МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

ISSN 0135-1710

# АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

**Всеукраинский межведомственный научно-технический сборник**

**Основан в 1965 г.**

## Выпуск 167

Харьков
2014

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 681.326:519.713

*В.И. ХАХАНОВ, И.В. ХАХАНОВА, TAMER BANI AMER,*
*FARID DAHIRI (ФАРИД ДАХИРИ)*

# ИМПЛЕМЕНТАЦИЯ КУБИТНЫХ СТРУКТУР ДАННЫХ В ПАРАЛЛЕЛЬНЫЙ ВЫЧИСЛИТЕЛЬ

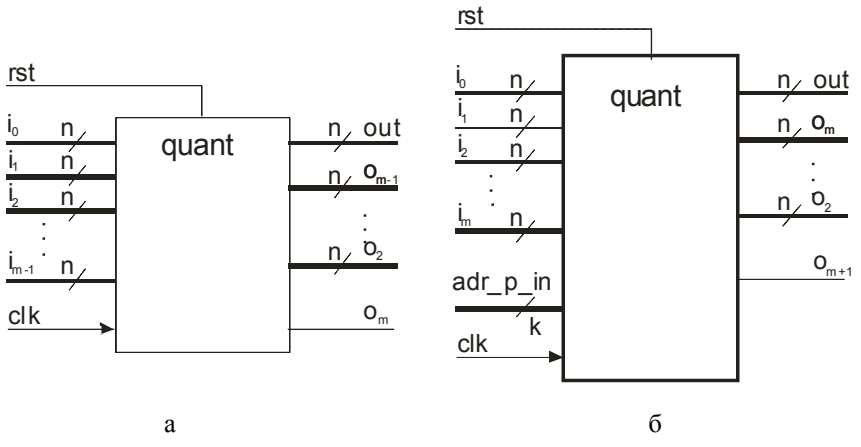
Предлагается практическая реализация генератора HDL-кода «квантовых» процессоров, использующих диаграммы Хассе для параллельных векторно-логических (теоретико-множественных) вычислений булеанов, применяемых для ускорения решения задач, моделирования, верификации, диагностирования. Программно-аппаратная реализация процессора основывается на использовании языков программирования: C++, Verilog, Python 2.7 и платформ: Microsoft Windows, X Window (в Unix и Linux) и Macintosh OS X. Генератор HDL-кода дает возможность автоматически синтезировать HDL-коды процессорной структуры от 1 до 16 двоичных разрядов для параллельной обработки соответствующего количества входных векторов или слов. Верификация HDL-кода процессора выполняется на тестовых примерах задачи покрытия, использующих две стратегии оптимизации: реверсивный алгоритм для устранения избыточности и разбиение матрицы покрытий на части в целях их последующей параллельной обработки процессорами Хассе.

## 1. Описание Verilog-модели устройства

Интерфейс. Модель процессора реализована с помощью одного из основных языков разработки ASIC-проектов – языка описания аппаратуры Verilog.

Интерфейс устройства в двух версиях представлен на рис. 1. Кроме входов синхронизации (clk) и сброса(rst), он содержит входные векторы данных $i_i$: m – количество входов данных, n – разрядность векторов. Выход out – результат объединения всех векторов входных данных. Его разрядность равняется разрядности входных векторов и равна n. Выходы $o_i$ содержат значения признаков соответствующего уровня, бит вектора равен 1, если вектор в соответствующей точке содержит все единицы, и 0 – в противном случае; i – обозначает уровень обработки векторов. Например, 8-входовой процессор с 8 разрядными входами данных будет иметь следующие выходы:

out – 8-битовый выход;
o7 – 8-битовый выход признака 7-го уровня;
o2, o6 – выходы признаков 2-го и 6-го уровней;
o3, o5 – выходы признаков 3-го и 5-го уровней;
o4 – выходы признаков 4-го уровня;
o8 – выход признака 8-го уровня.




а б

Рис. 1. Интерфейс квантового процессора

Разрядность выходов $o_i$ зависит от числа входов $i_m$ и вычисляется по формуле биномиальных коэффициентов

$$\binom{m}{k} = C_m^k = \frac{m!}{k!(m-k)!}, \quad (1)$$

где m – количество входов данных, а k – уровень их обработки в процессоре. Для m=8-16 параметры выходов приведены в табл. 1.

Таблица 1

Параметры выходов процесора

| Уровень | Выход | Разрядность | | | | |
|---|---|---|---|---|---|---|
| | | m=8 | m=10 | m=12 | m=14 | m=16 |
| 2 | o2 | 28 | 45 | 66 | 91 | 120 |
| 3 | o3 | 56 | 120 | 220 | 364 | 560 |
| 4 | o4 | 70 | 210 | 495 | 1001 | 1820 |
| 5 | o5 | 56 | 252 | 792 | 2002 | 4368 |
| 6 | o6 | 28 | 210 | 924 | 3003 | 8008 |
| 7 | o7 | 8 | 120 | 792 | 3432 | 11440 |
| 8 | o8 | 1 | 45 | 495 | 3003 | 12870 |
| 9 | o9 | | 10 | 220 | 2002 | 11440 |
| 10 | o10 | | 1 | 66 | 1001 | 8008 |
| 11 | o11 | | | 12 | 364 | 4368 |
| 12 | o12 | | | 1 | 91 | 1820 |
| 13 | o13 | | | | 14 | 560 |
| 14 | o14 | | | | 1 | 120 |
| 15 | o15 | | | | | 16 |

Как видно из табл. 1, количество выходов значительно увеличивается с ростом числа входных векторов, поэтому в данном случае реализуется интерфейс (см. рис. 1, б) с мультиплексированием выходов $o_i$. В нем вход adr_p_in используется как адрес для выбора необходимого выхода признаков или выводится только часть регистров признаков.

Интерфейс устройства на языке Verilog для 16-входового процессора представлен в листинге 1. По умолчанию для входов и выходов используется тип данных wire.

Листинг 1

Интерфейс устройства на языке Verilog

```
// Вариант 1
module device
 (input clk, rst,
  input [7:0] i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15,
  output [7:0] out, o15,
  output o16);
// Вариант 2
module device
 (input clk, rst,
  input [7:0] i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15,
   input  [24:0] adr_p,
   output [7:0] out, p_o16,
   output o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14, o15, o16,
   output [7:0] o_pr);
```

При увеличении числа входных векторов размер кода существенно увеличивается, что требует больше времени на моделирование и синтез схемы. Производительность систем, работающих с моделями цифровых устройств, существенно повышается, если модель имеет иерархическую структуру, поэтому выгоднее выделить элементарную ячейку процессора в виде отдельного Verilog-модуля (листинги 2, 3) и на основе его строить схему

процессора (рис.2). Выход регистра – вектора пересечений представлен сигналом O, сигнал p_in соответствует регистру признака для элементарной ячейки, вход en_p позволяет отключать или подключать выход признака p. Параметр BITS определяет разрядность входных векторов.

