ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МИКРОПРОГРАММНОГО КОНСТРУИРОВАНИЯ МАСШТАБИРУЕМЫХ ЦИФРОВЫХ АНАЛОГОВ НА АССОЦИАТИВНОЙ ПАМЯТИ ДЛЯ САУ БПЛА

Добротворский А.С.

АНО «Институт Поведения», Россия, г. Москва, ул. Новый Арбат, д.21

dobrotvorskiy@gmail.com

Аннотация: Раскрыты особенности построения и использования инструментальных средств микропрограммного конструирования контроллеров на ассоциативной памяти, которые позволяют решать на единой импортонезависимой аппаратной платформе широкий спектр задач САУ БПЛА.

Ключевые слова: бортовые системы искусственного интеллекта, системы автоматического управления летательных аппаратов, временно́е масштабирование, ассоциативная память, цифровые аналоги, объектно-ориентированное микропрограммное конструирование.

Введение

Рост уровня интеллектуализации задач, решаемых беспилотной авиацией, с одной стороны значительно ужесточил требования к тактовой частоте работы систем автоматического управления (САУ БПЛА), а с другой – увеличил разрыв между скоростями работы приводов и электронной компонентной базы, неотъемлемой частью которой являются контроллеры, обеспечивающие сопряжение в реальном времени центральных процессоров с датчиками и приводами. На данный момент этот разрыв уже составляет не менее 6 порядков, так как тактовая частота, а с ней и частота цикла исполнения инструкций в современных процессорах (контроллерах) общего назначения находится на уровне 1 ГГц, а максимальная частота спектра сигналов в каналах управления приводами не превышает значений (1–3) кГц.

Такой дисбаланс в скорости работы процессора (контроллера) общего назначения и объекта управления вынуждает использовать многоканальные механизмы буферизации в памяти вычисленных управляющих данных и временно́го (таймерного) масштабирования их потоков, которые обеспечивают согласование темпов формирования и реализации управляющих воздействий в САУ БПЛА.

Альтернативным и апробированным на практике решением проблемы синхронизации фаз выработки и реализации управляющих воздействий в САУ БПЛА служат *цифровые аналоги* [1–3], в которых *алгоритмическая* «сложность» решаемой задачи трансформируется в прямые аппаратные затраты на операционные конвейеры, а временно́е масштабирование осуществляется за счет перехода от *аналитической* (n) к *избыточной* разрядности $m>>n$.

Цель статьи: раскрыть специфику работы инструментальных средств микропрограммного конструирования масштабируемых во времени, отказоустойчивых и защищенных от несанкционированного доступа цифровых аналогов на ассоциативной памяти, комплектуемых импортонезависимой электронной компонентной базой.

1 Технология работы цифровых аналогов на ассоциативной памяти

Цифровые аналоги [1–2] по своей функциональной сути воспроизводят архаичные по современным меркам аналоговые устройства управления, но работают в дискретном времени T и целочисленном операционном базисе. Интерес к ним возродился благодаря широкому освоению ПЛИС-технологий [3] и технологий «процессоров в ассоциативной памяти» [4,5]. Первая критична к несанкционированному вмешательству в САУ БПЛА со стороны «противника», так как базируется на зарубежных СБИС и инструментальных средствах проектирования, а вторая является сугубо отечественной и импортонезависимой, что играет решающую роль в условиях активного противоборства в киберпространстве с технологически развитыми странами.

При реализации цифровых аналогов на *FIFO*-регистровой ассоциативной памяти [4,5] темп реального времени и время задержки определяются разными факторами: первый задается реальным циклом (τ) исполнения каждой инструкции $\tau_I=m\tau$, а второе – «глубиной» конвейера L_d:

$$T_z = \sum_{j=1}^{L_d} \tau_j , \qquad (1)$$

где $\tau_i \leq \tau_I$ – время задержки в j-ом операционном модуле конвейера, а L_d – длина критического пути граф-потока программы из L_p узлов.

Соотношение $\tau_I = m\tau$ справедливо, если аппаратной базой цифрового аналога (рисунок 1) служит *последовательный счетчик*, реализующий функцию цифрового интегратора

$$s(t) = \sum_{i=1}^{n} y_i(T) * 2^{-i}, \qquad (2)$$

в выходной реакции которого только n значащих бит из m возможных.

