T} = \frac{\sum_{n=1}^{K} \rho_{n2}/(1-\rho_{n2})}{T},$$

где $\rho_{n2} = \lambda_{n5}/\mu_{n2} = \lambda_{n2}P_{n5}/\mu_{n2} = \lambda_n P_{n2} P_{n5}/\mu_{n2} = \lambda P_n P_{n2} P_{n5}/\mu_{n2}$, $n = \overline{1,T};$

$L_{s3}^{\text{ср.}}$ можно вычислить по формуле:

$$L_{s3}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{sn3}}{T} = \frac{\sum_{n=1}^{K} \rho_{n3}/(1-\rho_{n3})}{T},$$

где $\rho_{n3} = \lambda_{n7}/\mu_{n3} = \lambda_{n5}P_{n7}/\mu_{n3} = \lambda_{n2}P_{n5}P_{n7}/\mu_{n3} = \lambda_n P_{n2} P_{n5} P_{n7}/\mu_{n1}$, $n = \overline{1,T};$

2) средняя продолжительность пребывания заявки на обслуживание в системе может быть вычислена по формуле: $W_s^{\text{сист}} = L_s^{\text{сист}}/\lambda,$

3) среднее число заявок на обслуживание, находящихся в очереди, можно вычислить по формуле:

$$L_q^{\text{сист}} = \frac{L_{q1}^{\text{ср.}} + L_{q2}^{\text{ср.}} + L_{q3}^{\text{ср.}}}{3},$$

где $L_{q1}^{\text{ср.}}$ вычисляем по формуле:

$$L_{q1}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{qn1}}{T} = \frac{\sum_{n=1}^{K} \rho_{n1}^{\,2}/(1-\rho_{n1})}{T};$$

$L_{q2}^{\text{ср.}}$ можно вычислить по формуле:

$$L_{q2}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{qn2}}{T} = \frac{\sum_{n=1}^{K} \rho_{n2}^{\,2}/(1-\rho_{n2})}{T};$$

$L_{q3}^{\text{ср.}}$ может быть вычислено по формуле:

$$L_{q3}^{\text{ср.}} = \frac{\sum_{n=1}^{K} L_{qn3}}{T} = \frac{\sum_{n=1}^{K} \rho_{n3}^{\,2}/(1-\rho_{n3})}{T};$$

4) средняя продолжительность пребывания заявки на обслуживание в очереди может быть вычислена по формуле: $W_q^{сист} = L_q^{сист} / \lambda$.

### 3. Разработка имитационной модели

Имитационные модели предполагают описание объекта моделирования в виде специального алгоритма, воспроизводящего процесс взаимодействия элементов объекта во времени, а решение (результаты проведения экспериментов) определяется на некотором интервале с заданным шагом. Таким образом, имитационную модель РБД с обобщенной структурой целесообразно представить в виде операторной схемы (рис. 2).

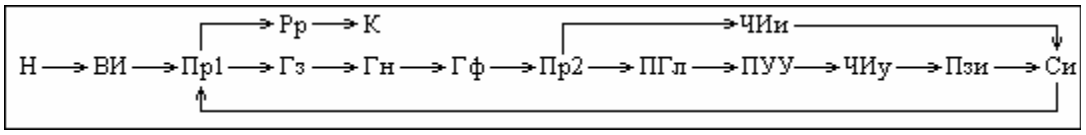



Рис. 2. Имитационная модель РБД с обобщенной структурой в виде операторной схемы

Описание операторов: Н – начало; ВИ – задание исходных данных (структуры РБД, частот поступления запросов, характеристик оборудования и линий связи, времени моделирования ); Пр1 – оператор условия. Если время завершения обслуживания i-го запроса меньше заданного времени моделирования, то переход к оператору Гз, если нет – то к Рр; Гз – генерация момента времени возникновения очередного запроса; Гн – генерация номера узла-источника; Гф – генерация файла, необходимого в рамках данного запроса; Пр2 – оператор условия. Если необходимый файл находится в узле-источнике, то переход к оператору Чии, если нет – то к оператору ПГл; ПГл – процесс передачи i-го запроса к узлу, содержащему глобальный справочник; ПУУ – процесс передачи i-го запроса из узла с главным справочником в удаленный узел, содержащий необходимый файл; ЧИу – процедура чтения/изменения необходимого файла в удаленном узле; Пзи – процесс передачи i-го запроса из удаленного узла в узел-источник; Си – сохранение информации о запросе, необходимой для расчета итоговых результатов (время нахождения запроса в очередях и время ее обслуживания, информация о возникших отказах оборудования и линий связи); ЧИи – процедура чтения/изменения необходимого файла в узле-источнике; Рр – расчет итоговых результатов моделирования (среднее число заявок на обслуживание, находящихся в системе; среднее число заявок на обслуживание, находящихся в очереди; средняя продолжительность пребывания заявки на обслуживание в системе; средняя продолжительность пребывания заявки на обслуживание в очереди).

### 4. Разработка программного средства моделирования

При выборе языка программирования следует руководствоваться его приспособленностью для описания моделей выбранного класса и доступностью. В данном проекте будет использоваться универсальный язык высокого уровня C#, являющийся частью Microsoft Visual Studio 2005, так как он позволяет создать пользовательский графический интерфейс, обладает большим числом подключаемых классов, упрощающих написание программы.

Для проведения экспериментов и сравнения двух моделей было разработано специальное программное средство RBD_MOD, которое осуществляет моделирование РБД аналитическими и имитационными методами, используя разработанные модели.

### 5. Экспериментальная проверка

Эксперименты проводились на персональном компьютере с тактовой частотой 3ГГц и объемом оперативной памяти 2 Гбайт. Для проведения экспериментов использовалась РБД с произвольной топологией, изображенная на рис. 3.

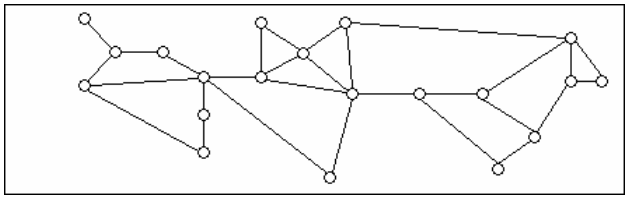


Рис. 3. РБД с произвольной топологией

Эксперимент 1. Цель эксперимента – сравнение зависимостей машинного времени, необходимого на проведение моделирования, от частоты поступления запросов пользователей при использовании имитационного и аналитического моделирования.

Исходные данные к эксперименту:

– быстродействие оборудования: 5 запросов в минуту (предполагается, что все запросы одного объема и характеристики оборудования узлов одинаковы);

– вероятность отказа каждого из узлов равна 0,0005;

– вероятность отказа каждой из линий связи равна 0,001.

Для имитационной модели вычислялось среднее машинное время, необходимое для моделирования, рассчитанное по результатам 10-ти экспериментов.

Результаты проведения 1-го эксперимента представлены в табл. 1.

Таблица 1

| Частота появления запросов(з/мин) | 1 | 5 | 20 | 50 | 100 | 150 | 250 | 350 | 500 |
|---|---|---|---|---|---|---|---|---|---|
| Затраты машинного времени (аналитическая модель) (с) | 0,5 | 1,4 | 2,2 | 3,1 | 4,8 | 5,9 | 7,3 | 8,5 | 9,8 |
| Затраты машинного времени (имитационная модель) (с) | 1 | 2,5 | 4,1 | 11,1 | 29,6 | 48,4 | 89,2 | 140,5 | 290,6 |

Графики зависимостей машинного времени, необходимого на моделирование, от частоты поступления запросов пользователей изображены на рис. 4.

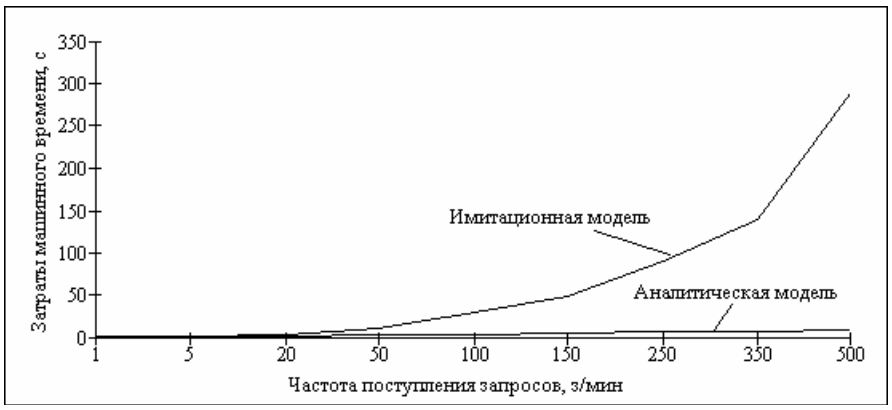



Рис. 4. Графики зависимости машинного времени, необходимого для моделирования, от частоты поступления запросов пользователей при использовании аналитического и имитационного моделирования

Из графиков, представленных на рис. 4, следует, что затраты машинного времени при использовании имитационной модели экспоненциально зависят от частоты поступления запросов, тогда как при использовании аналитической модели эти затраты несущественно возрастают с увеличением частоты поступления запросов.

Эксперимент 2. Цель эксперимента – сравнение зависимостей объема оперативной памяти, необходимого на моделирование, от частоты поступления запросов пользователей при использовании имитационного и аналитического моделирования.

Исходные данные к эксперименту:

– быстродействие оборудования: 5 запросов в минуту (предполагается, что все запросы одного объема и характеристики оборудования узлов одинаковы);

– вероятность отказа каждого из узлов равна 0,0005;

– вероятность отказа каждой из линий связи равна 0,001.

Для имитационной модели вычислялся средний объем оперативной памяти, необходимый для моделирования, рассчитанный по результатам 10-ти экспериментов.

Результаты проведения 2-го эксперимента представлены в табл. 2.

Таблица 2

| Частота появления запросов(з/мин) | 1 | 5 | 20 | 50 | 100 | 150 | 250 | 350 | 500 |
|---|---|---|---|---|---|---|---|---|---|
| Объем оперативной памяти (аналитическая модель)(Mb) | 8 | 9,3 | 12,2 | 17,1 | 22,2 | 27,0 | 32,8 | 44,9 | 54,2 |
| Объем оперативной памяти (имитационная модель)(Mb) | 6,2 | 7,1 | 9,4 | 15,6 | 20,1 | 29,4 | 39,1 | 48,5 | 62,1 |

Графики зависимостей объема оперативной памяти, необходимого на моделирование, от частоты поступления запросов пользователей изображены на рис. 5.

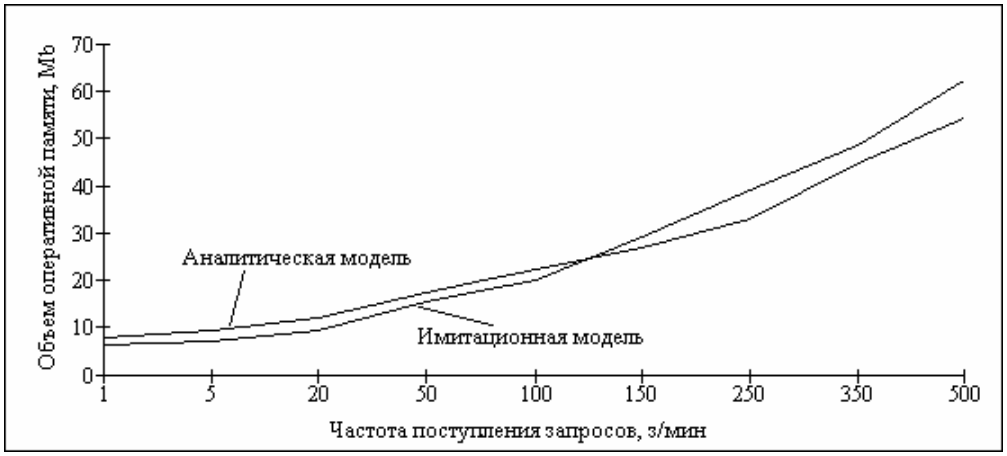



Рис. 5. Графики зависимостей объема оперативной памяти на моделирование от частоты поступления запросов пользователей при использовании аналитического и имитационного моделирования

Из графиков, изображенных на рис. 5, следует, что в случае моделирования РБД обобщенной структуры при равных частотах поступления заявок на всех узлах и при одинаковой производительности оборудования существует такая частота поступления запросов, при которой объем оперативной памяти, требуемый на проведение аналитического и имитационного моделирования, одинаков, а также существуют 2 промежутка значений частот, на которых более предпочтительной является либо имитационная, либо аналитическая модель.

**Выводы**

*Научная новизна* исследования состоит в том, что для моделирования распределенных баз данных с произвольной топологией предложены аналитическая и имитационная модели; разработаны алгоритмы реализации предложенных моделей, а также специальное программное средство RBD_MOD, реализующее предложенные модели. С помощью этого средства проведен ряд экспериментов, направленных на сравнение аналитических и имитационных методов моделирования РБД с точки зрения вычислительных и временных ресурсов, затрачиваемых на их реализацию, и ограничений каждого из методов.

*Практическая значимость* исследования состоит в том, что экспериментальным путем выяснено, что с точки зрения времени проведения моделирования аналитические методы работают быстрее, чем имитационные. При невысоких частотах поступлений запросов имитационные методы требуют меньший объем оперативной памяти, а при более высоких частотах меньше памяти требуют аналитические методы. Аналитические модели позволяют описывать работу лишь тех РБД, для параметров которых принимается ряд допущений (производительность аппаратуры в узлах РБД одинакова, частота поступления запросов на каждый узел – экспоненциально распределенная величина с одинаковыми параметрами для всех узлов). Имитационные же методы позволяют описать функционирование РБД любой сложности и с любыми параметрами.

**Список литературы: 1.** *Янбых Г.Ф., Столяров Б.А.* Оптимизация информационно-вычислительных сетей. М.: Радио и связь, 1987. **2.** *Цегелик Г.Г.* Системы распределенных баз данных. Львов: Свит, 1990.

168с. **3.** *Мищеряков Ю.В., Шовкопляс Ю.В., Евсеева Н.В.* Моделирование распределенных баз данных в условиях неопределенности // Вестник ХНТУ. 2007. №4(27). С. 240-247. **4.** *Марасанов В.В., Мамиконов А.Г., Кульба В.В., Коробко В.Б.* Модели синтеза систем баз данных в вычислительных сетях. Кишинев: «Штиинца», 1987. 268с. **5.** *Основы* моделирования сложных систем /Под общ. ред. И.В. Кузьмина. Киев: Вища шк., 1981. 360 с. **6.** *Гнеденко Б.В., Коваленко И.М.* Применение теории массового обслуживания к задачам больших систем. В кн.: Большие системы. Теория, методология моделирования. М.: Наука, 1971. **7.** *Мартин Дж.* Системный анализ передачи данных / Пер. с англ. М.: Мир, 1996. **8.** *Хаусли Т.* Системы передачи и телеобработки данных / Пер. с англ. М.: Радио и связь, 1994. **9.** *Советов Б.Я., Яковлев С.А.* Моделирование систем. М.: Высш. шк., 2001. 271 с.

*Поступила в редколлегию 18.09.2007*

**Евсеев Виктор Владимирович,** канд. техн. наук, профессор кафедры системотехники ХНУРЭ. Научные интересы: автоматизация проектирования сложных систем. Адрес: Украина, 61000, Харьков, пер. Хорошевский, 13, кв. 1, тел.: 372-56-16.

**Шовкопляс Юрий Витальевич,** студент факультета компьютерной инженерии и управления ХНУРЭ. Научные интересы: математическое моделирование. Адрес: Украина, 61000, Харьков, ул. Конева, 13, кв. 70, тел. 712-47-78.

**Самойленко Наталья Викторовна,** аспирантка кафедры системотехники ХНУРЭ. Научные интересы: информационные технологии моделирования и проектирования. Адрес: Украина, 61000, Харьков, ул. Полтавский шлях, 154, кв. 131, тел.: 372-56-16.

---

УДК 621.318.5:002

*Е.М. АНПИЛОГОВ, И.Е. АНПИЛОГОВА, Б.В. ДЗЮНДЗЮК, Л.И. МАРЧЕНКО, Л.В. ЛАРЧЕНКО*

# МОДЕЛЬ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКОЙ СИСТЕМОЙ В ЦЕЛЯХ ВЫБОРА ОПТИМАЛЬНЫХ ПАРАМЕТРОВ И УСТРАНЕНИЯ НЕЖЕЛАТЕЛЬНЫХ И ОПАСНЫХ ФАКТОРОВ

Моделируется технологический процесс при изготовлении системы с учётом входных, выходных параметров и режимов операций. Актуальность задачи заключается в том, что в зависимости от входных параметров и режимов отдельных операций можно прогнозировать выходные параметры готового изделия. На технологических операциях с помощью режимов устраняются нежелательные и опасные факторы.