Листинг 2

Элементарная ячейка

```
module element
  #(parameter BITS = 8)
  (input rst, clk,
   input [BITS-1:0] I0, I1,
   input en_p,
   output reg [BITS-1:0] O,
   output p);
  reg  p_in;
   always @(posedge clk, posedge rst)
      if (rst)    O <= 'b0;
      else        O <= I0 | I1;
   always @(posedge clk, posedge rst)
      if (rst)     p_in <= 1'b0;
      else         p_in <= &O;
    assign p = (en_p)? p_in: 1'bZ;
endmodule
```

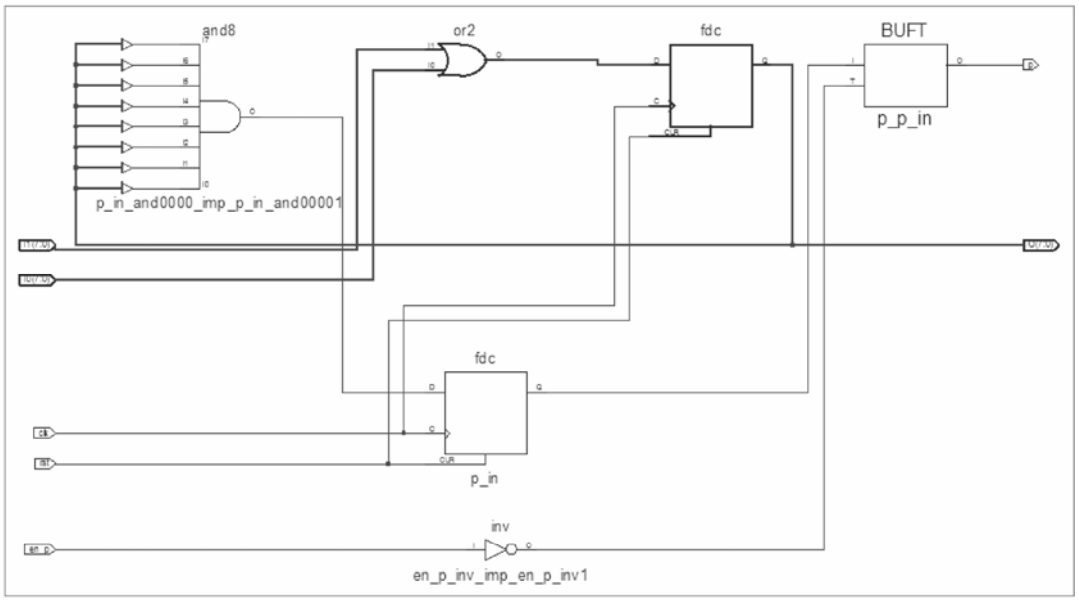



Рис. 2. Синтез схемы элементарной ячейки в Xilinx ISE Design Suite 11.1

Листинг 3

Фрагмент отчета синтеза элементарной ячейки процессора

```
=========================================================================
* HDL Synthesis *
=========================================================================

Performing bidirectional port resolution...
Synthesizing Unit <element>.
Related source file is "component.v".
Found 8-bit register for signal <O>.
Found 1-bit tristate buffer for signal <p>.
Found 1-bit register for signal <p_in>.
```

```
Summary:
inferred 9 D-type flip-flop(s).
inferred 1 Tristate(s).
Unit <element> synthesized.
=========================================================================

HDL Synthesis Report
Macro Statistics
# Registers : 2
1-bit register : 1
8-bit register : 1
# Tristates : 1
1-bit tristate buffer : 1
```

При использовании элементарной ячейки процессора ее подключение выполняется так, как показано в листинге 4. Каждая ячейка соединяется с декодером адреса adr_p_in, определяющим, будет ли данный признак в текущий момент подан на выход o_pr или нет. Декодер основан на тристабильном буфере. В данном фрагменте ячейки принадлежат второму уровню процессора и обрабатывают информацию с входов $i_0$ и $i_1$-$i_6$. Ячейке comp_0_1 соответствует адрес ‘d16, это означает, что если на вход adr_p_in подать значение 16, то значение признака поступит на внешний выход.

Листинг 4

Фрагмент структурной модели процессора

```
// COMPONENT LEVEL: 2
assign o_pr = (adr_p_in == ‘d16) ? p_0_1 : ‘bZ;
element #(8) comp_0_1 (rst, clk, r_0, r_1, r_0_1, p_0_1);

assign o_pr = (adr_p_in == ‘d32) ? p_0_2 : ‘bZ;
element #(8) comp_0_2 (rst, clk, r_0, r_2, r_0_2, p_0_2);

assign o_pr = (adr_p_in == ‘d48) ? p_0_3 : ‘bZ;
element #(8) comp_0_3 (rst, clk, r_0, r_3, r_0_3, p_0_3);

assign o_pr = (adr_p_in == ‘d64) ? p_0_4 : ‘bZ;
element #(8) comp_0_4 (rst, clk, r_0, r_4, r_0_4, p_0_4);

assign o_pr = (adr_p_in == ‘d80) ? p_0_5 : ‘bZ;
element #(8) comp_0_5 (rst, clk, r_0, r_5, r_0_5, p_0_5);

assign o_pr = (adr_p_in == ‘d96) ? p_0_6 : ‘bZ;
element #(8) comp_0_6 (rst, clk, r_0, r_6, r_0_6, p_0_6);
```

Присваиваемый ячейке адрес строится на основе ее индекса. Под каждый индекс отводится необходимое для его представления количество разрядов. Например, для 16 входных переменных на каждый индекс записывается 4 бита, индексы кодируются справа налево. Так, для ячейки с индексом 0_1 адрес в двоичном виде будет выглядеть как 0001_0000, что соответствует 16 в десятичной форме.

## 2. Генератор

Для генерации кода моделей устройств различной конфигурации разработан программный генератор. Создание моделей с большим числом входных векторов возможно только с помощью генератора.

Исходными данными являются: количество входных векторов и их разрядность, на их основе выполняется генерация Verilog-кода. Программа написана с использованием языка Python 2.7. Разработаны две версии генератора, которые реализуют поведенческую и структурные модели процессора.

2.1. Генератор поведенческой модели процессора

Структура поведенческой Verilog-модели представлена на рис. 3. Она включает декларацию имени модуля (Заголовок), декларативные части для объявления входных и выходных линий, переменных, соответствующих регистрам каждого уровня; код, описывающий регистры и значения признаков; завершающую часть, закрывающую модуль.

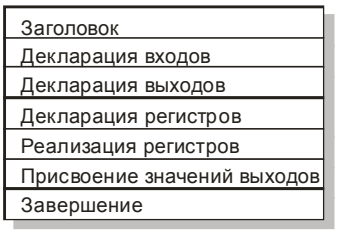



Рис. 3. Организация Verilog-модели

Для генерирования кода разработан основной класс Generator, а также вспомогательный класс genIndex, генерирующий индексы внутренних регистров модели. Структура классов, описанная с помощью инструмента ArgoUML, представлена на рис. 4. Переменные класса genIndex: levels – количество уровней модели (соответствует числу входов), level – текущий уровень для генерирования индексов регистров, ind_lisc – список строковых значений индексов. Два метода класса: genLevel() и getIndList() генерируют и возвращают список индексов. Объекты класса genIndex используются в методах создания описания и реализации регистров класса Generator.