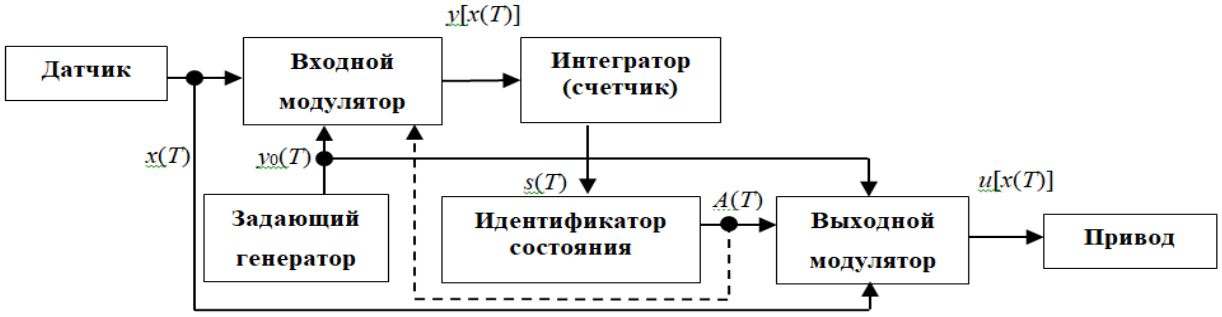


Рис. 1. Типовая схема регулятора, выполненного по схеме «цифрового аналога»

В этом случае на вход n-разрядного счетчика поступают возмущения в виде двоичных последовательностей постоянной «длины» m, из которых до n элементов могут быть «единичными»: $y[x(T)] \in \{0,1,2,...,n\}$ ($y_i \in \{0,1\}, i = \overline{1,m}$), а состояние $s(T) \in \{0,1,2,...,2^{n-1}\}$ ($s_i \in \{0,1\}$) идентифицируется содержимым n значащих разрядов и изменяется в дискретные моменты времени $T=(0,1,2,...)$. Оно зависит *не от структуры*, а от *количества* «*единичных*» бит в двоичной последовательности из m бит, которое формируется модулятором задающей последовательности в зависимости от содержимого входных возмущений $x(T)$ и управления $u[x(T)]$. Это состояние с циклом m тактов трансформируется идентификатором в двоичную *последовательность слов* $\{A(T)\}_N$ с периодом N, каждый элемент которой $A(T) \in \{A(T)\}_N$ ($a_i \in \{0,1\}, i = \overline{1,m}$) может *интерпретироваться* и как число. На выходе цифрового аналога элементы $A(T)$ с циклом m физических тактов трансформируются в целочисленное управление $u[x(T)]: A(T) \rightarrow u(T)$, где каждое «слово» разрядности m имеет только n значащих бит.

2 Термальный синтез функциональных компонент цифровых аналогов

2.1 Синтез счетчиков на FIFO-регистровой ассоциативной памяти

В основу технологии «процессоров в *FIFO*-регистровой ассоциативной памяти» [4,5] положена алогичная по современным меркам концепция, в рамках которой при синтезе не только «более сложных», но и «более простых» функций *неделимой единицей проекта* выступают ячейки такой памяти, а не булевы вентили. Благодаря этому удается автоматизировать процедуру синтеза всех функциональных компонент цифровых аналогов, основу которых составляют счетчики (рисунок 2– счетчик на 4 бита).

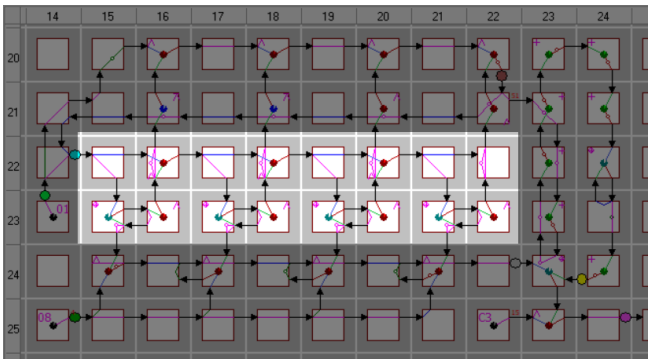


Рис. 2. Топологическая схема счетчика на 4 бита на ассоциативной памяти

Схемотехническое последствие алогичной для современной техники парадигмы синтеза состоит в том, что состояния, отвечающие четным и нечетным тактам, изменяются независимо, что создает эффект параллельной работы двух счетчиков с независимыми потоками аргументов и состояний.