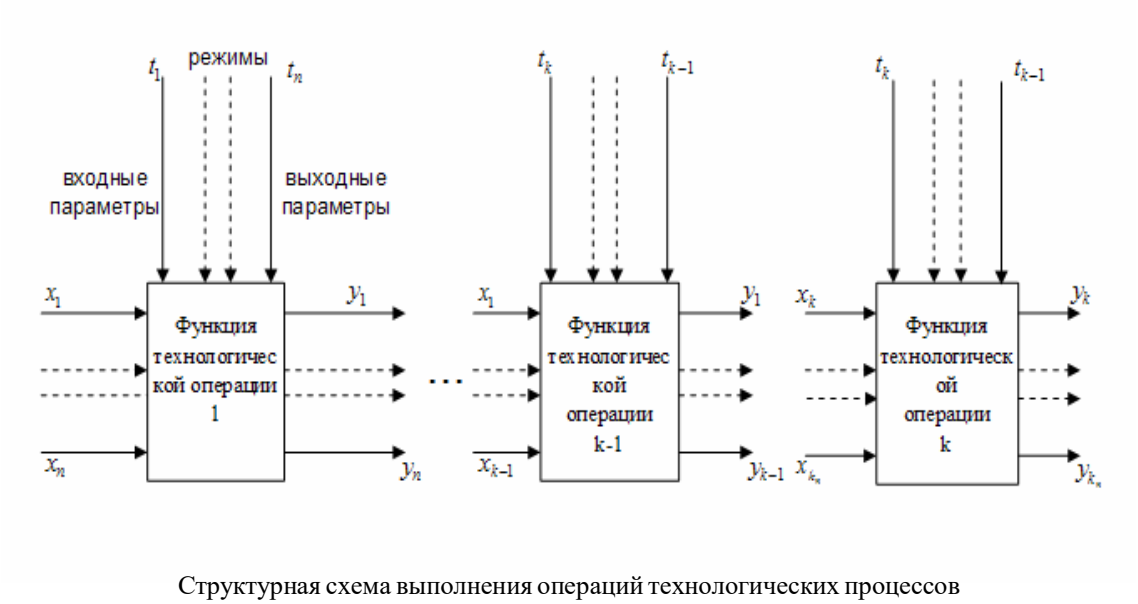
Научно-технический прогресс в интенсификации экономики предусматривает опережающее развитие новой техники и технологии. Увеличение производства сложных систем, насыщенности ими промышленности, транспорта и других отраслей народного хозяйства должны быть обеспечены условиями безопасной, безвредной и высокоэффективной работой персонала и оптимальной технологией.

Выполнение данных положений связано с расходом значительных средств, часто соизмеримых со стоимостью системы, поэтому правильное проектирование и эксплуатация сложных систем с точки зрения безопасности – очень важная задача производительности труда и дохода.

Изучение системы «Человек-Машина-Среда» может управлять, прогнозировать, а также учитывать состояние человека с учётом динамического характера реакций организма и специфических особенностей конкретного технологического процесса, управляемого оператором.

Данная работа посвящена рассмотрению системы, программы в целях учёта, управления и минимального влияния опасных и вредных факторов в процессе выбора рациональной технологии для изготовления системы.

Исследуемая система может быть представлена в виде совокупности технологических процессов (операций), выполняемых в определенной последовательности, как показано на рисунке. На каждую операцию действуют векторы – входные, выходные и режимов управления этими операциями. Все эти элементы объединяются в систему при помощи связей.




Структурная схема выполнения операций технологических процессов

Целью работы является оценки количественных критериев и нахождение оптимальных и граничных значений для управления ходом технологического процесса.

Для нахождения оптимальных и критических значений факторов воздействия на человека необходимо создание модели управления технологическим процессом, в котором на каждом этапе операции можно вмешиваться в ход процесса и своевременно компенсировать либо устранять нежелательные, а также опасные факторы.

Для учёта нежелательных последствий разработаем относительно вероятностную функцию $f_e\left(\overline{t_e}, y_{e-1}, \overline{y_e}\right)$, моделирующую е-ю операцию, и укажем способ её построения на основании (1), (2).

Считаем, что случайный вектор $\bar{y} = \{y_1,...,y_n\}$ распределен нормально и плотность его распределения выражается формулой:

$$f(y_1,...,y_n) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot |k|} \cdot \exp\left\{-\frac{1}{2}(\bar{y}-\overline{M})^* \cdot k^{-1} \cdot (\bar{y}-\overline{M})\right\}, \qquad (1)$$

где k – некоторая положительно определенная матрица; $|k|$ – детерминант матрицы k; $k^{-1}$ - матрица, обратная k; $\overline{M}$ – числовой вектор; $(\bar{y}-\overline{M})^*$ – транспонированный вектор $(\bar{y}-\overline{M})$ .

Обозначим через $MY_i(M_{y_i})$ и $DY_i(\sigma^2_{y_i})$ собственно математическое ожидание и дисперсию случайной величины $Y_i$.

Имеет место соотношение: $\sigma^2_{y_i} = DY_i = MY_i^2 - (MY_i)^2$.

Символ $k_{y_iy_j}$ будет обозначать ковариацию случайных величин $Y_i$ и $Y_j$:

$$K_{y_iy_j} = M \cdot \{(Y_i - MY_i) \cdot (Y_i - MY_i)\},$$

тогда $K_{y_iy_j} = DY_i$.

Для нормально распределенной совокупности $\bar{y}$ элементы вектора $\overline{M}$ и матрицы k, участвующие в формуле (1), имеют вполне определённый вероятностный смысл, а именно:

$$M_i = MY_i \;;\; K_{ij} = K_{y_iy_j} (i, j = 1,...,n).$$

Пусть $\{y_1,...y_n, x_1,...,x_k\}$ – некоторая совокупность случайных величин с плотностью совместного распределения $d(y_1,...,y_n,x_1,...,x_n)$.

Плотность распределения совокупности $\{x_1,...,x_n\}$ обозначим через $h(x_1,...,x_k)$.

Для нормально распределённой совокупности $\{y_1,...y_n, x_1,..., x_k\}$ ковариация $\frac{K_{y_i y_j}}{\bar{x}}$ не зависит явно от $\bar{x}$. При фиксированном векторе $\bar{x}$ плотность $f\left(\frac{\bar{y}}{\bar{x}}\right)$ задаёт нормальный закон распределения.

Построение переходной функции по экспериментальным данным.

Обозначим через $\bar{x} = \{x_1,..., x_k\}$ и $\bar{y} = \{y_1,..., y_n\}$ соответственно входящие и выходящие из операции $y$; через $\bar{t} = \{t_1,..., t_r\}$ - значения технологических факторов для данной операции.

Предположим, что совокупность $\{\bar{x}, \bar{y}\} = \{x_1,..., x_k; y_1,..., y_n\}$ при фиксированных значениях $\bar{t}$ по нормальному закону.

Переходная функция, моделирующая операцию $Y$.

Суть плотности распределения вектора $\bar{Y}$ при заданном значении вектора $\bar{X}$.

Обозначим переходную функцию через $f(t, \bar{x}, \bar{y})$ и укажем способ её построения по результатам эксперимента.

Фиксируем значения технологических факторов $\bar{t} = \{t_1,..., t_r\}$ и выполняем N раз операцию $Y$.

Значения величин $X_i, Y_j$ при e-м выполнении операции (эксперимента) обозначим через $X_{ie}, Y_{ie}$.

1) Вычислим $m_{x_i}, m_{y_j}; i = 1...k; j = 1...n$ - оценки математическим ожиданий (выборочные средние) величин $x_i, y_i$:

$$m_{x_i} = \frac{\sum_{e=1}^{N} x_{i,e}}{N}; \qquad m_{y_j} = \frac{\sum_{e=1}^{N} y_{i,e}}{N} \quad . \tag{2}$$

2) Вычислим элементы ковариационной матрицы k, определяющей нормальный закон распределения совокупности $\{\bar{x}, \bar{y}\}$.

Оценки ковариаций (выборочные ковариации) $K_{x_i x_j}, K_{x_i y_j}, K_{y_i y_j}$ обозначим через

$$\begin{aligned} K_{x_i x_j} &= \frac{\sum_{e=1}^{N} (x_{i,e} - m_{x_i})(x_{j,e} - m_{x,e})}{N-1}, \\ K_{x_i y_j} &= \frac{\sum_{e=1}^{N} (x_{i,e} - m_{x_i})(y_{j,e} - m_{y,e})}{N-1}, \\ K_{y_i y_j} &= \frac{\sum_{e=1}^{N} (y_{i,e} - m_{y_i})(y_{j,e} - m_{y,e})}{N-1}; \end{aligned} \tag{3}$$

Отсюда, в частности, получим $S^2_{x_i}, S^2_{y_i}$ - оценки для дисперсий (выборочные дисперсии) величин $X_i, Y_j$:

$$S^2_{x_i} = K_{x_i x_i} = \frac{\sum_{e=1}^{N}(x_{i,e} - m_{x_i})}{N-1}, \quad S^2_{y_i} = K_{y_i y_i} = \frac{\sum_{e=1}^{N}(y_{i,e} - m_{y_j})}{N-1}. \tag{4}$$

3) Перейдём от величин $X_i$ к центрированным $\widetilde{X}_i = X_i - m_{x_i}$ и применим алгоритм ортогонализации. В результате получим независимые случайные величины $\{Z_1,...,Z_k\}$:

$$Z_e = \widetilde{X}_e - \lambda_{e1}Z_1 - \lambda_{e2}Z_2 - ... - \lambda_{e-1}Z_{e-1},\ e = 1...k\ .$$

Коэффициенты $\lambda_{e_j},\ j = 1,...,e-1,$ полностью определяются выборочными ковариациями $K_{x_i x_j}$.

В процессе ортогонализации последовательно находим также выборочные дисперсии $S^2_{z_e}$ и выборочные ковариации $K_{y_i z_e}$. Отметим, что $m_{z_e} = \otimes$.

4) Пользуясь формулами (2),(3),(4), получим оценки для условных математических ожиданий $M_{\frac{y_i}{\overline{Z}}}$ и ковариаций $K_{\frac{y_i y_j}{\overline{Z}}}$:

$$\begin{cases} \beta_{ie} = \dfrac{K_{y_i z_e}}{S^2_{y_e}} \\ m_{\frac{y_i}{\overline{Z}}} = \sum_{e=1}^{k} \widetilde{\beta}_{ie} Z_e \\ K_{\frac{y_i y_j}{\overline{Z}}} = K_{y_i y_j} - \sum_{e=1}^{k} \widetilde{\beta}_{ie} \cdot \widetilde{\beta}_{je} \cdot S^2_{Z_e} \end{cases} \tag{5}$$

5) Составим матрицу $K = (K_{ij})$ и вектор $\overline{M} = \{M_1,...,M_k\}$, элементы $K_{ij}, M_i$ которых есть:

$$K_{ij} = \frac{K_{y_i y_j}}{\overline{Z}},$$

$$M_i = \frac{m_{y_i}}{\overline{Z}}.$$

Искомая переходная функция $f(\bar{t}, \bar{x}, \bar{y})$ даётся формулой:

$$f(\bar{t}, \bar{x}, \bar{y}) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot |K|} \cdot \exp\left\{-\frac{1}{2}(\bar{y} - \overline{M})^* \cdot k^{-1} \cdot (\bar{y} - \overline{M})\right\} \quad . \tag{6}$$

Определенная переходная функция зависит от вектора $\overline{Z} = (Z_1,...,Z_k)$, но поскольку величины $Z_i$ и $X_j$ связаны известной линейной зависимостью:

$$Z_i = C_{i1}\widetilde{X}_i + C_{i2}\widetilde{X}_2 + ... + C_{i,i-1}\widetilde{X}_{i-1} + \widetilde{X},\ \ i = 1,...,k\ , \tag{7}$$

то подставив в (7) выражение (6), получим в качестве аргументов переходной функции исходные величины $\{X_1,...,X_k\}$.

6) В пространстве $T = \{t_1,...,t_2\}$ выберем некоторую решетку $R_T$, целиком лежащую в области допустимых значений технологических факторов $D$. Для каждого узла решетки строим переходную функцию и на полученном семействе $f(\bar{t}, \bar{x}, \bar{y})$ реализуем алгоритмы оптимального управления технологическим процессом.

**Выводы**

Практическая значимость исследования состоит в том, что модель можно использована при проектировании технологических процессов и операций , а также прогнозировать эксплуатационные параметры изделия с учётом входных, выходных и режимных параметров. При этом на каждой операции при помощи режимов можно прогнозировать и устранять нежелательные и опасные факторы , что не учитывалось в работах [3-6].

*Научная новизна* работы состоит в управлении технологическими процессами как на уровне операций, так и на уровне общей технологической систем с прогнозированием эксплуатационных показателей, что позволяет существенно экономить время проектирования с использованием ЭВМ.

**Список литературы: 1.** *Белжман Р., Жиджи Э.* Динамическое программирование и современная теория управления. М.: МИР, 1984. 207с. **2.** *Меткин Н.П. , Щёголев В.А.* Математические основы технологической подготовки гибких производственных систем. М.: Издательство стандартов, 1985. 254с. **3.** *Невлюдов И.М., Анпилогов Е.М.* О моделировании технологического процесса с учётом явлений технологической наследственности // АСУ и приборы автоматики.1984. Вып. 72. С. 89-93. **4.** *Дзюндзюк Б.В., Анпилогов Е.М., Анпилогова И.Е., Кравченко Ю.В.* Модель управления различными наследуемыми факторами в системе «Человек-Машина» // АСУ и приборы автоматики. 2005. Вып. 132. с. 110-114. **5.** *Дальский А.М.* Технологическое обеспечение надёжности высокоточных деталей машин. М.: Машиностроение, 1975. 225с. **6.** *Ящерицин П.И., Рыжов Э.В., Аверченков В.И.* Технологическая наследственность в машиностроении. Минск: Наука и техника, 1977. 255с.

*Поступила в редколлегию 02.08.2007*

**Анпилогов Евгений Михайлович,** канд. техн. наук, доцент кафедры охраны труда ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-60.

**Анпилогова Ирина Евгеньевна,** инженер каф. охраны труда ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-60.

**Дзюндзюк Борис Васильевич,** зав. каф. охраны труда ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-60.

**Марченко Людмила Ивановна,** ст. пр.каф. охраны труда ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-60.

**Ларченко Лина Викторовна,** доцент каф. охраны труда ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-13-60.

---

УДК 681.586.37:004.5

*Н.Я. КАКУРИН, Ю.В. ЛОПУХИН, Н.Н. БЫКОВА*

## ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АНАЛИЗА ПРЕОБРАЗОВАНИЙ ЧИСЕЛ

Рассматривается структура и возможности программного средства для анализа числа тактов преобразования и статистики преобразований преобразователей кодов, функционирующих по методу накопления эквивалентов

### 1.Постановка задачи

Основными параметрами преобразователей кодов по методу накопления являются быстродействие (число тактов преобразования) и аппаратурные затраты (число корпусов ИС или число вентилей). К достоинству преобразователей кодов этого типа относятся возможность изменения соотношения между быстродействием и аппаратурными затратами за счет выбора числа шагов преобразования, значений шагов и стратегии использования различных шагов преобразования.

Стратегия использования шагов преобразования может быть последовательной или параллельной.

При последовательной стратегии показания разрядных счетчиков, значения которых равны или превышают значение шага, уменьшают на значение этого шага.

Если же во всех преобразуемых разрядах значения цифр оказываются меньше этого шага, происходит переход на более меньший шаг, и далее ведется уменьшение значений

разрядов на значение меньшего шага и т. д., пока не будут обнулены все разрядные счетчики.

Последовательная стратегия по сравнению с параллельной структурно реализуется достаточно просто. Аппаратурные затраты на построение основного блока ПК – формирователь элементов – будут наибольшими.

Для оценки сверху числа тактов преобразования целых чисел с последовательной стратегией в многошаговых ПК выполняют по формулам:

$$N_1^{цел} = K\text{-}1;$$

$$N_2^{цел} = ](K\text{-}1)/a[+a\text{-}1; \qquad (1)$$

$$N_3^{цел} = ](K\text{-}1)/b[+](b\text{-}1)/a[+a\text{-}1;$$

$$N_4^{цел} = ](K\text{-}1)/c[+](c\text{-}1)/b[+](b\text{-}1)/a[+a\text{-}1,$$

где К – основание системы счисления на входе; a, b, c – соответственно второй, третий и четвертый шаги преобразования (первый шаг всегда равен 1); $N_1^{цел}$, $N_2^{цел}$, $N_3^{цел}$, $N_4^{цел}$ – соответственно максимальное число тактов преобразования одно-; двух-; трех-; четырехшагового ПК целых чисел.