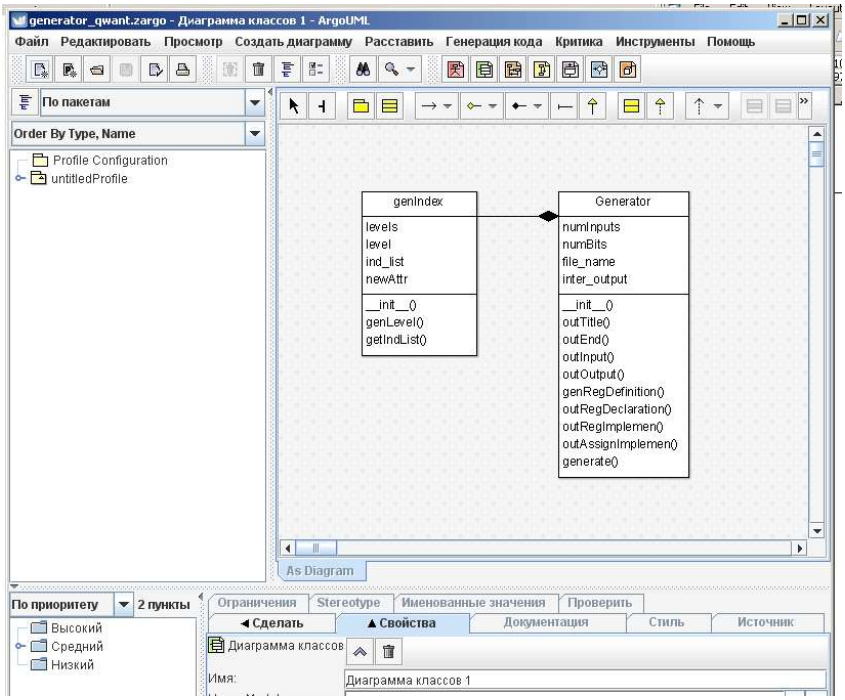



Рис. 4. Структура классов программы (ArgoUML)

Для реализации каждой части модели создан отдельный метод класса Generator, которые описаны в табл. 2. Класс Generator содержит поля: numInputs – число входов модели, numBits – разрядность входных/выходных векторов, file_name – имя файла модели, inter_output – позволяет при необходимости отключать генерирование выходов признаков для регистров внутренних уровней. Метод инициализации переменных задает значения переменных по умолчанию (листинг 5): numInputs = 4, numBits = 8, file_name = “model.v”, inter_output = True. Любой из параметров может быть изменен при создании экземпляра класса Generator, например следующим образом:

numInputs = 16

numBits = 8

obj1 = Generator(numInputs, numBits, “model”+str(numInputs)+”.v”, False).

Таблица 2

Методы класса Generator поведенческой модели

| Метод | Описание |
|---|---|
| outTitle() | Генерация заголовка |
| outInput() | Генерация деклараций входов |
| outOutput() | Генерация деклараций выходов |
| outRegDeclaration() | Генерация деклараций переменных для описания регистров |
| outRegImplemen() | Генерация операторов, реализующих регистры |
| outAssignImplemen() | Генерация операторов assign для формирования признаков уровней и присвоения значений выходов |
| outEnd() | Генерация завершающей части кода модели.Завершение |
| genRegDefinition(name1, name2, vec = True) | Вызывается из outRegDeclaration() |
| generate() | Является основным методом класса, который, используя остальные методы класса, выполняет генерацию целой модели устройства и запись ее в файл. |

Листинг 5

Фрагмент класса Generator с функцией инициализации полей class Generator

```
    “””” Model code generator “””
    def __init__(self, numInputs = 4, numBits = 8, file_name = “model.v”,
                 inter_output = True):
        self.numInputs = numInputs
        self.numBits = numBits
        self.file_name = file_name
        self.inter_output = inter_output
        if self.numInputs <= 4 :
            self.coef = 2
        elif self.numInputs <= 8 :
            self.coef = 3
        elif self.numInputs <= 16 :
            self.coef = 4
```

Класс Generator содержит также функции outTitle() и outEnd(), которые открывают и закрывают внешний файл (листинг 6).

Листинг 6

Функции outTitle() и outEnd()

```
    def outTitle(self):
        self.f = open(self.file_name, ‘w’)
        self.f.write(“module device\n  (input clk, rst,\n”)
    def outEnd(self):
    self.f.write(“\nendmodule”)
    self.f.close()
```

Рис. 5 представляет полную структуру классов программы генератора поведенческой модели процессора, включающую реализацию двух классов Generator и genIndex, а также вспомогательные функции для подготовки и формирования данных, работы со списками индексов регистров каждого уровня и признаков полноты покрытия.

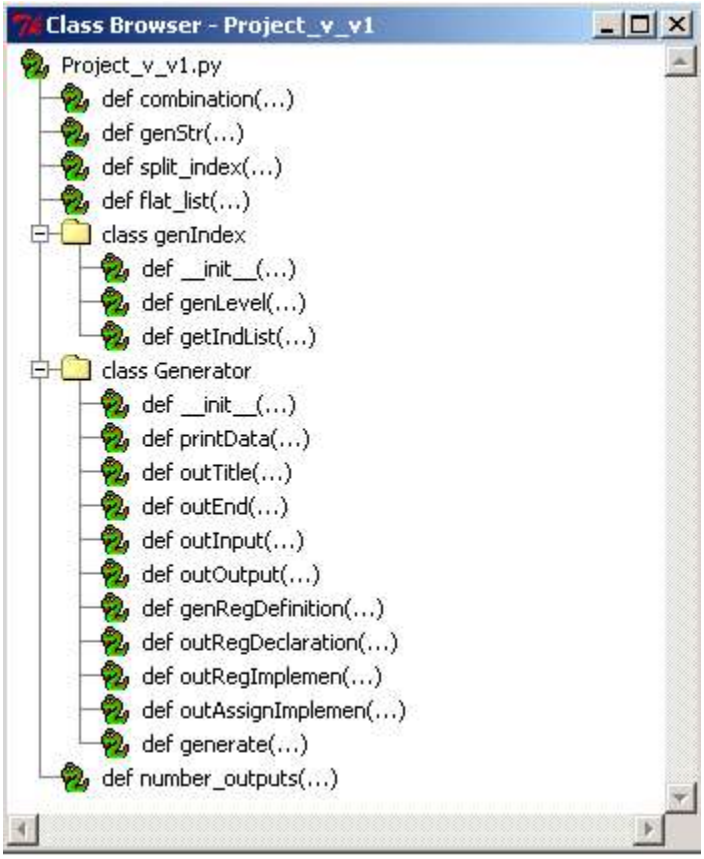



Рис. 5. Браузер классов из программы Python 2.7 Shell

Генерация индексов регистров и переменных является достаточно сложной задачей и для этой цели создан отдельный класс genIndex (листинг 7). Для генерации индексов используются списки, принадлежащие языку Python. Класс genIndex содержит поля: levels – число уровней модели, которое соответствует количеству векторов входных данных; level – уровень, для которого генерируются индексы, ind_list – список индексов. Метод genLevel() генерирует список индексов, а метод getIndList() – возвращает список индексов.