Этот эффект обусловлен тем, что состояние триггеров *на FIFO*-регистровой ассоциативной памяти рисунка 2 фиксируется *двухбитными кольцевыми регистрами*, которые образуют ячейки, расположенные в строке 23 и парах столбцов (15,16), (17,18), (19,20) и (21,22).

Согласно рисунку 2 для синтеза счетчиков произвольной разрядности требуется использовать всего один базовый терм (рисунок 3), в котором в отличие от кремниевых компиляторов параметры задержки модифицируются в зависимости от места расположения в схеме.

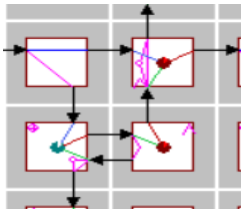


Рис. 3. Базовый терм счетчика на «FIFO-регистровой ассоциативной памяти»

В нижней строке терма расположен триггер, собранный на двух ячейках памяти. Основу его функционирования составляет операция *XOR*, которая выполняется над двумя операндами – флагом переключения состояния, поступающего сверху и флагом текущего состояния, который поступает справа из канала транзита соседнего элемента, который по сути и осуществляет хранение этого значения на предыдущем такте работы. Если флаг переключения состояния имеет значение «ложь», то согласно таблице истинности на правый выход ячейки *XOR* будет выдано текущее состояние, которое, пройдя через канал транзита соседнего элемента, опять окажется на входе *XOR*. Таким образом, импровизированный триггер сохраняет текущее состояние. Если же на вход сверху будет подан сигнал «истина», то операция *XOR* выполнит инверсию текущего хранимого значения и перейдет в новое состояние, которое сохранится до следующего возмущающего сигнала. В верхней строке терма справа находится элемент *AND*, который выполняет операцию побитового «И» между текущим значением данного триггера и значением сигнала переключения. Назначение этой операции – переключить следующий триггер счетчика в новое состояние только в том случае, если состояние текущего триггера «истина», что соответствует распространению единицы переноса в старший разряд при сложении в двоичной арифметике. Генерация счетчика заданной разрядности при помощи такого терма является тривиальной – за каждый разряд счетчика отвечает 1 терм. Таким образом, для реализации 4-х битного счетчика требуется 4 терма, расположенных последовательно слева направо, что видно из рисунка 2.

2.2 Синтез входного модулятора счетчиков на FIFO-регистровой ассоциативной памяти

Для переключения счетчика цифрового аналога с инкрементного на декрементный режим и наоборот, используется следующая схема. Состояния разрядов счетчика по параллельным шинам передают на вход дешифратора сингулярного состояния. В данном случае дешифратор идентифицирует состояние «s_0=s_1=0,s_2=s_3=1» (см. рисунок 4), которое определяет смену инкрементных и декрементных фаз и которое определяется с учетом пространственно-временных характеристик схемы и параметров *n* и *m* цифрового аналога.

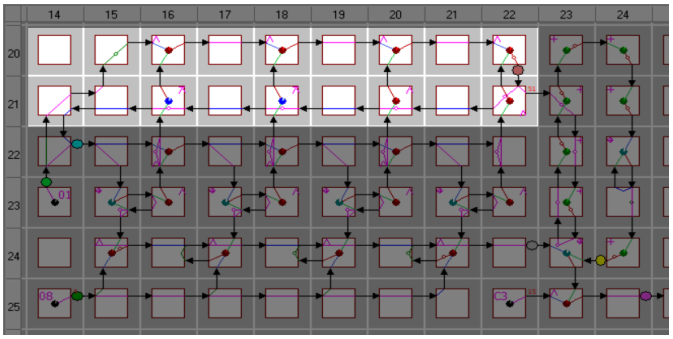
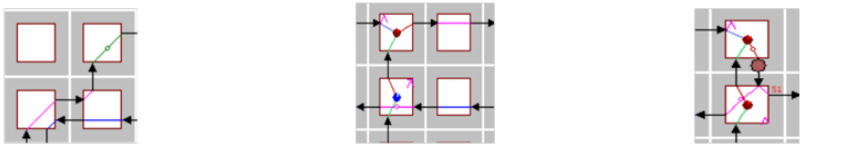


Рис. 4. Топологическая схема счетчика 4-битного входного модулятора с идентификацией сингулярного состояния

В случае если заданная последовательность бит в дешифраторе сингулярного состояния совпадет с состоянием счетчика, на выходе входного модулятора будет выдан флаг «истина» (коричневый выход рисунка 4). Затем этот единичный бит по каналам транзита ассоциативной памяти

распространяется на вход счетчика, где слиянием по «ИЛИ» объединяется со стандартным возмущающим сигналом (см. элемент (22,14) рисунка 4).