Для значений шагов преобразования должны выполняться следующие ограничения:

$$1< a< b;\ a < b < c;\ b < c \leq K\text{-}1.$$

Недостатком использования максимальных значений $N_i^{цел}$ ($i = \overline{1,4}$) для оценки быстродействия являются завышенные требования, которые для определенных подмножеств преобразуемых чисел будут значительны и потребуют применения в структуре ПК более быстродействующих ИС.

В целях реальной оценки быстродействия разработан программный пакет ”CONVERTER”, позволяющий промоделировать процесс преобразования чисел для последовательной стратегии и определить конкретное число тактов преобразования, а не оценку сверху.

**2. Назначение и структура программного средства**

К назначениям программного средства относятся: возможность имитации работы преобразователя кодов по методу накопления эквивалентов;

возможность работы в двух режимах: режиме преобразования целых и дробных чисел, режиме статистики преобразования целых и дробных чисел.

При разработке данного программного продукта было выбрана наиболее оптимальная по скорости проектирования и эффективности визуальная среда программирования Borland Delphi 7.0 для Windows 2000/XP.

Для нормальной работы данной программы необходимы следующие технические средства: IBM-совместимый компьютер с процессором не менее Pentium 233 МГц, монитор типа VGA, SVGA, ОЗУ 16 Мб, мышь, дисковод FDD (для установки программы), HDD (необходимая емкость свободного пространства определяется исходя из сложности решаемой задачи, но составляет не менее 1,5 Мб). Используемыми программными средствами являются: ОС Windows 95 и выше. Для более корректной работы программы рекомендуется ОС Windows 2000.

Разработанное программное обеспечение сформировано как независимый модуль и для своего функционирования не требует дополнительного программного обеспечения.

**3.Функционирование программы в режиме преобразования чисел**

На рис. 1 представлена структурная схема алгоритма режима преобразования чисел.

Этот режим используется для преобразования числа из К-ичной системы счисления в двоичную. Программа выдает не только окончательный результат преобразования, но и промежуточные результаты в виде таблицы: состояние счетчика, величину эквивалента и состояние накапливающего сумматора на каждом такте преобразования. Имеется возможность задавать различные число шагов преобразования от 1 до 8, основание системы счисления на входе преобразователя – от 3 до 15, разрядность числа – от 2 до 12, веса шагов преобразователя.

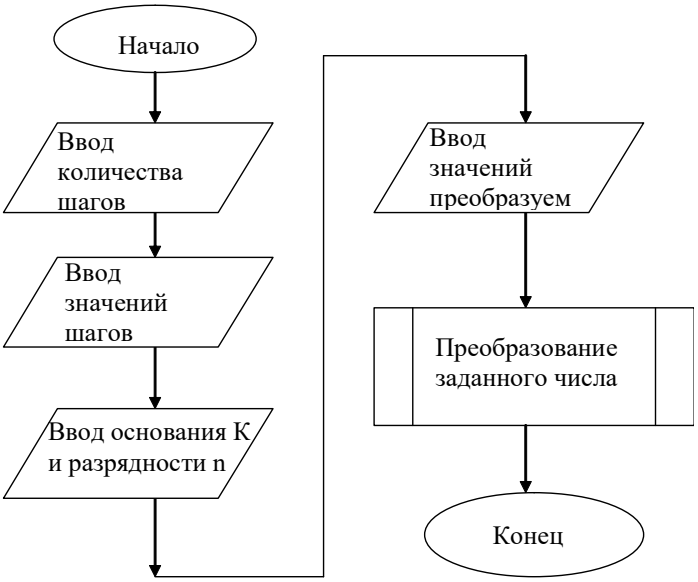



Рис. 1

Для того чтобы произвести расчет в данном режиме, необходимо запустить программу путем активизации исполняемого модуля Converter.exe.

После этого следует с помощью мышки активизировать главное меню программы или нажать клавишу Alt на клавиатуре и перейти к пункту Режим. В раскрывающемся меню следует выбрать пункт Режим преобразования. Для запуска данного режима работы программы имеется специальная горячая клавиша – F1. Нажав ее сразу после запуска программы, можно без перемещения по пунктам меню прийти к рассмотренному выше результату. После данных манипуляций на экране отобразиться окно для ввода исходных данных (рис. 2 ).

В этом окне имеются пять текстовых полей для ввода данных, три из которых снабжены кнопками инкремента. С помощью этих элементов управления необходимо задать исходные данные для режима преобразования чисел: количество шагов преобразователя, основание системы счисления на входе, разрядность преобразуемого числа, собственно само число, которое нужно преобразовать, и значения весов преобразователя. Программа содержит ряд проверок, которые предотвращают ввод некорректных данных. Если же пользователь по каким-либо причинам желает отказаться от расчета, то следует нажать кнопку Отмена. После этого программа, не сохраняя данные, вернется в исходное состояние.

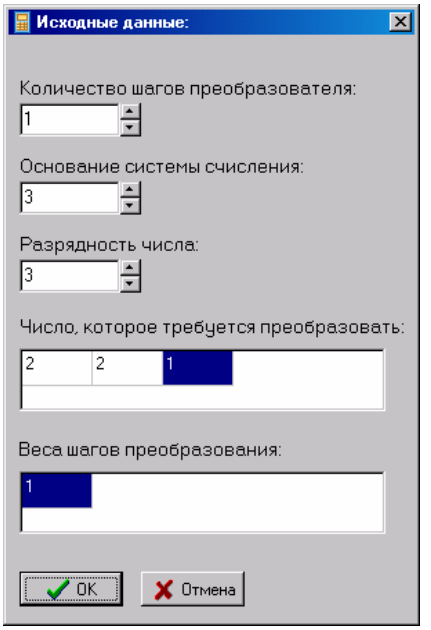



Рис. 2

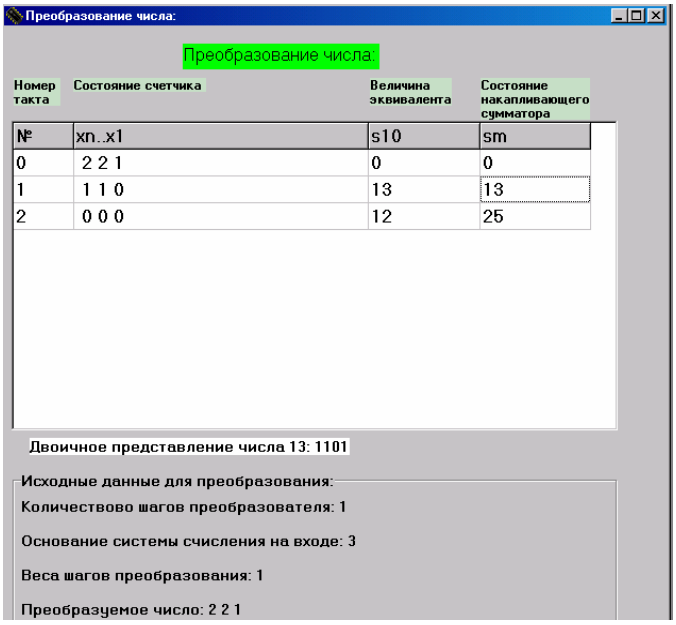



Рис. 3

Когда все значения будут введены в окно, следует нажать кнопку ОК для запуска преобразователя. После того, как программа произведет расчет, на экране высветится

окошко с результатами преобразования введенного числа (рис. 3). Данное окно выполнено с использованием современного многодокументного подхода к проектированию приложений на платформе Windows и является дочерним по отношению к главному окну программы. Это дает немалые преимущества для пользователя – имеется возможность сразу одновременно видеть на экране несколько результатов расчета в разных режимах работы приложения, сравнивать один с другим, упорядочивать окна, каскадировать их и т.д.

Программа отображает значения эквивалентов и состояния накапливающего сумматора в десятичном виде. Для того чтобы увидеть двоичные эквиваленты, необходимо щелкнуть левой кнопкой мыши на интересующем нас значении накапливающего сумматора, после чего в окошке с результатами отобразится двоичное представление данного числа. Тот же самый результат можно получить, если перемещаться по таблице клавишами клавиатуры с обозначениями стрелок.

Для того чтобы закрыть данное окошко, необходимо нажать кнопку с обозначением крестика.

**4. Функционирование программы в режиме статистики**

Режим статистики служит для отображения результатов преобразования заданного множества чисел в виде графической зависимости, которая позволяет определить количество преобразованных чисел за определенное число тактов преобразования. Программа отображает результаты в виде гистограммы. Кроме того, предусмотрена возможность получения статистики в виде таблицы.

Математическое ожидание рассчитывается по формуле: $M = (\sum_{i=0}^{i=N_{max}} i \cdot N_i) / 2^n$, где M — математическое ожидание числа тактов преобразования $i$; $N_i$ — количество чисел, преобразуемых за $i$ тактов; $N_{max}$ — максимальное значение числа тактов преобразования (зависит от основания системы счисления K и значений шагов преобразования a, b, c); n — число входных преобразуемых разрядов.

Затем рассчитывается дисперсия преобразования по формуле:

$$D = (\sum_{j=1}^{j=2^n} (M - N_j)^2) / 2^n \cdot (2^{n-1}),$$

$N_j$ – количество тактов, за которое преобразуется число $a_j$.

Результат записывается в переменную D. Для вывода на экран значений переменных a и D используется функция FormatFloat, которая форматирует их до двух знаков после запятой.

Имеется возможность задавать различные число шагов преобразования - от 1 до 8, основание системы счисления на входе преобразователя – от 3 до 15, разрядность чисел – от 2 до 12, веса шагов преобразователя. Кроме этого, данный режим позволяет вручную задавать набор преобразуемых чисел или автоматически на выбор. При автоматическом формировании чисел программа делает полный перебор всех возможных значений в диапазоне от 0 до n-1, где n - разрядность преобразуемых чисел.

Данный режим запускается как и предыдущий путем выбора пункта Режим статистики из меню Режим или нажатием горячей клавиши F2 на клавиатуре.

Сначала выбирается максимальный шаг преобразования и соответственно ему присваивается значение ноль. В переменную step заносится значение максимального шага из массива значений шагов. Переменная equiv хранит состояние накапливающего сумматора. Она инициализируется нулем на начальном этапе преобразования. После этого инициализируется нулевая строка таблицы. Первая, третья и четвертая строка заполняется нулями, так как на нулевом такте преобразования все счетчики обнуляются, а во вторую заносится исходное число.

Далее в цикле while выполняется преобразование числа:

```
while(true) do
begin
// подбираем максимально возможный шаг преобразования
if w_index<=l_kol_steps-2 then
 begin
 while(true) do
 begin
 for i:=0 to l_n-1 do
 if l_counter[i]-step>=0 then goto l1;
 w_index:=w_index+1;
 step:=l_weights[w_index];
 end;
 end;
l1:
sum:=0;
j:=l_n-1;
for i:=0 to l_n-1 do
 begin
 if l_counter[i]-step>=0 then
 begin
 l_counter[i]:=l_counter[i]-step;
  sum:=sum+step*power(l_k,j);
 end;
 j:=j-1;
 end;
equiv:=equiv+sum;
Child.StringGrid1.RowCount:=Child.StringGrid1.RowCount+1;
Child.StringGrid1.Cells[0,number+1]:=IntToStr(number);
for i:=0 to l_n-1 do
Child.StringGrid1.Cells[1,number+1]:=Child.StringGrid1.Cells[1,number+1]+' '+IntToStr(l_counter[i]);
Child.StringGrid1.Cells[2,number+1]:=FloatToStr(sum);
Child.StringGrid1.Cells[3,number+1]:=FloatToStr(equiv);
number:=number+1;
// условие завершения цикла
for i:=0 to l_n-1 do
 if l_counter[i]>0 then goto l2;
break;
l2:
end.
```

Структурная схема алгоритма для режима статистики представлена на рис. 4.

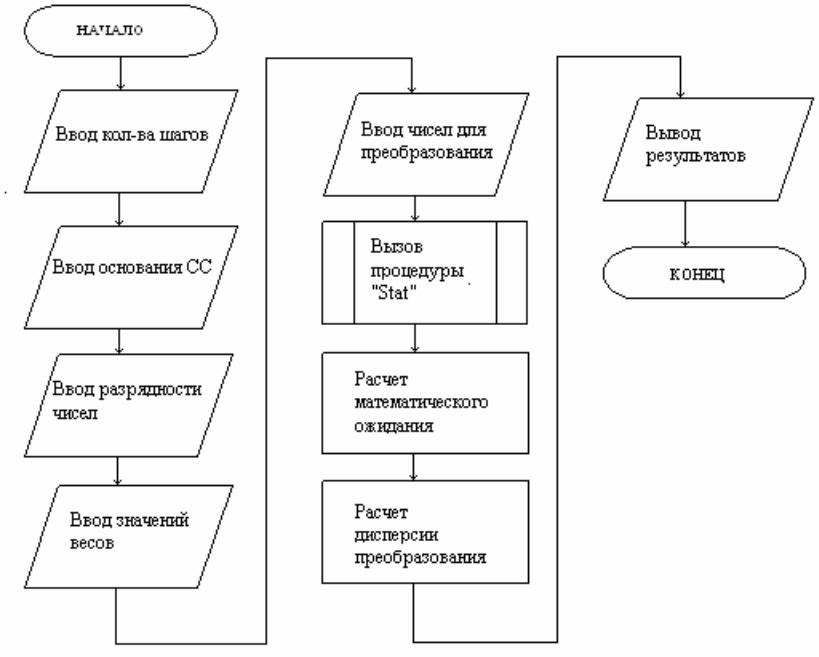



Рис. 4

После этого на экране монитора появится окно для ввода исходных данных в режиме статистики (рис.5).

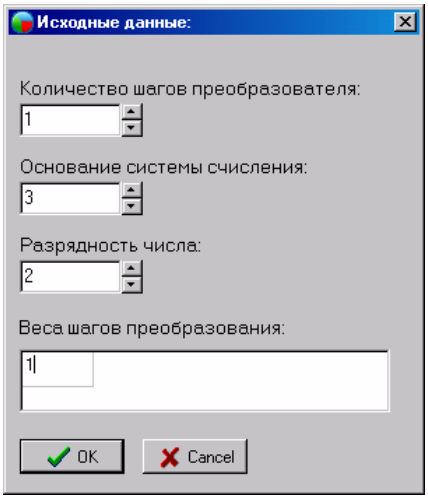



Рис. 5

В этом окне имеются четыре текстовых поля для ввода данных, три из которых снабжены кнопками инкремента. С помощью этих элементов управления необходимо задать исходные данные для режима статистики: количество шагов преобразователя, основание системы счисления на входе, разрядность чисел и значения весов преобразователя. Программа содержит ряд проверок, которые предотвращают ввод некорректных данных. Если же пользователь по каким-либо причинам желает отказаться от расчета, то следует нажать кнопку Отмена, после чего программа, не сохраняя данные, вернется в исходное состояние.

После того, как пользователь введет все необходимые данные на этом этапе работы программы, следует нажать кнопку ОК. После этого приложение выведет на экран еще одно окошко, в котором пользователь сможет задать необходимый список чисел для преобразования (рис. 6).

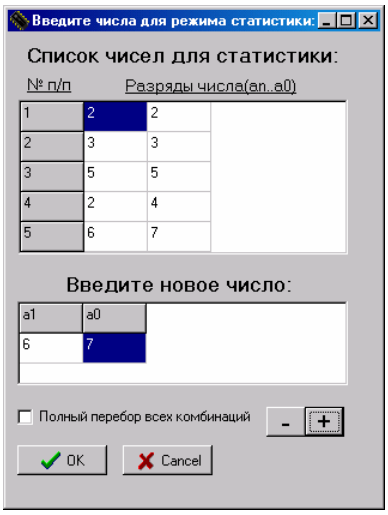



Рис. 6

После ввода всех данных программа в режиме статистики осуществляет полный перебор всех чисел и вывод результатов расчета как в виде гистограммы, так и в виде чисел в специальных окнах. Например, для К= 10; n = 3 и одного шага преобразования получены следующие расчетные данные количеств чисел $N_i$ за $i$ тактов: за 0 тактов – 1 число; за 1 – 7 чисел; за 2 – 19; за 3 – 37 чисел; за 4 –61 число; за 5 – 91 число; за 6 – 127 чисел; за 7 – 169 чисел; за 8 – 217 чисел; за 9 – 271 число. Математическое ожидание числа тактов преобразования M = 6,97, а дисперсия D = 3,71.

В этом окошке (см. рис.6) внизу имеется таблица для ввода чисел. Чтобы добавить число в список, пользователь должен ввести его в данную таблицу и нажать кнопку с изображением знака “+”. После этого введенное значение появится в верхней таблице, которая содержит все заданные числа на текущий момент. Если возникнет необходимость удалить последнее значение из списка, то необходимо нажать кнопку с изображением знака “-”.

Если нет необходимости проводить полную статистику преобразования по всему диапазону чисел, то следует дезактивизировать опцию «Полный перебор всех комбинаций».