Листинг 7

Класс генератора индексов переменных

```
class genIndex:
   “””” generate list of index for the level “””
   def __init__(self, levels = 16, level = 1):
      self.levels = levels
      self. levels = level
      self.ind_list = []
       self.genLevel()
   def genLevel(self):
      for i in xrange(self.levels):
         self.ind_list.append([str(i)])
      if self.level > 1:
          list_t = []
          ii=0
```

```
            for i1 in range(self.levels-1):
                ii += 1
                t=[]
               for i2 in xrange(ii, self.levels):
                    t.append(str(i1)+»_»+str(i2))
                #print t
                list_t.append(t)
           #print «list_level 1»,list_t
           self.ind_list = list_t
       if self.level > 2:
           iLevel = 3
           while iLevel <= self.level:
               list_t2 = []
               list_t.pop(0)
               i=0
              while (len(list_t) > 0):
                   t = []
                  for i1 in xrange(len(list_t)):
                      list_i1 = list_t[i1]
                      for i2 in xrange(len(list_i1)):
                          s = str(i) + «_» + list_i1[i2]
                           t.append(s)
                  i += 1
                  list_t.pop(0)
                  list_t2.append(t)
             #print «list_level «,iLevel, «:», list_t2
              list_t = list_t2
              iLevel += 1
          self.ind_list = list_t2
    def getIndList(self):
        return self.ind_list
```

В табл. 3 показано, как увеличивается число внешних линий устройства с увеличением числа входов (numInputs). При этом видно, что разрядность входных сигналов (numBits) несущественно влияет на общее число входов и выходов модели.

Таблица 3

Зависимость количества внешних линий модели от числа входов (numInputs) и их разрядности (numBits)

| numInputs | numBits | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 |
| 4 | 31 | 55 | 79 | 103 | 127 | 151 | 175 | 199 |
| 6 | 83 | 115 | 147 | 179 | 211 | 243 | 275 | 307 |
| 8 | 279 | 319 | 359 | 399 | 439 | 479 | 519 | 559 |
| 10 | 1 051 | 1 099 | 1 147 | 1 195 | 1 243 | 1 291 | 1 339 | 1 387 |
| 12 | 4 127 | 4 183 | 4 239 | 4 295 | 4 351 | 4 407 | 4 463 | 4 519 |
| 14 | 16 419 | 16 483 | 16 547 | 16 611 | 16 675 | 16 739 | 16 803 | 16 867 |
| 16 | 65 575 | 65 647 | 65 719 | 65 791 | 65 863 | 65 935 | 66 007 | 66 079 |
| 18 | 262 187 | 262 267 | 262 347 | 262 427 | 262 507 | 262 587 | 262 667 | 262 747 |

С увеличением числа входных векторов значительно увеличивается размер модели, а следовательно, и количество кода для ее реализации, в результате обработка таких моделей системами синтеза и имлементации занимает очень много времени. Подобные программы лучше работают с кодом, состоящим из нескольких модулей. В целях повышения скорости выполнения операций синтеза и имплементации разработана структурная модель процессора, в котором отдельный узел реализован в виде отдельного модуля – элементар-

ной ячейки. Кроме того, в эту модель добавлен вход адреса, позволяющий выборочно считывать информацию с узлов-признаков, что дало возможность существенно сократить число внешних выходов устройства.

2.2. Генератор структурной модели процессора

Структурная Verilog–модель состоит из областей, которые представлены на рис. 6. Вместо области реализации регистров и присвоения значений выходов, как было в поведенческой модели, эта модель содержит описание реализации регистров 1-го уровня с помощью конструкции always и реализацию регистров остальных уровней процессора, включая оператор реализации копии модуля элементарной ячейки, оператор assign, описывающий тристабильный буфер для подключения выхода признака и условный оператора реализации мультиплексора выбора ячейки. Программа генератора усложняется за счет добавления нескольких методов (структура программы из отладочного средства для Python 2.7 представлена рис. 7). Описание методов класса генератора включены в табл. 4. Программа содержит 380 строк кода.

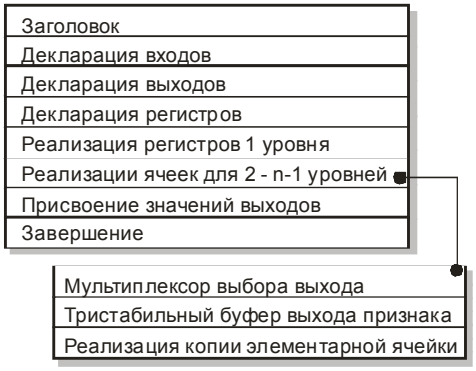



Рис. 6. Организация Verilog-модели

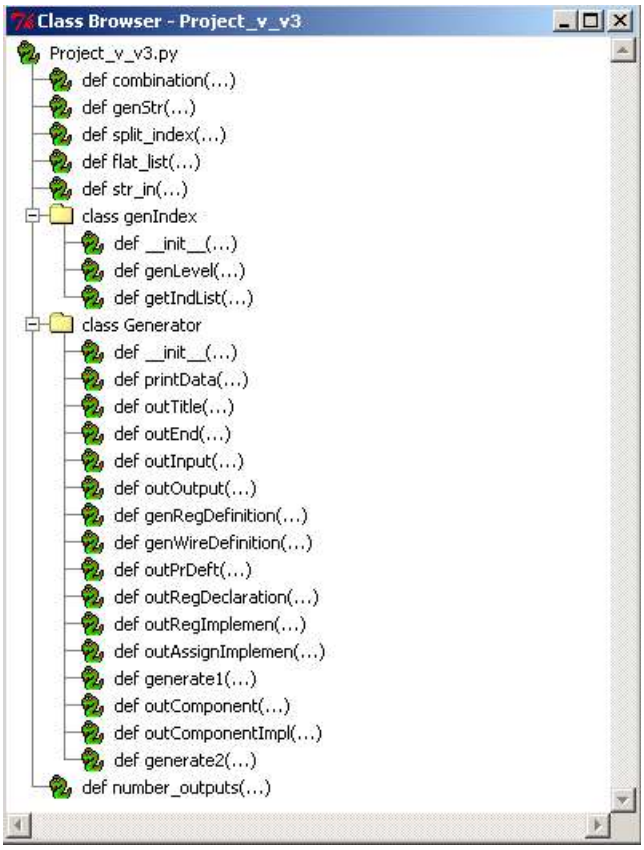



Рис. 7. Структура программы генератора в браузере классов из программы Python 2.7 Shell

Присваиваемый каждой ячейке адрес генерируется на основе строкового представления ее индекса с помощью функции str_in (листинг 8).