Топологическую схему такого входного модулятора под необходимую разрядность можно сгенерировать с использованием термов рисунка 5.



а) начальный терм *б) основной терм* *в) выходной терм*

Рис. 5. Базовые термы входного модулятора с идентификацией сингулярного состояния на «FIFO-регистровой ассоциативной памяти»

Генерация топологической схемы входного модулятора начинается с начального терма, который осуществляет входное и выходное взаимодействие с сопряженными операционными модулями. Далее следует набор одинаковых основных термов *б*) рисунка 5. Элемент *AND*, обозначенный на рисунке красным цветом, обеспечивает идентификацию текущего состояния одного бита счетчика, поступающего снизу.

При этом если идентификация сингулярного состояния счетчика должна срабатывать при значении «0» его текущего состояния, то в нижней строке терма применяется элемент *NOT AND*, а если при значении «1» – то элемент *AND*, как показано например в ячейке ассоциативной памяти (21,20) рисунка 4. Количество основных термов при генерации входного модулятора выбирается равным n–1, при этом они оказываются смещены на 1 ячейку вправо относительно модулируемого счетчика. Завершает генерацию рассматриваемого операционного модуля выходной терм *в*) рисунка 5. Таким образом, когда возмущающий сигнал задающего генератора формата 100..0 будет проходить через дешифратор, то «истинное» значение в его первом бите не будет обнулено тогда и только тогда, когда значение, закодированное в дешифраторе сингулярное состояние счетчика совпадет с его текущим значением, что позволяет выполнить 2 функции:

- перевести счетчик из инкрементного в декрементный режим за счет инверсии выходного сигнала;
- избавиться от паразитного дублирования выходного значения, отвечающего условиям переключения счета с инкрементного на декрементный режим и наоборот.

2.3 Синтез идентификатора текущего состояния цифрового аналога на FIFO-регистровой ассоциативной памяти

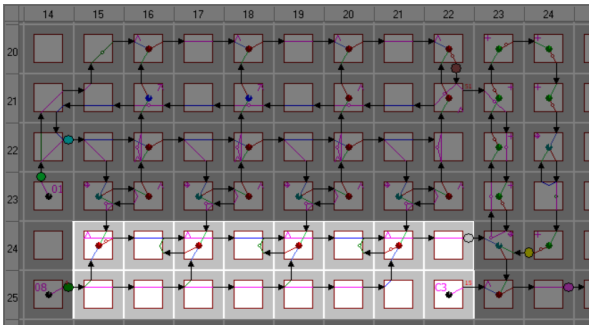


Рис. 6. Топологическая схема идентификатора текущего состояния цифрового аналога

Показанный контрастом на рисунке 6 идентификатор состояния трансформирует вектор состояния $S_4(s_0,s_1,s_2,s_3)$ в положительное целое число без знака $A_4(a_0,a_1,a_2,a_3)$, представленное в последовательном коде, которое направляется на дальнейшую обработку. Дополнительной функцией показанного на рисунке 6 варианта идентификатора является реверс последовательного представления данных таким образом, чтобы данные представлялись «младшим разрядом вперед». Отличительной особенностью данного функционального модуля является то, что каждый его терм уникален и определяется номером соответствующего извлекаемого разряда счетчика, то есть зависит от его фактического местоположения в топологической схеме (см. рисунок 7).