Когда все значения будут введены в окно, следует нажать кнопку ОК для запуска преобразователя. После того, как программа произведет расчет, на экране высветится окошко с результатами статистики преобразования чисел (рис.7, а, б).

В этом окошке появится график в виде гистограммы, который и отображает зависимость количества преобразованных чисел от числа тактов преобразования.

## Выводы

*Научная новизна* выполненного исследования заключается в следующем:

1. Разработано программное обеспечение для получения достоверных оценок числа тактов преобразований как для ограниченного подмножества заданных чисел, так и в случае полного перебора всех чисел заданной разрядности.

2. Рассмотрена структура программного пакета и его функционирование в двух основных режимах: преобразование числа и получение статистических оценок результатов преобразования.


а


б

Рис. 7

3. Результаты апробации программного пакета показали, что требования по быстродействию к преобразователям кодов могут быть существенно снижены (на 25-30%).

Практическая значимость исследования состоит в возможности использования программного пакета для проектирования преобразователей кодов с улучшенными характеристиками.

**Список литературы: 1.** А.с. 1126946 5G06F 5/02. Преобразователь двоично-К-ичного кода в двоичный код /А.Н. Слобожанин // Открытия, изобретения. 1984. №44. С.250. **2.** А.с. 1647908 5НОЗМ 7/12. Преобразователь двоично-К-ичного кода в двоичный код /Н.Я.Какурин, Ю.К. Кирьяков, А.Н. Макаренко // Там же. 1991. №17. С. 262-263. **3.** *Какурин Н.Я.* Выбор величин шага преобразования в преобразователях код-код / /АСУ и приборы автоматики. 1991. Вып.97. С. 14-17. **4.** *Голян В.В., Какурин Н.Я., Макаренко А.Н., Замалеев Ю.С., Николаев А.А.* Двухкритериальный системный синтез многоблочных преобразователей кодов по методу накопления эквивалентов // АСУ и приборы автоматики. 2005. Вып. 133. С.102-107.

*Поступила в редколлегию 17.09.2007*

**Какурин Николай Яковлевич**, канд. техн. наук, профессор кафедры автоматизации проектирования вычислительной техники ХНУРЭ. Научные интересы: прикладная теория цифровых автоматов, автоматизация проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

**Лопухин Юрий Владимирович,** ст. преподаватель кафедры АПВТ ХНУРЭ. Научные интересы: проектирование программного обеспечения, автоматизации проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

**Быкова Надежда Николаевна,** студентка группы КСМ 04-2 факультета компьютерной инженерии и управления ХНУРЭ. Научные интересы: прикладная теория цифровых автоматов, автоматизация проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

УДК 681.325.53: 37 : 004.5

*С.Н. КОВАЛЕНКО*

# ДВУХШАГОВЫЙ ПРЕОБРАЗОВАТЕЛЬ КОДОВ С ПАРАЛЛЕЛЬНЫМ ИСПОЛЬЗОВАНИЕМ ШАГОВ ПРЕОБРАЗОВАНИЯ

Предлагается способ повышения быстродействия преобразователей кодов на счетчиках за счет применения параллельной стратегии использования шагов преобразования. Описывается метод расчета числа тактов преобразования и программное обеспечение для его реализации.

## 1.Постановка задачи

Двухшаговый преобразователь кодов (ПК) с последовательной стратегией использования шагов преобразования в ряде случаев не обеспечивает требуемого быстродействия. Поэтому в случае, когда необходимо высокое быстродействие, применяют параллельную стратегию шагов преобразования. Структуру преобразователя кодов в этом случае необходимо изменить.

## 2. Структура и функционирование двухшагового ПК последовательного типа

Структура двухшагового ПК, ориентированного на повышение быстродействия с применением параллельной стратегии приведена на рис.1.

Двухшаговый ПК двоично-К-ичного кода в двоичный код содержит группу разрядных счетчиков I, блок управления (генератор) импульсов 2, содержащий прямой П, прямые задержанные П31 и П32 выходы, первую группу триггеров 3 состояния, вторую группу триггеров 4 состояния, комбинационный двоичный сумматор 5 и регистр 6 результата, образующие в совокупности накапливающий сумматор 7, группу элементов И-НЕ 8, группу элементов И-НЕ 9, группу дешифраторов нуля 10, группу дешифраторов превышения 11, группу шифраторов 12, формирователь эквивалента 13, включающий в свой состав первый 14 и второй 15 дешифраторы и шифратор 16, элемент ИЛИ-НЕ 17, группу элементов ИЛИ 18.

В состав формирователя эквивалентов 13 также входят элемент НЕ 19, группа элементов И 20 и группа элементов ИЛИ 21. Функционирование двухшагового ПК происходит следующим образом.

Группа триггеров 3 фиксирует ненулевое значение счетчиков соответствующих разрядов; группа триггеров 4 - значение старших разрядных счетчиков, превышающее заданное число а-1, например 1.

Шифраторы 12 реализуют следующую функцию:

$$Y = \{X - a; X \geq a; \{X; X < a;$$

где X – входной код тетрады; а - параметр, в частном случае, равный 2. Цепи инициирования и сброса на рис.1 не показаны. Так как в конкретном случае n = 4, *К* = 12, то диапазон изменения входного кода $0 - (12^4 - 1) = 0 - 20735_{10}$. Код состояний триггеров как первой группы 3, так и второй 4 имеет $2^3 = 8$ значений от 000 до III.

Формирователь эквивалентов (ФЭ) 13, выполненный в виде последовательного соединения первого 14 и второго 15 дешифраторов, реализует функцию

$$S = \begin{cases} KC_1 + K^2C_2 + K^3C_3; Д_1 = Д_2 = Д_3 = 0; \\ aKK_1 + aK^2Д_2 + aK^3Д_3; Д_1, Д_2, Д_3 \neq 0; \end{cases}$$

где $C_1, C_2, C_3$ – значения разрядов двоичного кода триггеров состояний первой группы 3; $Д_1, Д_2, Д_3$ – значения разрядов двоичного кода триггеров состояния второй группы 4.

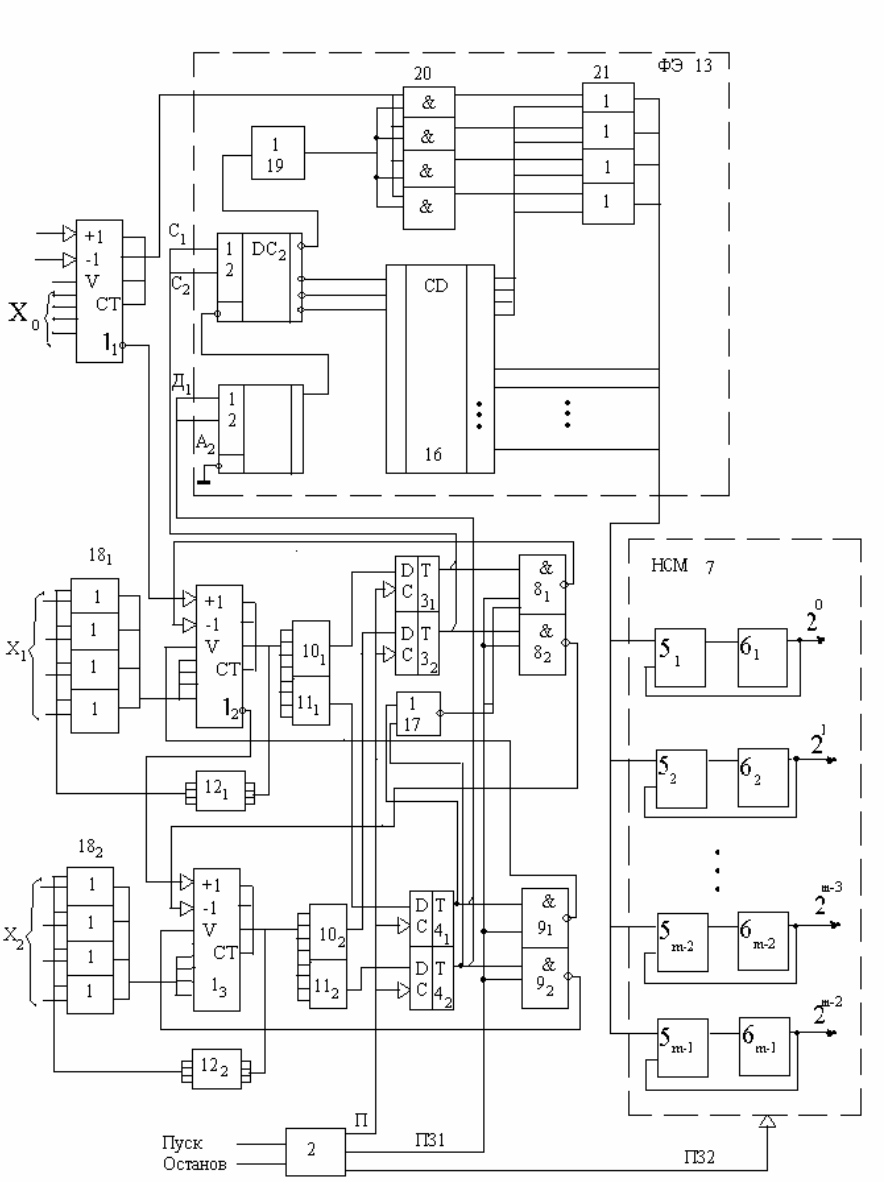



Рис 1. Структура двухшагового ПК с последовательным использованием шагов

Каждый из $C_m (m = \overline{1,3})$ триггеров 3 первой группы состояний разрядных счетчиков соответствует наличию (1) или отсутствию (0) информации в соответствующем старшем разряде преобразуемого кода, а разряд $Д_m (m = \overline{1,3})$ триггеров 4 второй группы состояний разрядных счетчиков равен 1, если соответствующий разряд преобразуемого кода имеет значение $x_i \geq a (2 \leq a \leq k-1)$, в противном случае $Д_m = 0$.

ФЭ 13 преобразует вначале двоичный код $Д_3, Д_2, Д_1$ триггеров состояний второй группы, затем при $Д_3, Д_2, Д_1 = 0$ преобразует двоичный код $C_3, C_2, C_1$ триггеров состояний первой группы и при $Д_3 = Д_2 = Д_1 = C_3 = C_2 = C_1$=000 выполняет трансляцию (передачу) двоично-К-ичного кода младшего разряда в двоичный код эквивалента на выходе.

**3. Структура и функционирование двухшагового ПК параллельного типа**

Логика управления в двухшаговом ПК параллельного типа выполнена так, чтобы запретить возможность вычитания шага 1, если в этом разряде имеется возможность вычитания шага а. И наоборот, если значение разряда $x_m$ находится в пределах $1 \leq x_m < a$, то

следует разрешить опрос вентиля, управляющего вычитанием I из разрядного счетчика, хранящего $x_m$.

Этот принцип управления реализуется в двухшаговом ПК параллельного типа путем замены (n-1) входного элемента ИЛИ-НЕ блоком инверторов, вход каждого из которых связан с единичным выходом соответствующего триггера старшего регистра состояний, а выход инвертора соединяется с управляющим входом схемы И, на информационный вход которой поступает сигнал с единичного выхода триггера этого разряда, но младшего регистра состоянии.

Закон функционирования ФЭ для двухшагового ПК параллельного типа для набора шагов 1,2 и К=12 приведен в таблице. ФЭ двухшагового ПК параллельного типа имеет 27 строк; ПК последовательного типа - всего 15 строк. Особенностью ФЭ для ПК параллельного типа является отказ от принципа последовательного соединения ряда стробируемых ДШ и применение ряда вентилей, каждый из которых стробирует определенный ДШ, выделяющий определенное подмножество входных наборов.

Дешифрование строк таблицы ФЭ ПК параллельного типа осуществляется не двумя последовательно включенными ДШ, а совокупностью ДШ, стробируемых состояниями триггеров $Д_3Д_2Д_1$ старшей группы, на информационные входы которых поступают сигналы с триггеров младшей группы. Каждой горизонтальной полосе соответствует определенный ДШ. Так, для дешифрования строк 0-7 следует применить ДШ 3-8, стробируемый инверсным выходом трехвходового элемента ЗИЛИ-НЕ, на входы которого поступают сигналы $Д_3Д_2Д_1$, а на информационные входы ДШ – сигналы $C_3C_2C_1$ (см. рис. 2.).

Вторую, третью и пятую полосы таблицы (строки 8-11; 12-15 и 18-21 соответственно) выделяют с помощью ДШ 2-4, на два информационных входа которых подают соответственно сигналы $C_3C_2, C_3C_1, C_2C_1$.

Выделение оставшихся строк 16-17 (четвертая полоса); 22-23 (шестая); 24—25 (седьмая) и 26 (восьмая) выполняют с помощью или ЛЭ или ДШ 2-4.

После выделения всех строк закона функционирования ФЭ реализация выходных функций шифратора $y_{12} - y_1$ для двухшагового ПК параллельного типа выполняется аналогично случаю ПК последовательного типа. В целях более простой реализации осуществляют выделение ряда общих частей в некоторых подмножествах функций. Так, $y_1 = y_2 = 0$;

$$y_{12} = Д_3; y_4 = C_1; y_3 = (\overline{Z_1}|\overline{Z_3}|\overline{Z_5}|\overline{Z_7}|\overline{Z_{13}}|\overline{Z_{15}}|\overline{Z_{19}}|\overline{Z_{21}}) \cup Z_{25};$$

$$M_1 = \overline{Z_4}|\overline{Z_5}|\overline{Z_6}|\overline{Z_7}; M_2 = \overline{Z_{10}}|\overline{Z_{11}}|\overline{Z_{14}}|\overline{Z_{15}}|\overline{Z_{17}}; M_9 = \overline{Z_{20}}|\overline{Z_{21}}|\overline{Z_{23}}|\overline{Z_{24}}|\overline{Z_{25}}|\overline{Z_{26}};$$

$$M_3 = M_5 \cup M_6; M_4 = \overline{Z_6}|\overline{Z_7}|\overline{Z_{16}}|\overline{Z_{17}}; M_5 = \overline{Z_{20}}|\overline{Z_{21}}; M_6 = \overline{Z_{23}}|\overline{Z_{24}}|\overline{Z_{25}}|\overline{Z_{26}},$$

$$y_{11} = Д_3 \cup M_1 \cup M_2; y_{10} = M_1 \cup M_2 \cup M_3,$$

$$y_9 = (\overline{Z_{11}}|\overline{Z_{12}}|\overline{Z_{13}}|\overline{Z_{14}}|\overline{Z_{15}}|\overline{Z_{18}}|\overline{Z_{19}}|\overline{Z_{22}}) \cup M_4,$$

$$y_8 = (\overline{Z_2}|\overline{Z_3}|\overline{Z_4}|\overline{Z_5}|\overline{Z_9}|\overline{Z_{10}}|\overline{Z_{14}}|\overline{Z_{15}}) \cup (\overline{Z_{17}}|\overline{Z_{18}}|\overline{Z_{19}}|\overline{Z_{22}}|\overline{Z_{24}}|\overline{Z_{25}}|\overline{Z_{26}}),$$

$$y_7 = M_1 \cup M_2; y_6 = (\overline{Z_9}|\overline{Z_{11}}|\overline{Z_{12}}|\overline{Z_{13}}|\overline{Z_{14}}|\overline{Z_{15}}|\overline{Z_{16}}|\overline{Z_{17}}) \cup M_6,$$

$$y_5 = (\overline{Z_2}|\overline{Z_3}|\overline{Z_6}|\overline{Z_7}) \cup (\overline{Z_8}|\overline{Z_{10}}|\overline{Z_{16}}|\overline{Z_{17}}|\overline{Z_{20}}|\overline{Z_{21}}|\overline{Z_{22}}|\overline{Z_{26}}).$$