Листинг 8

Функция генерации адреса ячейки

```
def str_in(l0, step):
    “””” Form integer address value based on element index”””””
    i = 0
    l = l0.split(“_”)
     l.reverse()
    for j in xrange(len(l)):
        i *= 2**step
       i += int(l[j])
       #print ‘i’, i
     return i
```

Таблица 4

Методы класса Generator поведенческой модели

| Метод | Описание |
|---|---|
| outTitle() | Генерация заголовка |
| outInput() | Генерация деклараций входов |
| outOutput() | Генерация деклараций выходов |
| outRegDeclaration() | Генерация деклараций переменных для описания регистров |
| outRegImplemen() | Генерация операторов, реализующих регистры |
| outAssignImplemen() | Геренация операторов assign для формирования признаков уровней и присвоения значений выходов |
| outEnd() | Генерация завершающей части кода модели.Завершение |
| genRegDefinition(name1, name2, vec = True | Декларация регистровых переменных, вызывается из функции outRegDeclaration() |
| genWireDefinition() | Генерация переменных типа данных wire класса цепи, моделирующих тристабильны е буферы |
| OutComponent() | Генерирует файл с моделью элементарной ячейки процессора |
| OutComponentImpl() | Генерирует оператор реализации копии модуля элементарной ячекки процессора |
| printData() | Вывод числа входов процессора и их разрядности |
| generate2() | Является основным методом класса, который, используя остальные методы класса, выполняет генерацию целой модели устройства и запись ее в файл. |

Несмотря на то, что python – это интерпретируемый язык программирования, для программ, созданных с помощью этого языка, в случае необходимости можно легко получить исполняемый файл, используя средства PyInstaller (http://www.pyinstaller.org/) или py2exe (http://www.py2exe.org/), или подобные им. Конвертор PyInstaller создает независимые исполняемые программы для операционных систем Windows, Linux, Mac OS X, Solaris и AIX, в то время как py2exe реализует только Windows приложения.

2.3. Графический интерфейс пользователя генератора

Для оформления графического интерфейса пользователя была применена открытая библиотека tkinter языка python, которая является де-факто стандартом для решения подобных задач. Доступность, переносимость, простота получения, документированность и наличие расширений определяют популярность tkinter для формирования GUI в Python-приложениях в течение многих лет: В отличие от более сложных систем, библиотека tkinter позволяет сразу же приступить к работе с ней, без необходимости предварительно осваивать крупные модели взаимодействия классов. Несмотря на простоту прикладного интерфейса библиотеки tkinter, она дает возможность добавлять новые виджеты, написанные на языке Python, или подключать дополнительные расширения, такие как Pmw, Tix и ttk. Важным преимуществом является переносимость. Сценарий на языке Python, в котором графический интерфейс строится с помощью библиотеки tkinter, будет работать без изменений на всех основных современных оконных платформах: Microsoft Windows, X Window (в Unix и Linux) и Macintosh OS X. Более того, внешний вид созданного интерфейса будет привычен для пользователей каждой из этих платформ. Эта особенность развивалась по мере того как библиотека Tk (на которой основана tkinter) становилась все более зрелой. Графический интерфейс, реализованный сценарием Python/tkinter, в Windows выглядит так, как должен выглядеть интерфейс программы для Windows; в Unix и Linux обеспечивает такое же взаимодействие и на Mac он выглядит так, как должна выглядеть программа Mac.

Библиотека tkinter является модулем стандартной библиотеки Python, поставляемой вместе с интерпретатором. Более того, в большинство пакетов установки Python (включая стандартный пакет установки Python для Windows, Mac и большинство дистрибутивов Linux) уже включена поддержка tkinter. Благодаря этому сценарии, написанные с использованием модуля tkinter, сразу могут работать с большинством интерпретаторов Python, не требуя дополнительных действий по установке. Поскольку задействованная в ней библиотека Tk используется также языками программирования Tcl и Perl (и многими другими), ей уделяется больше внимания и усилий разработчиков, чем другим имеющимся инструментариям.

Окно графического интерфейса генератора представлено на рис. 8. В нем перед генерацией Verilog-кода можно задавать количество входов Number of inputs (число входных векторов), разрядность входных векторов Number of bits, имя файла и путь для сохранения создаваемой модели File name.

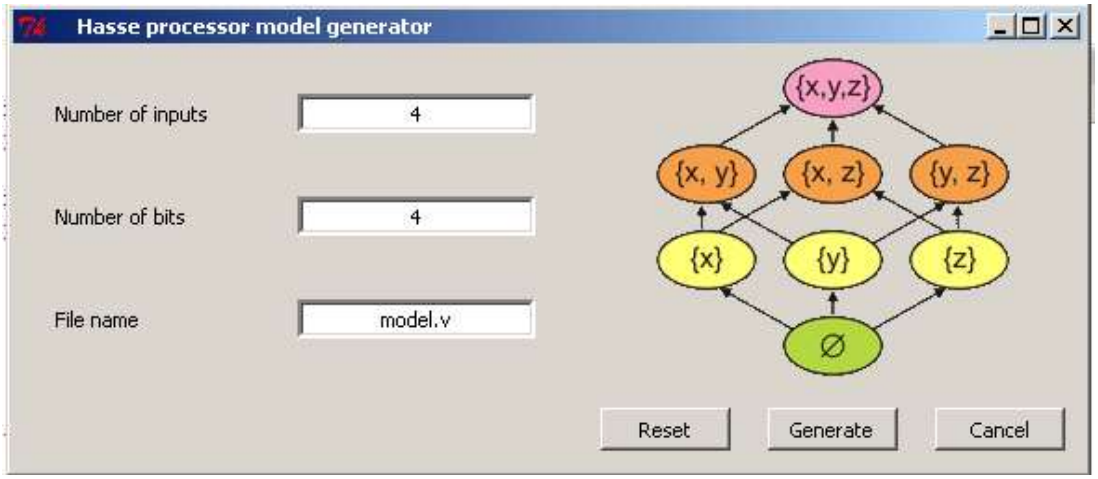



Рис. 8. Окно графического интерфейса для Windows

По умолчанию число входов и их разрядность равна 4, генерируемая модель сохраняется в файл model.v. Кнопка Reset сбрасывает введенные значения параметров к заданным по умолчанию, Generate – генерирует Verilog-модель процессора, Cancel – закрывает окно генератора. В программе использовались экземпляры классов метки Label, поля ввода Entry, кнопки Button, PhotoImage для вывода изображения. Все классы принадлежат библиотеке Tkiter, которая в свою очередь основана на библиотеке Tk():

```
import sys
from Tkinter import *
```

```
root = Tk()
root.title(“     Hasse processor model generator”)
```

Для упорядочивания графического отображения объектов классов в окне генератора используется метод сетки – grid. Следующий фрагмент кода представляет создание метки и поля ввода для ввода значения параметра Number of inputs и размещение его в нулевой строке и колонке сетки:

```
l1 = Label(root, text = ‘Number of inputs’)
l1.grid(row = 0, column = 0, padx = 20, pady = 20, sticky=W)
num_levels = IntVar(value = 4)
i1 = Entry(root, justify = “center”, width = 20, textvariable = num_levels)
i1.config(bd = 3, relief = SUNKEN)
i1.grid(row = 0, column = 1)
```

### 3. Синтез и имплементация

Реализация устройства на микросхемах FPGA фирмы Xilinx Spartan 3E (xc3s1600e-4-fg320) и Virtex 5 xc5vlx50. Для синтеза и имплементации использовался программный пакет фирмы Xilinx ISE Design Suite 11.1. Модель устройства, с конфигурацией 10 входов, разрядность входов 8 бит, реализована с использованием микросхемы фирмы Xilinx серии Spartan-3E: xc3s1600e. При этом полученная схема содержит 9 247 триггера FF и 18 062 4-х входовые таблицы преобразования LUT, что составляет 31 и 61% доступных ресурсов микросхемы:

Logic Utilization:

Number of Slice Flip Flops: 9,247 out of 29,504 31%

Number of 4 input LUTs: 18,062 out of 29,504 61%.