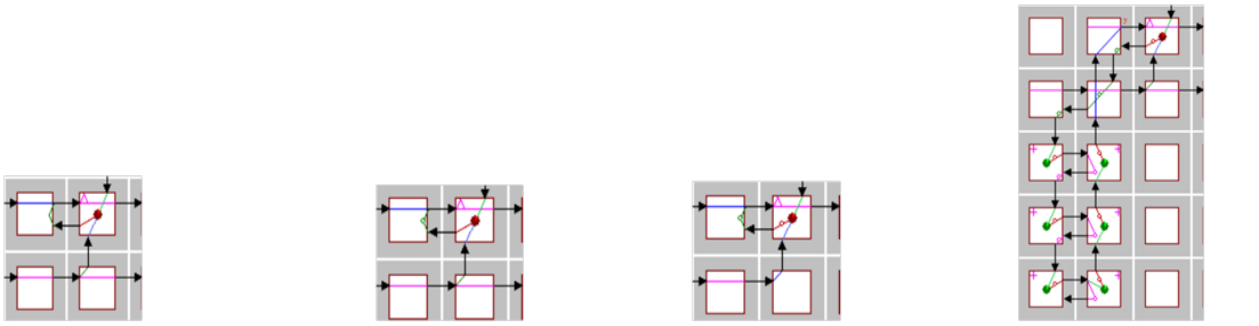


Рис. 7. Пример базовых термов реверсного идентификатора состояния «FIFO-регистровой ассоциативной памяти»

Верхняя часть каждого терма содержит стандартный для него элемент *AND*, через который осуществляется «считывание» текущего состояния соответствующего разряда счетчика. Циклический считывающий сигнал вида *1000…,1000…* распространяется по каналу транзита второй строки терма таким образом, чтобы извлекаемое состояние каждого разряда счетчика заняло свое место в последовательном коде. Как можно заметить, термы на рисунках 7*а*, 7*б* и 7*в*, отличаются линиями задержки выходной реакции *AND* при их включении в последовательный код A_m. Постоянное увеличение времени задержки распространения каждого разряда в FIFO-регистре, обеспечивает реверс значений A_m в формат «младшим разрядом вперед». Эта термальная особенность свойство обуславливает относительную сложность автоматической генерации идентификатора под заданную разрядность, что продемонстрировано на рисунке 7*г*, где изображен терм 32-го разряда счетчика. Он также состоит из стандартного для него элемента *AND*, канала транзита, а также двойного столбца ячеек ассоциативной памяти, в котором при помощи одинаковых под-термов собрана соответствующая линия задержки, которая позволяет «притормозить» 32-ой разряд выходного значения таким образом, чтобы он поступил на выход идентификатора последним.

Приведенные данные говорят о том, что, как и в кремниевых компиляторах, всегда можно выбрать термальный состав, который позволяет формализовать алгоритмы генерации топологических схем операционных устройств цифровых аналогов на базе *FIFO*-регистровой ассоциативной памяти. Благодаря этому удается в темпе реального времени адаптировать бортовые цифровые аналоги под заданную разрядность и действующее в ассоциативной памяти *множество отказов* [5], а также масштабировать временны́е и фазовые параметры компонент бортовых контроллеров БПЛА.

3 Структурная схема инструментальной платформы термального синтеза функциональных компонент цифровых аналогов

Структурная схема инструментальной платформы термального синтеза функциональных компонент цифровых аналогов во многом повторяет схему объектно-ориентированного программирования функционалов предметной области (рисунок 8).

Упрощенно внутреннюю работу такой инструментальной платформы можно описать следующим образом. Базовым элементом хранения информации о микропрограммном коде является одна ячейка *FIFO*-регистровой ассоциативной памяти («бит-процессор»), которая в терминах объектно-ориентированного программирования представлена классом *Processor*. Он содержит информацию о текущем коде операции одной ячейки, флаге задержки, направлении ее входов и выходов, параметрах каналов транзита, а также при помощи открытых методов позволяет выполнять стандартные аффинные преобразования, такие как поворот на 90 градусов в любом направлении, отражение относительно диагонали или любой из осей. Все эти операции необходимы для поддержки более высокоуровневых операций, которые в дальнейшем позволят лучше использовать «пространственно-временной» ресурс исполнительного оборудования.