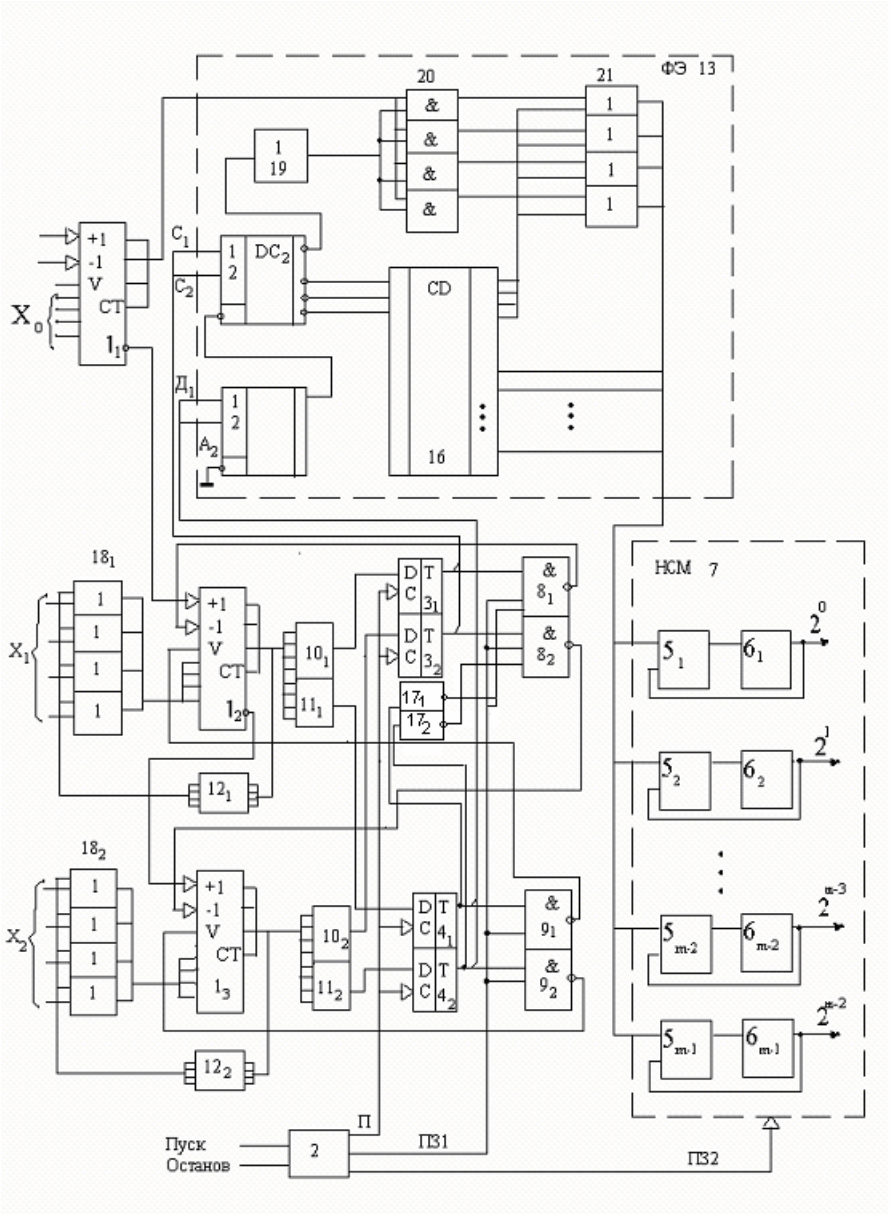



Рис 2. Структура двухшагового ПК с параллельным использованием шагов

**4. Математические модели, описывающие ФЭ в ПК параллельного типа**

Функционирование двухшагового многоблочного формирователя эквивалентов с параллельной стратегией использования шагов преобразования (рис.3), описывается выражением

$$S_m = \sum_{i=(m-1)P+1}^{i=mP} \gamma_i(h) K^{i-1} \cdot R_i(h), \; (i = \overline{1, mP}),$$

где m – номер блока $(m = \overline{1, M})$; i- номер разряда в блоке m; P – число разрядов в блоке $(P = \frac{n}{M})$; n – число входных преобразуемых разрядов; h– текущий шаг преобразований; M – число блоков в разбиении $(M = \frac{n}{p})$; $C_i$ – состояние i-го триггера регистра состояний.

Коэффициент $\gamma_i(h)$ при степени оснований $K^{i-1}$ может принимать три значения в зависимости от значений $D_i$ и $C_i$ триггеров i-го разряда соответственно старшего и младшего регистров состояний на этапе (такте) преобразования h:

$$\gamma_i(h)=\begin{cases} a, D_i \neq 0; C_i \neq 0; \\ 1, D_i = 0; C_i \neq 0; \\ 0, D_i = 0; C_i = 0. \end{cases}$$

Коэффициент $R_i(h)$ учитывает вхождение компоненты i-го разряда (ее вклад) в выражение для $S_m$ блока m и равен

$$R_i(h)=\begin{cases} 0, D_i = C_i = 0; \\ 1, D_i \vee C_i = 1. \end{cases}$$

В зависимости от номера текущего такта преобразования h($h=\overline{0,H}$) значения коэффициентов $\gamma_i(h)$ и $R_i(h)$ могут изменяться, т.е. являются динамически изменяющимися, что относится также и к величине $S_m$.

| Номер набора | Состояние триггеров | | Общий вид эквивалента | Десятичный код эквивалента | Двоичный код эквивалента | | |
|---|---|---|---|---|---|---|---|
| | Второй группы | Первой группы | | | | | |
| $Z_i$ | $Д_3Д_2Д_1$ | $C_3C_2C_1$ | S | $S_{10}$ | $у_{12}у_{11}у_{10}у_9$ | $у_8у_7у_6у_5$ | $у_4у_3у_2у_1$ |
| 0 | 000 | 000 | $X_0$ | $X_0$ | Трансляция младшей тетрады | | |
| 1 | 000 | 001 | $K^1$ | 12 | 0000 | 000 | 1100 |
| 2 | 000 | 010 | $K^2$ | 144 | 0000 | 1001 | 0000 |
| 3 | 000 | 011 | $K^2+K^1$ | 156 | 0000 | 1001 | 1100 |
| 4 | 000 | 100 | $K^3$ | 1728 | 0110 | 1100 | 0000 |
| 5 | 000 | 101 | $K^3+K^1$ | 1740 | 0110 | 1100 | 1100 |
| 6 | 000 | 110 | $K^3+K^2$ | 1872 | 0111 | 0101 | 0000 |
| 7 | 000 | 111 | $K^3+K^2+K^1$ | 1884 | 0111 | 0101 | 1100 |
| 8 | 001 | 001 | aK | 24 | 0000 | 0001 | 1000 |
| 9 | 001 | 011 | $K^2+aK$ | 168 | 0000 | 1010 | 1000 |
| 10 | 001 | 101 | $K^3+aK$ | 1752 | 0110 | 1101 | 1000 |
| 11 | 001 | 111 | $K^3+K^2+aK$ | 1896 | 0111 | 0110 | 1000 |
| 12 | 010 | 010 | $aK^2$ | 288 | 0001 | 0010 | 0000 |
| 13 | 010 | 011 | $aK^2+K$ | 300 | 0001 | 0010 | 1100 |
| 14 | 010 | 110 | $K^3+aK^2$ | 2016 | 0111 | 1110 | 0000 |
| 15 | 010 | 111 | $K^3+aK^2+K$ | 2028 | 0111 | 1110 | 1100 |
| 16 | 011 | 011 | $aK^2+aK$ | 312 | 0001 | 0011 | 1000 |
| 17 | 011 | 111 | $K^3+aK^2+aK$ | 2040 | 0111 | 1111 | 1000 |
| 18 | 100 | 100 | $aK^3$ | 3456 | 1101 | 1000 | 0000 |
| 19 | 100 | 101 | $aK^3+K$ | 3468 | 1101 | 1000 | 1100 |
| 20 | 100 | 110 | $aK^3+K^2$ | 3600 | 1110 | 0001 | 0000 |
| 21 | 100 | 111 | $aK^3+K^2+K$ | 3612 | 1110 | 0001 | 1100 |
| 22 | 101 | 101 | $aK^3+aK$ | 3480 | 1101 | 1001 | 1000 |
| 23 | 101 | 111 | $aK^3+K^2+aK$ | 3624 | 1110 | 0010 | 1000 |
| 24 | 110 | 110 | $aK^3+aK^2$ | 3744 | 1110 | 1010 | 0000 |
| 25 | 110 | 111 | $aK^3+aK^2+K$ | 3756 | 1110 | 1010 | 1100 |
| 26 | 111 | 111 | $aK^3+aK^2+aK$ | 3768 | 1110 | 1011 | 1000 |

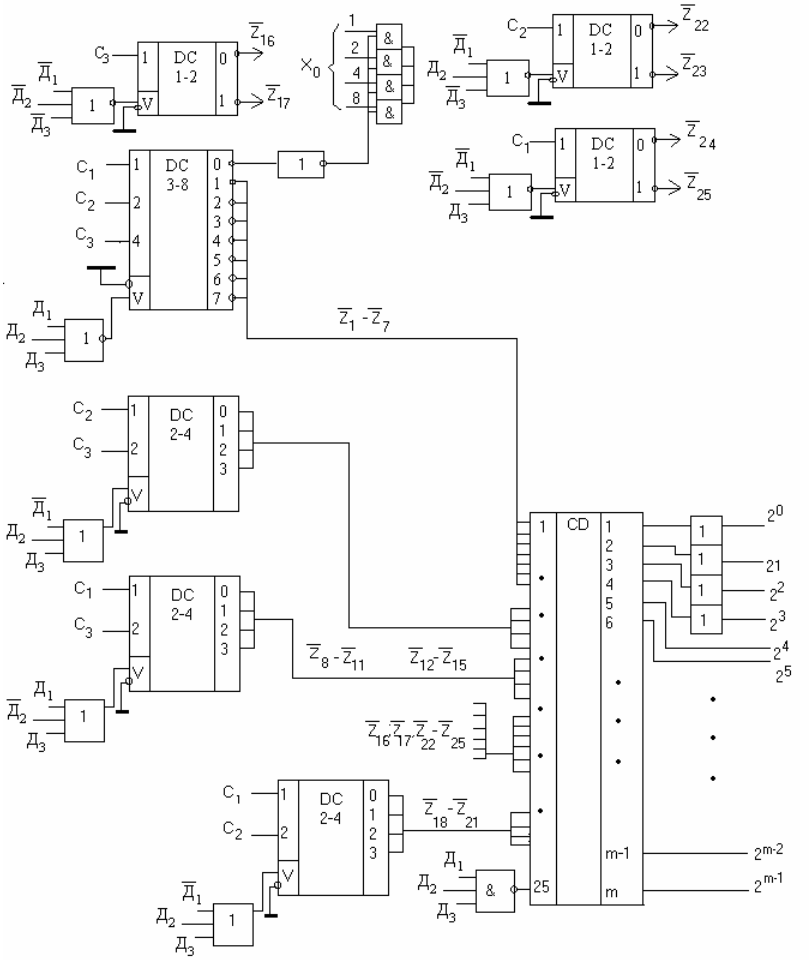



Рис. 3. Структура ФЭ для двухшагового ПК параллельного типа

Процесс преобразования заканчивается, если на каком-либо из тактов преобразования h состояния всех триггеров $D_i$ и $C_i$ всех блоков оказываются равными нулю. В случае разбиения шестиразрядного ПК и ФЭ на 6 блоков имеем следующие выражения для $S_1$- $S_6$:

$$S_1 = \gamma_1(h)K^0R_1(h); \quad S_2 = \gamma_2(h)K^1R_2(h); \quad S_3 = \gamma_3(h)K^2R_3(h); \quad S_4 = \gamma_4(h)K^3R_4(h);$$

$$S_5 = \gamma_5(h)K^4R_5(h); \; S_6 = \gamma_6(h)K^5R_6(h).$$

Число блоков разбиения (декомпозиции) М ФЭ и ПК на блоке является делителем числа входных разрядов n. В этом случае все блоки содержат одинаковое число разрядов. Так, при n=6 делителями являются числа 6,3,2,1 (M=1,2,3,6).

При М=3 получим следующие выражения для $S_m$:

$$S_1 = \sum_{i=1}^{i=2}\gamma_i(h)K^{i-1}R_i(h) = \gamma_2(h)K^1R_2(h) + \gamma_1(h)K^0R_1(h),$$

$$S_2 = \sum_{i=3}^{i=4}\gamma_i(h)K^{i-1}R_i(h) = \gamma_4(h)K^3R_4(h) + \gamma_3(h)K^2R_3(h),$$

$$S_3 = \sum_{i=5}^{i=6}\gamma_i(h)K^{i-1}R_i(h) = \gamma_6(h)K^5R_6(h) + \gamma_5(h)K^4R_5(h).$$

При М=2 выражения для вычисления значений $S_m$ ФЭ преобразуются в следующие:

$$S_1 = \sum_{i=1}^{i=3}\gamma_i(h)K^{i-1}R_i(h) = \gamma_3(h)K^2R_3(h) + \gamma_2(h)K^1R_2(h) + \gamma_1(h)K^0R_1(h),$$

$$S_2 = \sum_{i=6}^{i=6} \gamma_i(h) K^{i-1} R_i(h) = \gamma_6(h) K^5 R_6(h) + \gamma_4(h) K^3 R_4(h) + \gamma_3(h) K^2 R_3(h) .$$

Наконец, при M=1 получим:

$$S_1 = \sum_{i=1}^{i=6} \gamma_i(h) K^{i-1} R_i(h) = \gamma_6(h) K^5 R_6(h) + \gamma_4(h) K^3 R_4(h) + \gamma_3(h) K^2 R_3(h) +$$

$$+ \gamma_3(h) K^2 R_3(h) + \gamma_2(h) K^1 R_2(h) + \gamma_1(h) K^0 R_1(h) \cdot$$

Следует отметить, что если M является делителем числа n, то все блоки разбиения будут идентичными и иметь одинаковое число разрядов в блоке. Если это условие не выполняется, то последний блок разбиения будет иметь меньшее число разрядов по сравнению с предыдущими.

Так, при n=8 и M=3, будет два блока по три разряда ($S_1$ и $S_2$) и один блок (третий) с двумя разрядами. Функционирование ФЭ этих блоков описывается следующими формулами:

$$S_1 = \sum_{i=1}^{i=3} \gamma_i(h) K^{i-1} R_i(h) = \gamma_3(h) K^2 R_3(h) + \gamma_2(h) K^1 R_2(h) + \gamma_1(h) K^0 R_1(h);$$

$$S_2 = \sum_{i=4}^{i=6} \gamma_i(h) K^{i-1} R_i(h) = \gamma_6(h) K^5 R_6(h) + \gamma_5(h) K^4 R_5(h) + \gamma_4(h) K^3 R_4(h);$$

$$S_3 = \sum_{i=7}^{i=8} \gamma_i(h) K^{i-1} R_i(h) = \gamma_8(h) K^7 R_8(h) + \gamma_7(h) K^6 R_7(h).$$

**Выводы**

*Научная новизна* выполненного исследования заключается в следующем:

1. Предложен способ повышения быстродействия двухшаговых преобразователей кодов на базе счетчиков, функционирующих по методу накопления эквивалентов и использующих параллельную стратегию шагов преобразования.

2. Проведена сравнительная оценка последовательной и параллельной стратегии использования шагов преобразования и разработаны обобщенные математические модели, описывающие закон функционирования ФЭ в ПК с параллельной стратегией использования шагов преобразования.

3. Приведен пример построения закона функционирования ФЭ в ПК параллельного типа.

*Практическая значимость* исследования состоит в возможности существенного увеличения быстродействия двухшаговых ПК за счет небольших дополнительных аппаратурных затрат.

**Список литературы: 1.** А.с. 1126946 5G06F 5/02. Преобразователь двоично-К-ичного кода в двоичный код /А.Н. Слобожанин //Открытия, изобретения. 1984. №44. С.250. **2.** А.с. 1647908 5НОЗМ 7/12. Преобразователь двоично-К-ичного кода в двоичный код /Н.Я.Какурин, Ю.К. Кирьяков, А.Н. Макаренко // Там же. 1991. №17. С. 262-263. 3. А.С. 1783618 5G06F 5/02. Преобразователь двоично-К-ичного кода в двоичный код / Н.Я.Какурин, А.Н. Макаренко, Д.Ю. Исхаков, В.А. Толмацкий //Открытия, изобретения. 1984. №44. С.250.

*Поступила в редколлегию 17.09.2007*

**Коваленко Сергей Николаевич,** соискатель кафедры АПВТ ХНУРЭ. Научные интересы: цифровые датчики, устройства преобразования кодов, автоматизации проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

УДК 004.652.4+004.827

*Н.Б. ШАХОВСЬКА, Д.І. УГРИН*

# ВИРІШЕННЯ ПРОБЛЕМ КОМПЛЕКСНОГО АНАЛІЗУ В ПРОСТОРАХ ДАНИХ ТУРИЗМУ

Аналізуються проблеми, що виникають під час роботи з розрізненими джерелами з використанням сховищ та баз даних у галузі туризму. Вводиться модель простору даних як засобу інтеграції та опрацювання даних з розрізнених джерел.

## Вступ

Туризм є однією із провідних і найбільш динамічних галузей економіки й за швидкістю темпу розвитку він визнаний економічним феноменом століття.

У багатьох країнах туризм відіграє значну роль у формуванні валового внутрішнього продукту, активізації зовнішньоторговельного балансу, створенні додаткових робочих місць і забезпеченні зайнятості населення. Туризм впливає на такі ключові галузі економіки, як транспорт і зв'язок, будівництво, сільське господарство, виробництво товарів народного споживання й інші, тобто виступає своєрідним стабілізатором соціально-економічного розвитку. У свою чергу, на розвиток туризму впливають різні фактори: демографічні, природно-географічні, соціально-економічні, історичні, релігійні й політико-правові. На основі використання величезної кількості даних та інформації, а також з появою виникнення проблеми впорядкування та використання даної інформації логічним є також і те, що усе більше значущу роль у туристичному бізнесі відіграють бази даних, їх сховища та простори.