Исходя из результатов статического временного анализа (табл. 5), минимальный рабочий период синхросигнала равен 41 ns, что соответствует частоте 24 МГц.

Таблица 5

Временные параметры устройства

Clock clk to Pad

| Destination | clk (edge) to PAD | Internal Clock(s) | Clock Phase |
|---|---|---|---|
| o2 | 13.343(R) | clk_BUFGP | 0.000 |
| o3 | 15.381(R) | clk_BUFGP | 0.000 |
| o4 | 16.611(R) | clk_BUFGP | 0.000 |
| o5 | 19.541(R) | clk_BUFGP | 0.000 |
| o6 | 20.813(R) | clk_BUFGP | 0.000 |
| o7 | 15.613(R) | clk_BUFGP | 0.000 |
| o8 | 12.385(R) | clk_BUFGP | 0.000 |
| o9 | 13.186(R) | clk_BUFGP | 0.000 |
| o_pr<0> | 40.912(R) | clk_BUFGP | 0.000 |
| o_pr<1> | 37.572(R) | clk_BUFGP | 0.000 |
| o_pr<2> | 37.575(R) | clk_BUFGP | 0.000 |
| o_pr<3> | 37.859(R) | clk_BUFGP | 0.000 |
| o_pr<4> | 38.169(R) | clk_BUFGP | 0.000 |
| o_pr<5> | 37.284(R) | clk_BUFGP | 0.000 |
| o_pr<6> | 38.165(R) | clk_BUFGP | 0.000 |
| o_pr<7> | 37.858(R) | clk_BUFGP | 0.000 |
| out<0> | 7.854(R) | clk_BUFGP | 0.000 |
| out<1> | 8.086(R) | clk_BUFGP | 0.000 |
| out<2> | 8.088(R) | clk_BUFGP | 0.000 |
| out<3> | 8.057(R) | clk_BUFGP | 0.000 |
| out<4> | 7.534(R) | clk_BUFGP | 0.000 |
| out<5> | 7.465(R) | clk_BUFGP | 0.000 |
| out<6> | 8.073(R) | clk_BUFGP | 0.000 |
| out<7> | 8.032(R) | clk_BUFGP | 0.000 |
| p_o10<0> | 7.716(R) | clk_BUFGP | 0.000 |

Clock to Setup on destination clock clk

| Source Clock | Src:Rise\|Dest:Rise | Src:Fall\|Dest:Rise | Src:Rise\|Dest:Fall | Src:Fall\|Dest:Fall |
|---|---|---|---|---|
| clk | 4.862 | | | |

Для вычисления временных параметров используются результаты статического анализа, выполняемые программным обеспечением на этапе Place&Route. Система вычисляет задержки путей между регістрами (Tclk_to_clk) и от синхровхода до выхода (Tclk_to_pad). Максимальный из этих параметров определяет минимальный рабочий период синхросигнала, который может быть подан на вход устройства.

Period = max{Tclk_to_clk , Tclk_to_pad_max };

Например, для временных параметров:

Tclk_to_clk = 4.862 ns

Tclk_to_pad_max = 40.912ns

Минимальный рабочий период синхросигнала будет равен Period = 40.912 ns, что соответствует частоте Fclk = 24 МГц.

**Выводы**

Представлена практическая реализация генератора HDL-кода специализированных процессоров, использующих диаграммы Хассе для параллельных векторно-логических вычислений булеанов, применяемых для ускорения моделирования, верификации и диагностирования. Программно-аппаратная реализация процессора основана на использовании языков программирования: C++, Verilog, Python 2.7 и платформ: Microsoft Windows, X Window (в Unix и Linux) и Macintosh OS X. Генератор дает возможность автоматически синтезировать HDL-коды процессорной структуры от 1 до 16 двоичных разрядов для параллельной обработки соответствующего количества входных векторов или слов. Теоретические и практические результаты дают возможность за счет аппаратной и структурной избыточности на 50% – быстродействие интерпретативного моделирования, на 5% повысить выход годной продукции, на 12% – глубину диагностирования неисправных функциональных блоков и на 15% уменьшить время отладки HDL-кода в процессе проектирования цифровых систем на кристаллах.

**Список литературы: 1.** *Michael A. Nielsen & Isaac L. Chuang.* Quantum Computation and Quantum Information. Cambridge University Press. 2010. 676p. **2.** *Stig Stenholm, Kalle-Antti Suominen.* Quantum approach to informatics. John Wiley & Sons, Inc., 2005. 249p. **3.** *Mark G. Whitney.* Practical Fault Tolerance for Quantum Circuits. PhD dissertation. University of California, Berkeley. 2009. 229p. **4.** *Mikio Nfrfhara.* Quantum Computing. An Overview. Higashi-Osaka. Kinki University, 2010. 53p. **5.** *Горбатов В.А.* Основы дискретной математики. М.: Высш. шк. 1986. 311 с. **6.** *Hahanov V.I., Litvinova E.I., Chumachenko S.V., Baghdadi Ammar Awni Abbas, Eshetie Abebech, Mandefro.* Qubit Model for solving the coverage problem // Proc. of IEEE East-West Design and Test Symposium. IEEE. USA. Kharkov. 14-17 September 2012. P.142 - 144.

*Поступила в редколлегию 11.06.2014*

**Хаханов Владимир Иванович**, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. IEEE Senior Member. IEEE Computer Society Golden Core Member. Научные интересы: проектирование и тестирование вычислительных систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, теннис, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: hahanov@kture.kharkov.ua.

**Хаханова Ирина Витальевна**, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование цифровых систем и сетей на кристаллах. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: hahanova@mail.ru.

**Tamer Bani Amer,** аспирант кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование вычислительных систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: hahanov@kture.kharkov.ua.

**Dahiri Farid (Дахири Фарид),** магистрант кафедры АПВТ ХНУРЭ. Научные интересы: компьютерная инженерия, методы оптимизации, программирование. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: dahiri.farid@gmail.com.

## РЕФЕРАТИ


УДК 681.326:519.713

**Імплементація кубітних структур даних в паралельний обчислювач /** В.І. Хаханов, І.В. Хаханова, Tamer Bani Amer, Farid Dahiri (Фарід Дахірі) // АСУ та прилади автоматики. 2014. Вип. 167. С. 4-16.