На следующем уровне абстракции находится класс *ProcessorDevice*, который является моделью законченного функционального модуля, состоящего из матрицы взаимодействующих между собой ячеек *FIFO*-регистровой ассоциативной памяти. Исходя из требований предметной области, каждый объект *ProcessorDevice* содержит информацию о своих размерах и положении в пространстве двумерного исполнительного субстрата. Кроме того, он обладает списком внешних входов и выходов, представленных в программе классом *DevicePort*, благодаря чему пользователь в

графическом интерфейсе может наглядно «стыковать» входные и выходные интерфейсы различных функциональных устройств для их информационного сопряжения. Так же, как и объекты *Processor*, рассматриваемый класс позволяет выполнять все необходимые аффинные преобразования над своей внутренней структурой, позволяя менять направления и местоположение входных и выходных интерфейсов без разрушения выполняемой функции и необходимости перегенерации такого модуля. Для поддержки удобной работы с данными сущностями в графическом пользовательском интерфейсе, каждый *ProcessorDevice* содержит информацию о цвете, описании и другие параметры. Стоит отметить, что этот класс не содержит конкретной логики создания какого-либо законченного функционального устройства, и используется в качество базового. Поэтому его важным свойством является наличие нескольких виртуальных методов генерации функциональных устройств *GenerateDevice* и *GenerateDeviceWithInterface*. Эти методы можно считать реализацией объектно-ориентированного шаблона проектирования «фабрика».

Рис. 8. Структурная схема инструментальной платформы термального синтеза

Инструментальная платформа содержит встроенную библиотеку генерации готовых функциональных устройств самых востребованных типов. Каждое такое устройство в программе представлено отдельным классом, в котором переопределен один или несколько из рассмотренных выше методов *GenerateDevice* и *GenerateDeviceWithInterface*, которые по сути инкапсулируют один конкретный алгоритм (модуль) термального синтеза топологии. Их входными параметрами могут являться разрядность, физический размер, длина линии задержки и другие, при этом есть возможность обеспечить собственный специальный графический интерфейс для запроса нужных параметров у пользователя. Каждый модуль синтеза топологии использует подходящий для решения задачи набор термов, а также соответствующий алгоритм их соединения и модификации как описано выше.

Для решения задачи парирования отказов аппаратуры, которые заключаются в полной или частичной неработоспособности некоторых ячеек *FIFO*-регистровой ассоциативной памяти, можно использовать способ «пространственно-временно́го» изменения топологии размещения функциональных модулей на поверхности исполнительного субстрата таким образом, чтобы неисправные ячейки не были задействованы в работе. Инструментальная платформа может реализовать этот алгоритм в одном или нескольких адаптерах топологии цифрового аналога, которые будут выбираться на основании действующей карты отказов. Такой принцип позволяет вести борьбу за живучесть аппаратуры за счет небольшого аппаратного резерва и парировать даже множественные отказы, что недоступно современным классическим ЭВМ.

Заключение

Технологию цифровых аналогов на *FIFO*-регистровой ассоциативной памяти можно рассматривать как естественное дополнение традиционных вычислительных технологий по следующим основаниям:

а) в ней базовой парадигмой служит не вычислимость «того и только того, что требуется», а перечислимость «всего возможного» и отбор «функционально значимого», которая адекватна биологическим парадигмам естественного отбора;

б) в ней естественным образом совмещаются форматы и операторы, свойственные как системам цифровой обработки сигналов, так и системам искусственного интеллекта, работающим по правилам «нечеткой» логики;

в) в ее основе лежат не арифметические операторы, а процедуры перечисления, результаты которых допускают контекстно-зависимую интерпретацию.

Литература

1. *Алексеенко А.Г., Каляев А.В., Лукиенко В.И. и др.* Перестраиваемые цифровые структуры на основе интегрирующих процессов. – М.: Радио и связь, 1982.
2. *Данчеев В.П.* Возможности счетчиковой параллельной схемотехники //Параллельная обработки информации. Том 3: Вычислительные системы, структуры и среды для решения задач большой размерности. /Под ред. д.т.н. В.В. Грицыка. – Киев: Наукова Думка, 1986, с.258–275.
3. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. Ростов-на-Дону: из-во ЮНЦ РАН, 2008
4. Исследование путей создания полупроводниковых БИС вычислительной ячейки ОВС для построения систем обработки информации – Отчет по ОКР, номер государственной регистрации Ф30334. – Рига, 1988.
5. *Алакоз Г.М., Котов А.В., Курак М.В., Попов А.А., Сериков А.П.* Вычислительные наноструктуры /Под редакцией д.т.н., профессора кафедры Г.М. Алакоза, т.1. Задачи, модели, структуры, т.2. Программно-аппаратные платформы. – М.: «ИНТУИТ–БИНОМ», 2009.