Як бази даних, так і сховища даних, будуються з допомогою систем управління базами даних (СУБД). Саме вони забезпечують загальні умови для зберігання і запиту структурованих даних. СУБД підтримує набір взаємозв'язаних послуг і гарантує розробникам можливість зосереджуватися на специфічних проблемах їх додатків, а не на завданнях, що повторюються, які виникають при потребі в узгодженому і ефективному управлінні великими об'ємами даних.

На жаль, в сучасних сценаріях управління даними рідкісні випадки, коли всі дані можуть знаходитися під управлінням традиційної реляційної СУБД або якої-небудь іншої моделі даних чи системи. Натомість розробники часто стикаються з набором слабозв'язаних джерел даних і тому доводиться кожного разу вирішувати низькорівневі завдання управління даними, що повторюються, в різнорідних колекціях. У число цих завдань входять забезпечення можливостей пошуку і запиту даних; дотримання правил, обмежень цілісності, угод про іменування і т.д.; відстежування походження даних; забезпечення доступності, відновлення і контролю доступу; керований розвиток даних і метаданих.

Проблеми такого роду виникають на всіх ланках туристичного бізнесу – на туристичних об'єднаннях (туристичних операторів і туристичних компаній): всередині адміністративних туристичних агентств і між ними, у великих туристичних готелях (електронних і звичайних) і навіть у великих мережевих компаніях. Проте в кожному з цих сценаріїв є деякі пізнавані і контрольовані межі між даними і базовими системами. Тому для управління даними туристичної сфери необхідно використовувати простори даних.

## 1. Актуальність роботи

*Мета* роботи – моделювання простору даних туристичної сфери.

Зазначимо особливості туристичної сфери, які не дозволяють для опрацювання їхніх даних використовувати сховища даних.

Оскільки прогнозувати розвиток інформаційних технологій у туризмі неможливо без розуміння й прогнозування тенденцій розвитку самого туристичного ринку, то розвиток піде по шляху глобалізації. Важливими факторами в туристичному бізнесі є швидкість і якість доведення до клієнта (агентства або туриста) інформації про турпродукт, хто швидше й зручніше обслужить клієнта. Тому сьогодні вже недостатньо автоматизації окремих аспектів діяльності компанії. Наприклад, якщо в туроператора автоматизована бухгалтерія,

але облік клієнтів ведеться окремо, а сайт є незалежне інформаційне сховище, то він постійно буде мати проблеми, пов'язані з інформаційним обміном між всіма підрозділами компанії. Так само й в агентствах. Якщо пошук турів ведеться в одній (або декількох) системі, а облік туристів - в іншій незалежній програмі, то витрати на перенесення даних в одне інформаційне сховище й усунення з іншого неминуче призводить до накладок, які будуть зводити нанівець переваги, що дає автоматизація.

## 2. Постановка задачі та аналіз літературних джерел

На сьогоднішній день туристичний ринок вимагає комплексної автоматизації всіх бізнес-процесів у рамках однієї компанії або навіть всього ринку. А це насамперед простір інформаційних даних туризму з декількома „вікнами" (інтерфейсами).

Через одне „вікно" (сайт) дивляться клієнти, через інше (внутрішню програму) –менеджери. Третє й четверте вікна – для бухгалтерії й адміністрації. Зрозуміло, що забезпечити єдиний інформаційний простір в рамках усього туристичного ринку нереально. Занадто багато гравців, занадто багато протиріч. Тому прогнозуючи та автоматизуючи інформаційні дані можна говорити лише про єдиний інформаційний простір.

Для повноцінної роботи простору даних туристичного бізнесу потрібна підтримка декількох моделей туристичних даних, оскільки проблеми в цьому напрямку знаходяться тільки в стані вивчення та досліджень. Щоправда інформація по опрацюванню різних джерел даних та використанні цих даних в єдиній предметній області з'явилася у 2006 р. Роботи розглядали проблеми, які привели до необхідності введення такої абстракції даних, як простір даних туризму. Серед них:

1. Інтеграція тексту, даних, коду і потоків (частково вирішена завдяки введенню Коддом 12-ти правил побудов сховищ даних та їх подальшої модифікації).
2. Забезпечення можливості багатоконтрольності даних (у концепції сховищ даних вирішувалася через процедури *витягання, перетворення та завантаження даних* (extract, transform and load – *ETL*) [1], а у випадку Інтернет із тисячами джерел інформації, поданої у різних форматах, ETL не придатна для використання).
3. Створення простих способів аналізу, узагальнення, пошуку і огляду електронних підбірок мультимедійної інформації, включаючи розробку стандартів опису метаданих (на даний момент нема єдиних стандартів опису та опрацювання мультимедійної інформації).
4. Підтримка неточних та невчасних (тих, що поступають із запізненням або в неочікуваному порядку) даних та реалізація неточних запитів. Останніми роками достатньо інтенсивно досліджується окремий випадок цієї проблеми, так звані top-K-запити та неточні запити. Знову ж таки, розроблені моделі та технології працюють лише із визначеними моделями даних (зокрема реляційною – Fquery, Fuzzy Grouping та Fuzzy Lookup [2]). Стосовно невчасних даних взагалі відсутні методи, які б дозволяли не тільки фіксувати факт запізнення даних, але й на основі цих тимчасово відсутніх даних приймати рішення (ведуться роботи в області машин для обробки подій, проте на даний момент задекларовані лише проблеми, які призводять до задачі маніпулювання потоками. Роботи, що були проведені у середині 90-х років в області активних баз даних (Active DBMS, ADBMS), залишилися невикористаними через великий час відклику запитів та неврахування тимчасової відсутності даних).
5. Проблема інтеграції даних, у тому числі надоперативних (частково вирішена оперативними сховищами даних – дані збираються та оперативно опрацьовуються, але залежать від зовнішньої структури) та частково структурованих (частково вирішено засобами пошуку неструктурованих даних [2]).
6. Використання природних мов запитів до баз даних (так звані системи з природномовним інтерфейсом) – передбачають формування запитів до системи у вигляді запитальних речень природної мови.
7. Підтримка систем обробки потокових даних (наприклад, система Postgres Майкла Стоунбрейкера, високорівнева мова «STREAMSQL» з вбудованими орієнтованими на потоки примітивами і операціями).
8. Повинна бути можливість ефективного зберігання, доступу і модифікації інформації про стан, а також її комбінування з реальними потоковими даними.

9. Для інтеграції в системі повинна використовуватися однорідна мова для роботи з усіма різновидами даних.

## 3. Основний матеріал

В туристичній сфері наперед не можна визначити, які саме моделі даних використовуватимуться. Тому виключно за допомогою баз даних та сховищ даних не можна організувати ефективної взаємодії між усіма об'єктами цієї предметної області. Розробники часто зустрічаються з набором слабозв'язаних джерел даних і тому повинні кожного разу вирішувати низькорівневі завдання управління даними. У число цих завдань входять забезпечення можливостей пошуку і запиту даних; дотримання правил, обмежень цілісності, угод про іменування і т.д.; відстежування походження даних; забезпечення доступності, відновлення і контролю доступу; керований розвиток даних і їх мета.

Щоб вирішити ці задачі, необхідно надавати запити конкретним засобам, задавати потрібні питання в потрібний час, а потім застосовувати отримані відповіді для досягнення переваг у конкурентній боротьбі в галузі туристичного бізнесу. Розглянемо схему концепції самостійного аналізу туристичних даних (рис. 1).

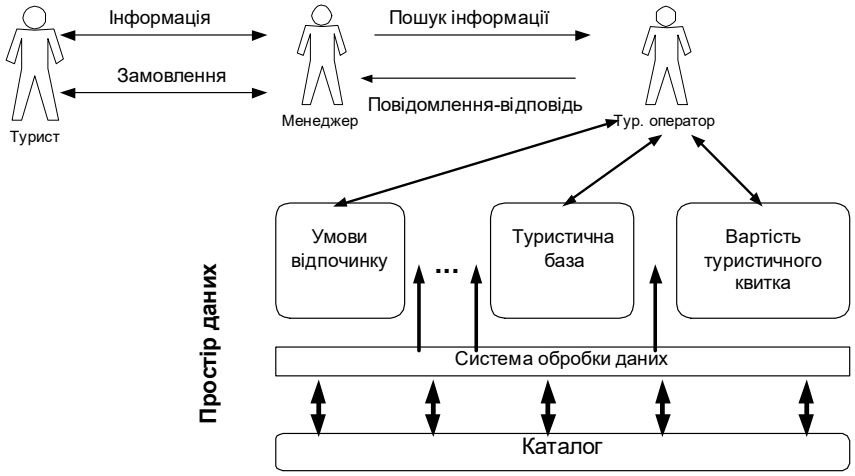



Рис.1. Схема аналізу туристичних даних

Як видно зі схеми, інформація, що повинна надатися туристичному клієнту, повинна бути знайдена в просторі даних, яка в свою чергу через менеджера та туристичного оператора буде відібрана відносно вподобань та вимог замовника.

В [1] зазначено, що простір даних DS – це множина даних, поданих у різних моделях (баз даних DB, сховищ даних DW, статичних веб-сторінок Wb, неструктурованих даних Nd, графічних та мультимедійних даних Gr), локальних сховищ та індексів ODW, а також засобів інтеграції Int, пошуку Se та опрацювання інформації Wo, об'єднаних середовищем управління моделями EM:

$$DS=<DB, DW, ODW, Wb, Nd, Gr, Int, Se, Wo, EM>.$$

Оскільки основним джерелом витягування інформації з простору даних вважається каталог CG, який в свою чергу є реєстром ресурсів даних, що містять найбільш базову інформацію про кожний з них: джерело, ім'я, місцеположення в джерелі, розмір, дата створення і власник і т.д., то він є інфраструктурою для більшості інших сервісів простору даних, але каталог також може підтримувати базовий, призначений для користувача інтерфейс переглядання простору даних:

$$DB, DW, Wb, Nd, Gr \Rightarrow CG.$$

Він не тільки містить описову інформацію (тобто виконує роль метаданих), але й зберігає для кожного учасника схему джерела, статистичні дані, швидкість зміни, точність, можливості відповідей на запити, інформацію про власника і дані, про політику доступу і підтримку конфіденційності. Оскільки джерела простору даних фізично не переносять у нього інформацію та можуть обмінюватись між собою інформацією, то у каталозі необхідно зберігати дані і про зв'язки між джерелами [1].

Поверх каталога розміщене середовище управління моделями, яке дозволяє створювати нові зв'язки і маніпулювати існуючими зв'язками (наприклад, об'єднувати або інвертувати відображення, зливати схеми і створювати єдині подання декількох джерел).

Для ідентифікації та роботи з неоднорідними колекціями в просторі даних можна використовувати глобальну схему імен (*Uniform Resource Identifiers – URI* ) як механізм посилань на глобальні константи, щодо яких є деяка угода між декількома постачальниками даних.

Важливою компонентою простору даних є компонента зберігання й індексування (ODW) для досягнення таких цілей:

– для створення асоціацій між об'єктами даних від різних учасників;

– для вдосконалення доступу до джерел з обмеженими власними засобами доступу;

– для забезпечення можливості виконання деяких запитів без доступу до реального джерела даних;

– для підтримки високого рівня доступності та відновлення.

Отже, зв'язок між каталогом CG, середовищем управління моделями EM, локальним сховищем та індексами ODW можна подати як функцію $EM(CG) \Rightarrow ODW$.

Чим більше моделей здатне «розрізнити» середовище управління, тим точніша буде інформація в ODW і тим ефективніше можна буде проводити процедури інтеграції, пошуку та опрацювання даних у просторі даних DS.

Оскільки одним із ключових питань простору даних є питання інтеграції, то розглянемо задачі їх виникнення:

– неоднорідність програмного середовища,

– розподілений характер організації,

– підвищені вимоги до безпеки даних,

– необхідність наявності багаторівневих довідників метаданих,

– потреба в ефективному зберіганні та обробці дуже великих об'ємів інформації.

Інтеграція інформаційних систем на основі веб-служб Int пов'язана з використанням чотирьох ключових стандартів:

*Extensible Markup Language* (XML) – розширена мова розмітки інформації. Описує інформацію, що пересилається по Інтернету. Запит на одержання яких-небудь даних чи виконання певних дій іншим застосуванням вимагає наявності способів передачі параметрів і одержання назад певних результатів. При використанні веб-служб ця інформація описується за допомогою мови XML, що є міжнародним загальноприйнятим стандартом для опису довільних даних, якими у свою чергу можуть обмінюватися інформаційні системи.

*Simple Object Access Protocol (SOAP)* – простий протокол доступу до об'єкта. Цей стандарт описує протокол виклику веб-служби (віддалений процес доступу до послуг інформації деякої прикладної системи), тобто передані параметри описуються за допомогою мови WSDL, а сам процес виклику описується за допомогою SOAP. У типовій ситуації взаємодії система однієї організації може викликати систему іншої організації, використовуючи протокол SOAP. Запит, що зазвичай містить ту чи іншу форму бізнес-документа, посилається ініціатором до запитуваної системи. Остання приймає запит, і вхідний документ, який міститься в запиті, обробляється. У результаті запитана система генерує відповідь, що повертається ініціатору взаємодії. Ініціатор також інформується про статус (успіх або інше) запиту.

*Web Services Description Language (WSDL)* – мова опису веб-служб. Це мова, яка базується на стандарті XML, що визначає спосіб доступу до веб-служб. Вона описує функціональні можливості веб-служб і групує операції взаємодії у певні інтерфейси, що задають способи виконання операцій і ті параметри, які повинні бути на вході і виході.

*Universal Description, Discovery, and Integration (UDDI)* – універсальний метод опису, виявлення та інтеграції. Технологія UDDI надає засоби, за допомогою яких можна зробити так, щоб будь-які застосування чи послуги, описані в термінах веб-служб, можуть бути розпізнані іншими застосуваннями та організаціями. Отже, це стандарт створення регістра, використовуючи який, можна описати організації і послуги, які вони надають, у вигляді, доступному для динамічного виявлення і взаємодії.

Інтеграція на основі веб-сервісів має декілька рівнів. Розглянемо її схему (рис. 2).

На рівні даних програмні застосування можуть обмінюватись інформацією. Цей рівень передбачає інтеграцію даних і є найпростішим. Наступний рівень — об'єктна взаємодія. Тут мова йде про те, що програмне застосування, розташоване на одному сервері, може запускати програмні процеси на іншому. Третій рівень інтеграції — інтеграція на рівні стандартної семантики. На цьому рівні сервіси можуть «спілкуватися спільною мовою», обходячи технологічні розбіжності. Один сервіс може звертатись до іншого із «запитом на виконання покупки», «запитом на виконання пошуку», «запитом на отримання статистики» та ін. На цьому рівні інтеграції сервіси будуть потребувати лише стандартизації семантики, тобто під словами «покупка», «пошук» і «статистика» вони повинні розуміти одне й те ж саме. Якщо семантичних розбіжностей між ними немає, інтеграція не має особливих труднощів, тобто використовуючи специфікацію WSDL, програмне застосування може «говорити» системно-незалежною мовою. З одного боку, системна незалежність застосувань постає з використання мови XML при створенні WSDL-описів, а з іншого — специфікація SOAP дозволяє взаємодіяти серверному та клієнтському застосуванням. Потрібно лише надати вхідні дані, а турботи про те, яким чином доставити їх застосуванню на обробку та повернути її результати назад, протокол SOAP повністю бере на себе [1].

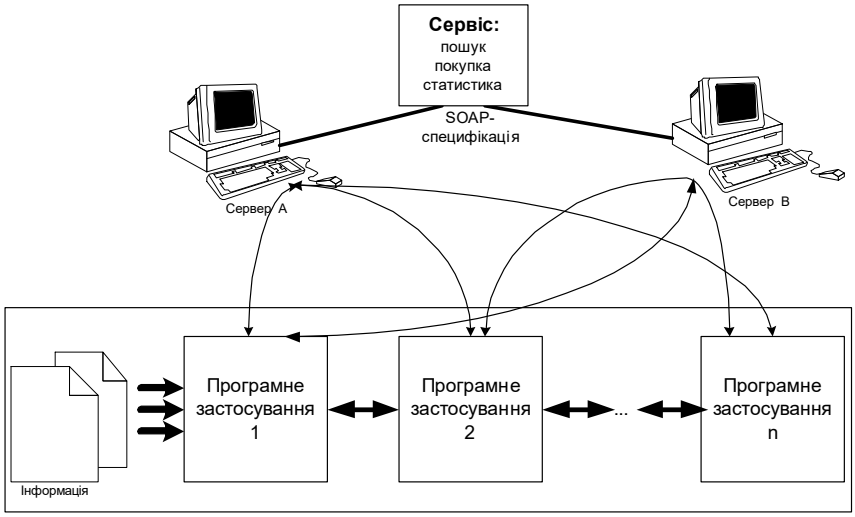



Рис. 2. Схема інтеграції на основі веб-сервісів

В той самий час слід сказати й про деякі недоліки технології веб-сервісів. В [2] зазначено основні недоліки: неоднозначність специфікації SOAP, недостатня безпека та недостатня швидкість роботи веб-сервісів.