Запропонована практична реалізація генератора HDL-коду «квантових» процесорів, що використовують діаграми Хассе для паралельних векторно-логічних (теоретико-множинних) обчислень булеанів, що застосовуються для прискорення вирішення завдань, моделювання, верифікації, діагностування. Програмно-апаратна реалізація процесора ґрунтується на використанні мов програмування: C ++, Verilog, Python 2.7 і платформ: Microsoft Windows, X Window (в Unix і Linux) і Macintosh OS X. Генератор HDL-коду дає можливість автоматично синтезувати HDL-коди процесорної структури від 1 до 16 двійкових розрядів для паралельної обробки відповідної кількості вхідних векторів або слів. Верифікація HDL-коду процесора виконується на тестових прикладах задачі покриття, що використовують дві стратегії оптимізації: реверсивний алгоритм для усунення надмірності і розбиття матриці покриттів на частини з метою їх подальшої паралельної обробки процесорами Хассе.

Табл. 5. Іл. 8. Бібліогр.: 6 назв.

UDC 681.326:519.713

**Implementation of qubit data structures in parallel computer** / V.I. Hahanov, I.V. Hahanova, Tamer Bani Amer, Farid Dahiri // Management Information System and Devices. 2014. N 166. P.4-16.

A practical implementation of the HDL-code generator for "quantum" processors is proposed; it is based on the use of the Hasse diagram for parallel vector-logic (set-theoretic) calculations of the power set, applied to accelerate solving the problems of modeling, verification, diagnosis. Software-hardware implementation of the processor is based on the use of
programming languages C++, Verilog, Python 2.7 and the following platforms: Microsoft Windows, X Window (on Unix and Linux) and Macintosh OS X. HDL-code generator makes it possible to automatically synthesize the HDL-code of the processor structure from 1 to 16 bits for parallel processing a corresponding number of input vectors or words. Verification of the processor HDL-code is performed by using the test patterns of the coverage problem based on two optimization strategies: a reversible algorithm to eliminate redundancy and partitioning the coverage matrix for subsequent parallel processing by Hasse processors.

Tab. 5. Fig. 8. Ref.: 6 items.

---

УДК 044.03;658.11.05.06

**Архітектура інформаційно-аналітичної системи управління безпекою виробництва**/ Н.Н. Сердюк / / АСУ та прилади автоматики. 2014. Вип. 167. С. 17-22.

Головною проблемою, що утрудняє розробку процесних моделей управління безпекою процесів підприємства, є відсутність системного підходу до процесного опису задач управління безпекою діяльності співробітників підприємства. Запропонована структурна схема ІАС і узагальнена концептуальна схема сховища даних, які дозволяють сформувати набір специфікацій на розробку забезпечуючої частини ІАС управління безпекою виробництва, настройка якої на конкретний тип виробництва вимагає мінімальних витрат.

Іл. 4. Бібліогр.: 6 назв.

UDC 044.03;658.11.05.06

**Architecture of the informatsionno-analiticheskoy system of management by safety of production** / N. N. Serdyuk // Management Information System and Devices. 2014. N 166. P.17-22.

By a main problem hampering development of protsessnih case frames by safety of processes of enterprise, there is absence of systems approach to protsessnomu description of tasks of management by safety of activity of employees of enterprise. The offered flow diagram IAS and generalized conceptual chart of depository of data allow to form the set of specifications for development of providing part of the IAS management by safety of production, tuning of which on the concrete type of production requires minimum expenditures.

Fig. 4. Ref.: 6 items.

УДК 62-519:681.5

**Метод зниження інтенсивності відеопотоку в інфокомунікаційних системах** / В.В. Бараннік, О.Ю. Отман Шаді, Г.В. Хаханова // АСУ та прилади автоматики. 2014. Вип. 167. С. 23-27.

Показана актуальність питань, пов'язаних з підвищенням якості представлення відоеінформаційних послуг з використанням бездротових телекомунікаційних технологій. Обґрунтована важливість зниження напруги на мережу на основі зниження інтенсивності стисненого відеопотоку. Виявлені уразливі сторони функціонування стандартизованих технологій обробки базових кадрів. Розглянуті відмінні етапи технології кодування базових кадрів. Викладені основні етапи оцінки інтенсивності потоку, що припадає на базовий кадр, з урахуванням формування кодових конструкцій стисненого представлення трансформант на основі діагонально-нерівномірного позиційного кодування. Здійснена розробка методу для оцінки інтенсивності на групу кадрів і всього відеопотоку з урахуванням: диференційованого вкладу типів кадрів в інтенсивність і якість візуального сприйняття реконструйованого відеопотоку; компресії базових кадрів на основі їх трансформування та подальшого діагонально-нерівномірного позиційного кодування.

Табл. 1. Іл. 1. Бібліогр.: 6 назв.

UDC 62-519:681.5

**Method of intensity lowering the video stream in infocommunication** / V. Barannik, O. Otman Shadi, A. Hahanova // Management Information System and Devices. 2014. N 166. P. 23-27.

Relevance of the questions connected with improvement of quality of video information provision services with use of wireless telecommunication technologies is shown. Importance of network load lowering on the basis of intensity lowering of an oblate video stream is justified. The vulnerable sides of standardized technologies functioning of processing of basic frames is identified. Distinctive stages of technology of basic frames coding for intensity lowering of their code representation are considered. The main evaluation stages of intensity of the flow falling on a basic frame taking into account formation of code constructions of transforms oblate representation on the basis of diagonal and non-uniform positional coding are explained. Development of a method for an intensity assessment on group of frames and all video streams is carried out taking into account: a differentiated contribution of frames types to intensity and quality of visual acceptability of the reconstructed video stream; compressions of basic frames on the basis of their transformation and the subsequent diagonal and non-uniform positional coding.

Tab. 1. Fig. 1. Ref.: 6 items.

---

УДК 004.942: 004.89

**Представлення образів динамічних процесів в системах автоматичного керування за допомогою самоналагоджувальних агентів** / І.С. Конох // АСУ та прилади автоматики. 2014. Вип. 167. С. 28-37.

Розглянуто питання побудови універсальної моделі програмного агента, що здатен утворювати самоналагоджувальну структуру для запам'ятовування часових образів сигналів систем автоматичного керування. На основі подібних структур можливо визначати ступінь схожості поточної ситуації з будь-якої, що відбувалася в минулому, та прогнозувати подальшу поведінку об'єкту керування з використанням функцій узагальнення. Показана можливість директивного внесення в мультиагентну систему апріорних знань. Запропоновано UML-діаграми програмного агента з самоналаштуванням, що побудовані за принципами об'єктно-орієнтованого програмування.

Іл. 9. Бібліогр.: 5 назв.

UDC 004.942: 004.89

**A dynamic process images presentation in the system automation by means of a self-tuning agents** / I.S. Konokh //Management Information System and Devices. 2014. N 166. P. 28-37.