Проте простори даних не є підходом до інтеграції даних; швидше це підхід співіснування даних. Мета підтримки простору даних полягає в забезпеченні базового набору функцій між всіма джерелами даних, а не в їх інтеграції. Наприклад, DSSP може забезпечити між всіма своїми джерелами даних пошук за ключовими словами, аналогічно тому, що забезпечують існуючі пошукові системи в десктопах. При потребі в складніших операціях, таких як запити в реляційному стилі, аналіз даних (data mining) або моніторинг яких-небудь джерел, можна докласти додаткові зусилля до тіснішої інтеграції цих джерел в інкрементній манері «оплати поточних рахунків» («pay-as-you-go»).

DSSP повинні працювати з даними і застосуваннями в різноманітних форматах, доступних від багатьох систем через різні інтерфейси. Від DSSP потрібна підтримка всіх даних простору даних, без яких-небудь виключень (як це буває при використанні СУБД). Тому одною із ключових задач побудови простору даних є визначення виразної потужності запитів із Se**.**

Хоча DSSP забезпечує засоби інтегрованого пошуку, запиту, оновлення і адміністрування просторів даних, ті ж самі дані часто можуть бути доступні для читання і оновлення через власний інтерфейс системи, що безпосередньо управляє даними. Тому, на відміну від СУБД, DSSP не має повного контролю над своїми даними.

Засоби опрацювання даних **Wo** повинні підтримувати:

– видобування даних (Data mining) – асоціативні правила, дерева рішень, генетичні алгоритми тощо,

– засоби аналізу даних (Online Analytical Processing – OLAP) – реляційний OLAP (Relational OLAP – ROLAP), багатовимірний OLAP (Multidimensional OLAP – MOLAP), гібридний OLAP (Hybrid OLAP – HOLAP), динамічний OLAP (Dynamic OLAP – DOLAP),

– засоби природно-мовного пошуку – побудова нечітких запитів, запитів у вигляді природних питань, запитів до метаданих,

– засоби підбору контенту на основі аналізу характеристик користувача,

– засоби миттєвого аналізу даних (наприклад, визначення причин отруєння або епідемічного захворювання туристів на певній території та пропонування методів усунення проблем).

Підхід сховищ (DataWarehouse) і вітрин (data mart) даних на основі витягання операційних даних, їх трансформації до єдиної схеми і завантаження даних в сховищі (процедура ETL – extraction, transformation, loading) придатний для використання для сфери з декількома десятками операційних баз даних, що знаходяться під єдиним контролем. У Internet парадигма ETL не прийнятна.

Зв'язок між елементами сховища даних поданий на рис. 3.

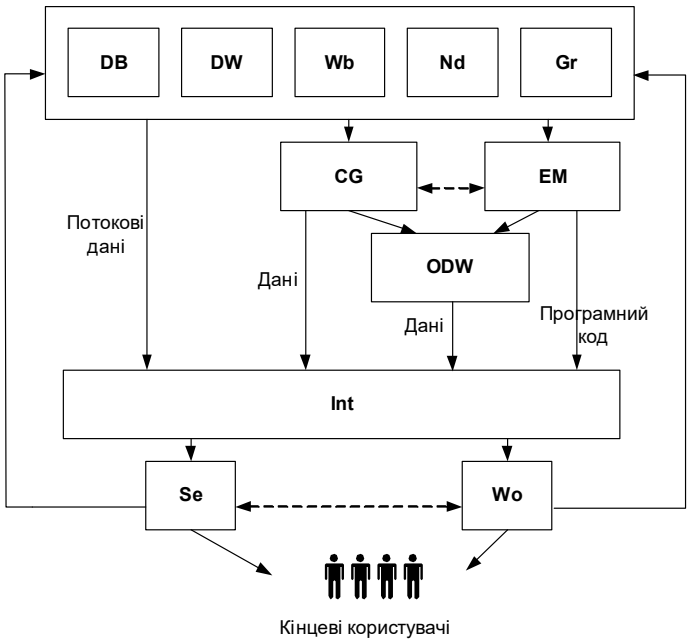



Рис. 3. Схема зв'язку елементів простору даних

Особливості просторів даних [5]:

– простори даних складаються з широкої різноманітності форматів та інтерфейсів і усі, без виключення формати даних повинні підтримуватися;

– дані у просторі даних не знаходяться у повному контролі;

– інтеграція тексту, даних, коду і потоків;

– підтримка структурованих, текстових, просторових, темпоральних, мультимедійних, процедурних даних; тригерів; потоків і черг даних як рівноправних компонентів;

– простори даних повинні забезпечувати вбудовану підтримку неточних даних. Мусить бути можливість задання неточних запитів, і процесор запитів повинен відноситися до цього як до додаткового джерела неповноти і неточності;

– відповіді на запити повинні залежати від профілю користувача. Відповідь на запит експерта мусить відрізнятися від відповіді на запит новачка. Релевантність відповіді теж повинна залежати від користувача і від контексту. Тому виникає необхідність середовища для накопичення і використання відповідних метаданих;

– система повинна знати точні взаємозв'язки між елементами, що використовуються у кожній схемі;

– DSSP пропонує рівні обслуговування та методи отримання приблизних відповідей;

– DSSP повинен запропонувати інструменти і шляхи створення щільнішої інтеграції даних в просторі в міру необхідності.

Можуть забезпечуватися різні рівні послуг з обробки запитів до DSSP, і в деяких випадках вони можуть повертати якнайкращі з можливих приблизні відповіді. Наприклад, якщо деякі джерела даних стають недоступними, DSSP може забезпечити найкращий з можливих результат на основі даних, доступних під час виконання запиту.

Для кращого розуміння принципів побудови та функціонування просторів даних розглянемо простір даних туристичної сфери. Об'єкти простору даних та його задачі подані на рис. 4.

Опишемо об'єкти простору даних в галузі туризму. Для простору даних необхідна інтергація інформації про такі об'єкти:

– місцеві органи управління – надають інформацію про відпочинкові, рекреаційні та оздоровчі ресурси, а також правила їх експлуатації; лінії сполучення, особливості місцевості тощо;

– туристичні агентства – надають інформацію про себе, про послуги, що вони надають.

– адміністративні одиниці – описуються через інформацію місцевих органів управління, а також через відклики попередніх відвідувачів;

– особа (відпочиваючий) – надає інформацію про себе, про умови, які він хоче отримати, ціни тощо.

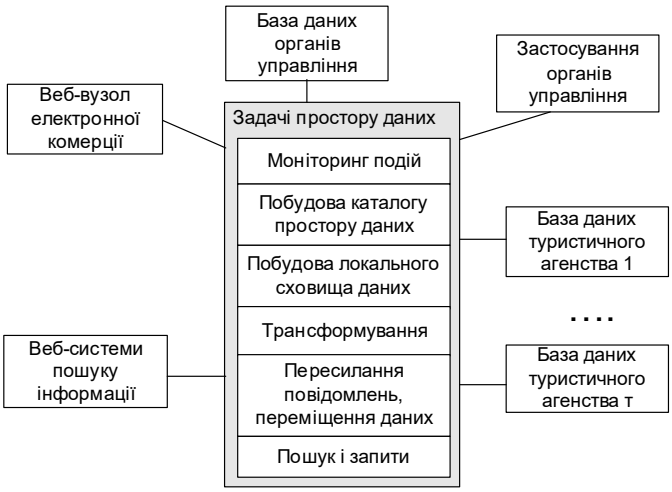



Рис. 4. Об'єкти простору даних та його задачі на прикладі туристичної сфери

Залежно від типу об'єкта інформація може зберігатися у різних моделях та надходити з різних джерел:

туристичне агентство – база даних, динамічний веб-сайт з базою даних, розміщеною на веб-сервері,

адміністративна одиниця – сховище даних,

особа – веб-сайт, база даних, текстові дані тощо,

відпочинковий ресурс – база даних, веб-сайт.

Оскільки специфікою галузі туризму є подання інформації в Інтернет у вигляді реклами, замовлень тощо, то на найвищому рівні ієрархії моделей даних знаходяться колекції іменованих ресурсів з базовими властивостями - розмір, дата створення і тип (наприклад, зображення JPEG, база даних MYSQL).

Галузь туризму накопичить мільйони даних протягом всього лише декількох років. Через велику кількість розрізнених ресурсів жодна людина не зможе знати ні все сховище повністю, ні те, що означає кожен файл. Людям, що звертаються до цих даних, особливо

тим, які не входять до сховища даної групи, знадобиться зведений реєстр основних атрибутів файлів, таких як період часу, до якого відноситься даний файл, географічний район тощо. Коли врешті-решт інформація знайдена, постає питання її узгодження та інтерпретації.

Незабаром таким групам потрібно буде об'єднуватися з іншими групами для створення туристичних просторів даних регіонального або національного масштабу. Їм потрібно буде якомога простіше імпортувати свої дані в стандартних форматах і з глибиною деталізації (частина файлу або декілька файлів). Користувачі федеральних просторів даних можуть захотіти побачити колекції даних, що належать різним групам федерації, наприклад, всі описи та відгуки відпочиваючих щодо опису певного природного ресурсу. Для швидкого пошуку в таких колекціях можуть знадобитися локальні копії або додаткові індекси.

**Висновки**

Подано формальну модель простору даних та наведено концептуальну модель простору даних в галузі туризму, що забезпечує взаємодію між джерелами іформації, поданої за допомогою різних моделей даних, з різними методами подання та опрацювання.

*Наукова новизна.* Новизна полягає в уведенні формального опису простору даних та окресленні його основних задач.

*Практична цінність* полягає у побудові простору даних в галузі туризму, виділенні основних об'єктів та учасників.

*Подальші дослідження* стосуватимуться формалізації методів інтеграції даних та пошуку неструктурованих, напівструктурованих та строго структурованих даних.

**Список літератури: 1.** *Шаховська Н.Б.* Простори даних: поняття та призначення //Матеріали конференції CSIT-2007. Львів. 2007. С.269-277. 2. *Кэтэрин Дрюэк (Katherine Drewek).* "Хранилища данных: сходство и различия подходов Билла Инмона и Ральфа Кимболла", 2005, http://www.b-eye-network.com/view/743. 3. *Dan Linstedt.* Data Vaulttm overview the next evolution in data modeling. – 2005, http://www.tdan.com/i021hy01.htm 4. *Donald Kossmann*, Jens-Peter Dittrich. Personal Data Spaces. http://www.inf.ethz.ch/news/focus/res_focus/feb_2006/index_DE. **5.** Garretts Summary of Principles of Dataspace Systems, http://aravaipa.eas.asu.edu/wiki/index.php/Garretts_Summary_of_Principles_of_Dataspace_Systems#Overview.

*Надійшла до редколегії 19.09.2007*

**Шаховська Наталія Богданівна,** канд. техн. наук, доцент кафедри інформаційних систем та мереж НУ «Львівська політехніка». Наукові інтереси: сховища даних, простори даних, методи інтеграції та опрацювання даних.

**Угрин Дмитро Ілліч,** аспірант НУ «Львівська політехніка». Наукові інтереси: простори даних у сфері туризму, методи опрацювання даних.

## РЕФЕРАТИ

УДК 006.91

**Автоматизація вимірювання частотних параметрів НВЧ-сигналу у хвилеводі** / І.П. Захаров, М.П. Сергієнко // АСУ та прилади автоматики. 2007. Вип. 140. С.4-8.

Розроблено новий метод вимірювання частоти НВЧ-сигналу в хвилеводі на основі вимірювання прохідної потужності. Метод дозволяє визначати відхилення частоти сигналу в хвилеводі за умов, коли основна частота сигналу відома або невідома. Метод може використовуватися у широкому діапазоні частот, легкий у практичній реалізації, піддається автоматизації.

Табл. 1. Іл. 1. Бібліогр.: 2 назви.

UDC 006.91

**The method of measurement of ultrahigh frequency signal's frequency and frequency deviation in a waveguide** / I.P. Zakharov, M.P. Sergienko // Management Information System and Devises. 2007. N 140. P.4-8.

The new method of ultrahigh frequency signal's frequency measurement based on throughput power measurement. The method is allow to determine frequency deviation in a waveguide if basic frequency is known or not known. The method can be in use in broad band, it's easy in practical realization and give away to automation.

Tab. 1. Fig. 1. Ref.: 2 items.

---

УДК 519.713:681.326

**Синтез моделей керуючих автоматів для SoC-фільтрів з конвеєрною архітектурою** / І.В. Хаханова/ // АСУ та прилади автоматики. 2007. Вип. 140. С.8-30.

Запропоновано моделі керуючих автоматів для пристроїв з конвеєрною архітектурою, що реалізують DSP задачі обробки зображень на основі SoC. Розроблено програмне середовище для генерування HDL-коду для згаданих типів керуючих автоматів. Даний підхід дозволяє спростити й автоматизувати процес проектування DSP- блоків з конвеєрною архітектурою.

Іл.17. Бібліогр.: 13 назв.

UDC 519.713:681.326

**Synthesis of the control FSM for SoC pipeline filter /** I.V. Hahanova // Management Information System and Devises. 2007. N 140. P.8-30.

Control FSM for pipeline devices implemented image processing DSP tasks on SoC was proposed. Program for generation of the proposed models HDL code was designed. This approach allows to simplify and automata process of the pipeline architecture DSP block design.

Fig. 17. Ref.: 13 Items.

---

УДК 004.5; 004.7; 004.8

**Адаптивний метод розробки структури інформаційно-аналітичної системи до умов її використання** / О.Я. Кузьомін, В.М. Левикін // АСУ та прилади автоматики. 2007. Вип. 140. С.31-39.

Запропоновано адаптивний метод розробки комплексу, що забезпечує інформаційно-аналітичний комплекс (ІАС), що відрізняється від існуючих більш якісною оцінкою ситуацій у заданий строк і з мінімальними економічними й соціальними ризиками в надзвичайних природних ситуаціях, що дозволяє одержати адаптивну до умов надзвичайних природних ситуацій структуру комплексу, що забезпечує ІАС.

Іл. 2. Бібліогр.: 3 назви.

UDC 004.5; 004.7; 004.8

**Adaptive Method of Information-Analytical System Structure Development to Conditions of its Application** / Kuzemin A. // Management Information System and Devises. 2007. N 140. P.31-39.

Adaptive method of information-analytical system (IAS) supporting complex development is offered; it differs from available ones in more qualitative estimation of the situations in the specified term and with minimal economic and social risks in natural emergencies, this makes it possible to receive the IAS supporting complex structure adaptive to the conditions of natural emergencies.

Fig. 2. Ref.: 3 items.

---

УДК 004.5; 004.7; 004.8

**Розробка моделі системи підтримки прийняття рішень в області сервісного обслуговування автоматів фінансового самообслуговування /**Н.В. Головій (Гусарь), Асер Даюб // АСУ та прилади автоматики. 2007. Вип. 140. С.39-43.

Запропоновано оригінальний метод, що базується на дослідженні процесу як сукупності ситуацій, представлених у гранульованому виді, дозволяє врахувати багато фактів, їх прямих і зворотних зв'язків, які не під силу "ручної" технології підтримки рішень, а також динамічно оцінити альтернативи прийнятого рішення.

Іл. 1. Бібліогр.: 3 назви.

UDC 004.5; 004.7; 004.8

**The development of decision-making support system model in the field of ATM maintenance / N.V. Goloviy (Gusar), Aser Dayub //** Management Information System and Devises. 2007. N 140. P.39-43.

The original method which is based on research of process as sets of situations, presented in the granulated kind is offered, allows to consider many facts, their straight lines and feedback which not on forces of "manual" technology of support of decisions, and also dynamically to estimate alternatives of the made decision.

Fig. 1. Ref.: 3 items.

---

УДК 004.896, 004.932

**Технологія розробки ПЗ СТЗ, тестування й адаптації до умов експлуатації /** І.В. Гарячевська // АСУ та прилади автоматики. 2007. Вип. 140. С. 43-48.

При створенні систем технічного зору (СТЗ) однієї з основних завдань є вибір методів обробки зображень і їхнього порядку. При цьому велике значення має стикування апаратного (обчислювальна техніка) і програмного (алгоритм обробки зображень) забезпечення. Технологія розробки, тестування й адаптації ПЗ СТЗ до умов експлуатації дозволить вирішити проблему стикування без етапу натурного випробування.