The article deals with the issue of building a universal model of software agent that is capable of forming self-tuning structure for storing temporary images of automatic control systems signals. On the basis of such structures may determine the degree of similarity of the current situation from any that happened in the past and predict the future behavior of object management functions using generalization. The possibility of policy making in Multi-agent system of apriority knowledge. Self-adjusting agent software UML-diagrams built on the principles of object-oriented programming.

Fig. 9. Ref.: 5 items.

УДК 629.391

**Метод динамічного управління бітовою швидкістю відеопотоку із використанням тривимірного представлення трансформант** / В.В. Бараннік, Д.Е. Двухглавов, В.В. Твердохліб // АСУ та прилади автоматики. 2014. Вип. 167. С.37-43.

Розглянута актуальність досліджень, спрямованих на зменшення бітової швидкості обробки кадрів відеопотоку для підвищення якості надання послуг відеосервісу для енергоефективних інфокомунікаційних систем. Запропновано представлення блоків відеокадру у вигляді бітового кубу та принципи організації управління передачею кадру для такого представлення, які викладені у вигляді готових для реалізації алгоритмів обробки кадру на стороні, яка передає та яка приймає. Також розглянуто підхід для вибору оптимальних параметрів передачі блоків на основі принципів динамічного програмування.

Іл. 3. Бібліогр.: 4 назви.

УДК 629.391

**Method for dynamic management of the bit rate of the video stream using a three-dimensional presentation of transformant** / V. Barannik, D.Dvukhglavov, V. Tverdokhleb //Management Information System and Devices. 2014. N 166. P. 37-43.

The fctuality of research aimed at reducing the bit rate processing frames of the video stream to improve of service video service provision the quality for energy-efficient infocommunication systems has considering. Invited presentation blocks of the video frame as a bitmap cube and principles of the transmission control frame for such representations which are stated in the form of ready to implement the algorithms of frame processing to the transmitting and receiving side. Also the approach for selection of optimal parameters of transfer units based on the principles of dynamic programming.

Fig. 3. Ref.: 4 items.

---

УДК 519.6

**Розв'язання детермінованої задачі оптимізації трирівневої мережі поставок одного товару** / М.В. Новожилова, І.В. Штань // АСУ та прилади автоматики. 2014. Вип. 167. С. 44-50.

Розглянуто побудову методу розв'язання задачі оптимізації трирівневої мережі поставок одного товару на основі визначення характеристик рівноваги ринку за умови можливості подання функцій, що описують детерміновану поведінку основних суб'єктів ринку поліномами другого порядку через визначення сідлової точки відповідно до побудованої функції Лагранжу.

Іл.2. Бібліогр.: 11 назв.

UDC 519.6

**Solution approach for an one-product supply chain optimization problem** / M.V. Novozhylova, I.V. Shtan' //Management Information System and Devices. 2014. N 166. P.44-50.

It is proposed construction and analysis of solution method for an one-product supply chain optimization problem based on defining equilibrium market parameters due to definition of saddle point for appropriate Lagrangian taking into account presentation of functions being considered as polunoms having second order.

Fig. 2. Ref.: 11 items.

---

УДК 004.627(043.2)

**Методика оцінки ефективності технології стиснення зображень на базі методів інваріантно-просторового кодування по швидкодії** / О.К. Юдін, К.О. Курінь, Р.В. Зюбіна // АСУ та прилади автоматики. 2014. Вип. 167. С.51-58.

Проведено аналіз технологій обробки і передачі даних в інформаційно-телекомунікаційних системах (ІТКС). Визначено основні показники ефективності функціонування ІТКС. Визначено роль і місце алгоритмів стиснення зображень для скорочення часу доставки даних в ІТКС. Створено методику оцінки ефективності технології стиснення на базі методу инваріантно-просторового кодування по швидкодії. Проведено розрахунок тимчасових витрат на обробку тестових зображень кодеком на базі инваріантно-просторового кодування.

Табл. 6. Іл. 5. Бібліогр.: 9 назв.

UDC 004.627(043.2)

**Technique of an assessment of image compression technology of methods of invariant and spatial coding on high-speed performance** / O.K. Judin, K.O. Kurin', R.V. Zjubina //Management Information System and Devices. 2014. N 166. P. 51-58.

The analysis of technologies of data processing and transmission is information telecommunication systems (ITCS) provided in. The basic metrics of efficiency of ITCS functioning are defined. The role and place of algorithms of images compression in reduction of information delivery time in ITCS is grounded. The expedience of increase of compression coefficient and development of new technologies and algorithms of images compression is grounded. The estimation of the developed technology of images compression on the base of method of invariant spatial coding comparing to the case of the use of JPEG algorithm for images processing by the value of data delivery time in ITCS is provided. The method of estimation of compression technology efficiency on the base of the method invariant spatial coding by the high-speed performance is created. The analytical model of images calculation processing procedure of time for the technology of compression on the base of method of invariant spatial coding is created. The calculation of time spent on processing of test images with the codec on a base of invariant spatial coding is provided. Comparative analysis of the offered technology of images compression by the high-speed performance with codecs on the base of JPEG algorithm is provided. That results allow to do next conclusions: use of the developed method on a base of the method invariant spatial coding for images processing allows to decrease the time of delivery compared with the case of the use of JPEG algorithm on a value which changes in the range of from 2 to 32 ms depending on the size of image and provides the increase of efficiency in a range from 1% to 10%, that proves the expedience of the use of the developed method of image compression in data transmission channels.

Tab. 6. Fig. 5. Ref.: 9 items.

УДК 567.456

**Метод обробки відеоресурсів із збереженням цілісності в інформаційних системах** /Ю.Н. Рябуха //АСУ та прилади автоматики. 2014. Вип. 167. С. 59-64.

Показано, що існуючі технології обробки відеоданих не задовольняють потребам сучасного інформаційного простору. Обгрунтована необхідність вдосконалення теоретичної бази і технологій обробки відеоданих (кадрів, відеопотоку) в напрямку формування кодів для тривимірних структур даних. Розробляється тривимірне кодування даних в режимі рівномірних тривимірних чисел та змінної довжини кодового слова на представлення їх кодового значення.

Іл.1. Бібліогр.: 11 назв.

UDC 567.456

**Processing method video data saving integrity in information systems**/Yu.N.Ryabuha//Management Information System and Devices. 2014. N 166. P. 59-64.

Is shown that the existing technologies of video processing don't satisfy to needs of the modern information space. Reasons for need of enhancement of theoretical basis and technologies of video processing (frames, a video stream) in the direction of formation of codes for three-dimensional data structures are explained. Three-dimensional coding of the uniform three-dimensional polyadic numbers this in the mode and variable length of the code word on representation of their code value is developed. The created coding provides creation of compact representation of video data in three-dimensional space for cases, when: there are no rigid restrictions on codegram length; formation of length of the code word depends on dynamically changing characteristics on a computing resource; for transmission on communication links a packing of variable length is used; the code shall be built for assigned amount of elements in a frame or in a flow. Compression is provided for the score an exception of the structural redundance caused by limitation and non-uniformity of dynamic ranges of elements of video data at the same time on three coordinates of three-dimensional data structures.

Fig. 1. Ref.: 9 items.