Іл. 4. Бібліогр.: 2 назви.

UDC 004.896, 004.932

**Design and test Technology and adaptation for software sts in operation** / I. V. Garyachevskaya // Management Information System and Devises. 2007. N 140. P.43-48.

At creation of systems of technical sight (STS) one of the primary goals is the choice of methods of processing of images and their order. Thus great value joining hardware (computer facilities) and program (has algorithm of processing of images) maintenance. The technology of working out, testing and adaptation STS to operation conditions will allow to solve a problem of joining without a stage of natural test.

Fig. 4. Ref.: 2 items.

---

УДК 001:002

**Принципи побудови сеансу тестування в комп'ютерній системі тестування знань OpenTEST2 //** О.С. Шкіль, С.В. Чумаченко, С.В. Напрасник, Г.В. Бабич, О.В. Гаркуша // АСУ та прилади автоматики. 2007. Вип. 140. С.49-56.

Розглянуто питання формування частково адаптованого сеансу тестування в комп'ютерній системі тестування знань OpenTEST2. Сеанс тестування формується як репрезентативна вибірка зі структурованої бази тестових завдань із використанням методів дискретної оптимізації. Дано рекомендації щодо формування структури бази даних тестових завдань системи OpenTEST2.

Іл. 10. Бібліогр.: 5 назви.

UDC 001:002

**Principles of construction of a testing session in knowledge testing computer system Opentest2** // A.S. Shkil, S.V. Chumachenko, S.V. Naprasnik, A.V. Babich, Ye.V. Garkusha // Management Information System and Devises. 2007. N 140. P.49-56.

The issues of forming of partially adopted testing session in computer system of knowledge testing OpenTEST2 have been considered. Testing session is formed as a representational sampling from structured database of test tasks with usage of methods of discrete optimization. The recommendations concerning formation of structured database of test tasks for OpenTEST2 system have been given.

Fig. 10. Ref.: 5 items.

---

УДК 658.5

**До питання аналізу структури задачі проектування топології мікроелектронних пристроїв /** В.В. Семенець, В.Г. Іванов // АСУ та прилади автоматики. 2007. Вип. 140. С.57-65.

Описано рішення задач трасування у класі простіших конфігурацій. Розглянуті способи введення фіктивних вершин. Наведено декілька алгоритмів трасування. Розглянуті можливості спрощення складних задач трасування до більш простих.

Іл. 7. Бібліогр.: 5 назв.

UDC 658.5

**To a question of analysis structure for topology development of microelectronics devises** /V. Semenetc, V. Ivanov // Management Information System and Devises. 2007. N 140. P.57-65.

Solving for class of simple configuration tracing task is described. Methods of null vertex are review. Several tracing methods are described. Possibilities of reduction for difficult trace tasks are considered.

Fig. 7. Ref.: 5 items.

УДК 658.52.011.56

**Алгоритми рішення задач теорії розкладів на основі прогнозу. Частина 2** / М.В.Костікова, В.О.П'яніда // АСУ та прилади автоматики. 2007. Вип. 140. С.66-75.

Описано алгоритм рішення загальної задачі теорії розкладів, в якому використовується прогноз поточної довжини розкладу, представлені статистичні характеристики похибок алгоритму, одержані експериментальним шляхом.

Табл. 3. Іл. 5. Бібліогр.: 4 назви.

UDC 658.52.011.56

**Algorithms of scheduling theory problem solutions on the basis of prediction. Part 2** / M.V.Kostikova, V.A.Pyanida // Management Information System and Devises. 2007. N 140. P.66-75.

The article describes an algorithm of general scheduling theory problem solution wherein a prediction of current schedule length is used, statistical characteristics presented of algorithm error as obtained experimentally.

Tab. 3. Fig. 5. Ref.: 4 items.

---

УДК 519.713:681.326

**Верифікація цифрових пристроїв на основі використання аналізу тестопридатності та ассерційних бібліотек** / В.І. Хаханов, М.О. Камінська, С.О. Зайченко // АСУ та прилади автоматики. 2007. Вип. 140. С.75-83.

Запропоновано метод розрахунку показників тестопридатності, а також стратегія вибору точок для подальшої модифікації графа FSM з метою збільшення його тестопридатності та покращення якості покриття несправностей тестом. Зменшено час синтезу тестів шляхом введення додаткових змінних при розділенні режимів тестування та функціонування. Запропонований метод дозволяє проводити аналіз на самих ранніх стадіях проектуваня, що дозволить підвищити Yield Ratio.

Табл. 3. Іл. 4. Бібліогр.: 15 назв.

UDC 519.713:681.326

**Digital Devices verification based on the testability analysis and assertions libraries** / V.I. Hahanov, M.O. Kaminska, S.O. Zaychenko // Management Information System and Devises. 2007. N 139. P.75-83.

New method of testability analysis for FSM and strategy of bottlenecks selection is offered. It is proposed the methodology of FSM modification by using test control points and assertions for further test quality and fault coverage improving. Test time was decreased by using additional variables for separating of normal mode (device functionality) and test mode. Proposed method allow to provide analysis on earliest design stages to avoid great number of defects and increase Yield Ratio.

Tab. 3. Fig. 4. Ref.: 15 items.

---

УДК 004.652

**Моделювання РБД з довільною топологією /** В.В. Євсєєв, Ю.В. Шовкопляс, Н.В. Самойленко // // АСУ та прилади автоматики. 2007. Вип. 140. С.83-92.

Досліджена розподілена база даних з довільною топологією. У зв'язку з високою складністю об'єкта дослідження формалізація процесу моделювання реалізована на основі імітаційного та аналітичного підходів. Як формальний апарат вибрано теорію масового обслуговування. Дані методи реалізовані на ЕОМ. Отримані експериментальні дані підтверджують доцільність та ефективність використаних методів залежно від вихідних даних.

Табл. 2. Іл. 5. Бібліогр.: 9 назв.

UDC 004.652

**Modeling of free-topology DDB /** V.V. Evseev, Y.V. Shovkoplyas, N.V. Samoylenko // Management Information System and Devises. 2007. N 140. P.83-92.

Object of research is a free-topology distributed database. In this work, in dependence of high complication of object of research, formalization of modeling process realized on imitation an analytic approach base. As the formal tool, was chosen queuing theory. These methods were realized with computer. Experimental results corroborate expediency and efficiency of used methods in dependence on initial data.

Tab. 2. Fig. 5. Ref.: 9 items.

---

УДК 621.318.5:002

**Модель управління технологічною системою з метою вибору оптимальних параметрів й усунення небажаних та шкідливих факторів** //Є.М. Анпілогов, І.Є. Анпілогова, Б. В.Дзюндзюк, Л. І.Марченко, Л.В. Ларченко // АСУ та прилади автоматики. 2007. Вип. 140. С.92-96.

Описано технологічний процес при виготовленні системи з урахуванням вхідних, вихідних параметрів та режимів операцій. Актуальність задачі полягає в тому, що залежно від вхідних параметрів та режимів окремих операцій можна спрогнозувати вихідні параметри готового виробу. На технологічних операціях за допомогою режимів усуваються небажані та шкідливі фактори.

Іл.1. Бібліогр.: 6 назви.

UDC 621.318.5:002

**Model of controlling a technological system with the aim of selecting optimum parameters and eliminating undesirable and dangerous factors** //E.Anpilogov, I.Anpilogova, B.Dzyundzyuk, L.Marchenko, L. Larchenko // Management Information System and Devises. 2007. N 140. P.92-96.

The technological process of making the system with taking into consideration of input and output parameters and operation regimes is modulated.The actuality of the problem is that one can forecast output parameters of a ready-made product depending on input parameters and regimes of individual operation. Within the technological operations undesirable and dangerous factors can be eliminated with the help of regimes.

Fig.1. Ref.: 6 items.

---

УДК 681.586.37:004.5

**Програмний засіб для аналізу перетворення чисел /** М.Я. Какурін, Ю.В. Лопухін, Н.М. Бикова // АСУ та прилади автоматики. 2007. Вип. 140. С.96-102.

Розглянуто структуру та функціональні можливості програмного засобу для аналізу перетворювачів кодів за методом накопичення еквівалентів. Його використання дозволяє скоротити етап системного проектування.

Іл. 7.Бібліогр.: 4 назви.

UDC 681.586.37:004.5

**The software for analyzing of the number conversion** / N.Ya Kakurin, Yu.V. Lopukhin, N. N. Bykova // Management Information System and Devises. 2007. N 140. P.96-102.

The structure and the functional possibilities of the software for analyzing code convertors with the method of accumulation of an equivalents are considered. It's using is permitting time minimization of system design.

Fig. 7. Ref: 4 items.

---

УДК 681.325.53: 37 : 004.5

**Двокроковий перетворювач кодів з паралельним використанням кроків перетворення** / С.М. Коваленко // АСУ та прилади автоматики. 2007. Вип. 140. С.103-109.

Запропоновано спосіб підвищення швидкодії перетворювачів кодів на базі лічильників, що використовують метод накопичення еквівалентів, за рахунок паралельного використання кроків перетворення. Розглянуто структуру та функціонування швидкодіючого двокрокового перетворювача кодів та математичні моделі формувачів еквівалентів багатоблокових ПК.

Табл. 1. Іл. 3. Бібліогр.: 3 назви.

UDC 681.325.53:37 : 004.5

**The two-steps code converter with parallel using of conversion steps** / S.N. Kovalenko // Management Information System and Devises. 2007. N 140. P.103-109.

The way for rising of the speed of code convertor based on a counters is offered by the parallel using of a conversions steps/ The structure and functioning of the two-steps code convertor and mathematical model of equivalents former for multiblocks code converter are considered.

Tab. 1. Fig. 3 Ref.: 3 items.

---

УДК 004.652.4+004.827

**Вирішення проблем комплексного аналізу в просторах даних туризму** / Н.Б. Шаховська, Д.І. Угрин // АСУ та прилади автоматики. 2007. Вип. 140. С.110-117.

Системи інтеграції даних і обміну даними традиційно призначаються для підтримки багатьох інших осмислених служб в системах просторів даних. Особливість статті полягає у тому, що в системах інтеграції даних потрібна семантична інтеграція до того, як можуть бути забезпечені які-небудь інші послуги. Тому в статті уведений формальний опис простору даних у галузі туризму та окреслення його основних задач.

Іл. 4. Бібліогр.: 5 назв.

UDC 004.652.4+004.827

**Complex analysis of dataspace in tourism sphere** / N.B.Shakhovska, D.I.Ugrin /// Management Information System and Devises. 2007. N 140. P.110-117.

Systems integrations given and it is traditionally targeted an exchange information at support many other intelligent services in the systems of spaces given. The feature of the article consists in that in the systems integrations given semantic integration is needed till can be well-to-do some other services. That is why in the articles the formal specification of space of information is entered in industry of tourism and outlining of his basic tasks.

Fig. 4. Ref.: 5 items.

## ПРАВИЛА

### оформления рукописей для авторов научно-технического сборника *"АСУ и приборы автоматики"*

Формат страницы — А4 (210х297мм), поля: сверху, справа, слева, снизу – 30 мм. Редактор: Pagemaker 6.0, 6,5 (можно, но нежелательно Word), гарнитура Times New Roman Cyr, кегль – 11 пунктов, межстрочное расстояние — 110 %, табуляция — 5 мм.

Объем рукописи – до 10 с. (языки: русский, украинский, английский). Содержание должно отражать актуальность исследования, постановку задачи, цель, сущность, научные и практические результаты, сравнение с лучшими аналогами, выводы.

Структура рукописи: заголовок, аннотация, текст, литература, реферат на украинском и английском языках, сведения об авторах.

### *ОБРАЗЕЦ ОФОРМЛЕНИЯ*

---

УДК 519.713

*И.О. ФАМИЛИЯ*

**НАЗВАНИЕ РУКОПИСИ**

---

Аннотация (абзац 5-10 строк, кегль 10) помещается в начале статьи и содержит информацию о результатах описанных исследований.

Основной текст можно разделять на 2 и более подразделов с заголовками, выделенными полужирным шрифтом, пронумерованными арабскими цифрами, как показано в следующей строке.

**1. Название раздела**

Рисунки и таблицы (черно-белые, контрастные) помещаются в текст после первой ссылки в виде *переносимых объектов* и раздельно нумеруются, при наличии более одного рисунка (таблицы), арабскими цифрами. Рисунок содержит подрисуночную центрированную подпись (текстовая строка, расположенная вне рисунка, кегль 10) под иллюстрацией, как показано на рис. 1.

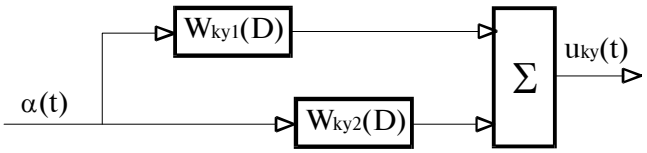



Рис.1.Двухзвенная система

Табличный заголовок располагается справа над таблицей, что иллюстрируется табл.1. Редакторы: CorelDraw, Table Editor и др.

Таблица 1

| Шаг i | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\Phi_1(1,3)$ | 1 | 2 | 2 | 4 | 6 | 1 |
| $\Phi_1(1,4)$ | 1 | 2 | 2 | 4 | 4 | 8 |

Формулы нумеруются при наличии ссылок на них в рукописи. Рекомендуемый кегль формульного набора: обычный (переменная) – 11 пунктов, крупный индекс – 8, мелкий индекс (над- и подиндекс) – 8, крупный символ (основной) – 12, мелкий (индексный) математический символ – 10:

$$F_{i+j} = \sum_{i=1}^{b^k} F_j^i - \prod_{j=1}^{1+h^2} P_{R\,j+i} + F^{j-1} + X^{\Sigma n^k} . \qquad (1)$$

Формат переменных (желательно не курсивом – без наклона) в тексте и формулах должен быть идентичным. В тексте над- и подиндексы составляют 70 % от кегля, которые рекомендуется опускать (поднимать) на 17 (33) % относительно основной строки.

Список литературы (включает опубликованные источники, на которые имеются ссылки в тексте, заключенные в квадратные скобки) печатается без отступа, кегль 9 пунктов.

Образец окончания текста рукописи (литература, сведения об авторах, реферат) представлен ниже.

**Список литературы:** 1. *Фамилия И.О.* Название книги. Город: Издательство, 1900. 000 с. 2. *Название* сборника / Под ред. И.О. Фамилия. Город: Издательство, 1900. 000 с. 3. *Фамилия И.О.* Название статьи / / Название журнала. Название серии. 1997. Т. 00, № 00. С. 00-00 .

*Поступила в редколлегию 00.00.00*

**Фамилия, имя, отчество**, ученая степень, звание, должность и место работы. Научные интересы. Адрес, контактный телефон.

Рефераты на украинском и английском языках. Текст аннотации не должен дублировать реферат.

---

УДК 000.000.00

**Назва статті українською мовою** / Ініціали. Прізвище // АСУ та прилади автоматики. 2000. Вип. 00. С. 000-000.

Текст реферату.

Табл. 00. Іл. 00. Бібліогр.: 00 назв.

UDC 000.000.00

**Title of paper** / Initials. Surname // Management Information System and Devices. All-Ukr. Sci. Interdep. Mag. 2000. N 00. P. 000-000.

Text.

Tab. 00. Fig. 00. Ref.: 00 items.

Представление материалов

Рукопись, реферат, сведения об авторах — в одном файле, *поименованном фамилией первого автора*, на дискете 3,5 дюйма. Твердая копия материалов – для граждан Украины — в одном экземпляре: рукопись, подписанная авторами, рефераты, акт экспертизы, внешняя рецензия, подписанная доктором наук, заявление на имя главного редактора со сведениями об авторах.

Адрес редакции: Украина, 61166, Харьков, просп. Ленина, 14, ХНУРЭ, комната 321, тел. 70-21-326.

E-mail: hahanov@kture.kharkov.ua.

Тематика статей, публикуемых в сборнике:

– Компьютерная инженерия
– Математическое моделирование
– Оптимизация и процессы управления
– Автоматизация проектирования и диагностика
– Информационные интеллектуальные системы
– Проектирование интегральных схем и микросистем
– Компьютерные технологии в образовании

Відповідальний випусковий В.І. Хаханов
Редактор О.П. Гужва
Комп'ютерна верстка Г.В. Хаханова, С.В. Чумаченко

Підп. до друку 27.09.2007. Формат 60x84 1/8. Умов. друк. арк. .
Обл.-вид. арк. 11,8. Тираж 300 прим.
Зам. № . Ціна договірна.

Харківський національний університет радіоелектроніки (ХНУРЕ).
Україна, 61166, Харків, просп. Леніна, 14.

Оригінал-макет підготовлено і збірник віддруковано
в навчально-науковому видавничо-поліграфічному центрі ХНУРЕ.
Україна, 61166, Харків, просп. Леніна, 14.