ISSN 1684–8853

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

4(17)/2005

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

4(17)/2005

РЕЦЕНЗИРУЕМОЕ ИЗДАНИЕ

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Адрес редакции: 190000, Санкт-Петербург,
Б. Морская ул., д.67
Тел.: (812) 710-66-42, (812) 313-70-88
Факс: (812) 313-70-18
E-mail: ius@aanet.ru

Журнал зарегистрирован
в Министерстве РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-12412 от 19 апреля 2002 г.

Журнал распространяется по подписке.
Подписку можно оформить через редакцию, а также в любом отделении связи
по каталогам агентства «Роспечать»: «Газеты и журналы» – № 15385,
«Издания органов НТИ» – № 69291

ЛР № 010292 от 18.08.98.
Сдано в набор 20.05.2005. Подписано в печать 01.07.2005. Формат 60×90 1/8.
Бумага офсетная. Гарнитура SchoolBookC. Печать офсетная.
Усл. печ. л. 8,0. Уч.-изд. л. 9,0. Тираж 1000 экз. Заказ 267.

Оригинал-макет изготовлен
в отделе электронных публикаций и библиографии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

Отпечатано с готовых диапозитивов
в отделе оперативной полиграфии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

ВИКТОР ИЛЬИЧ ВАРШАВСКИЙ (1933–2005)

3 января 2005 года ушел из жизни доктор технических наук, профессор Виктор Ильич Варшавский. Поэт Лонгфелло как-то заметил: «Мы оцениваем самих себя по ощущениям того, на что мы способны, тогда как другие судят о нас по тому, что мы сделали». Виктора Ильича нет больше с нами, поэтому нам – его коллегам, ученикам, друзьям и всему научному сообществу – необходимо осознать, что им было сделано, каково его наследие.

На первом этапе своей творческой жизни В. И. Варшавский занимался проблемами междисциплинарных исследований (математика, биология, теория автоматов, коллективное поведение, распознавание образов, вычислительная техника, передача информации и др.), уже тогда гениально предугадав их ценность для развития кибернетики и исследований в области искусственного интеллекта. Позже В. И. Варшавский увлекся новой, малоисследованной тогда проблематикой – созданием асинхронных электронных устройств и систем.

По-видимому, асинхроника проистекает из основополагающей работы Д. Хаффмена (1954)[1], который предложил модель асинхронного конечного автомата. Д. Гильберт утверждал, что всякая физическая или математическая теория проходит три фазы развития: наивную, формальную и критическую. Работа Д. Хаффмена знаменовала начало формальной стадии асинхроники, которая стартовала после скрытой от нас наивной фазы, и родила тысячи работ, посвященных противогоночному кодированию автоматов и расширениям модели. Между тем все известные модели асинхронных автоматов базируются на допущении, что инициатором перехода автомата в следующее состояние является изменение входного символа, а новый входной символ может быть подан лишь после завершения переходных процессов предыдущего такта. В противном случае будут иметь место сбои. Другим существенным ограничением моделей хаффменовского типа является жесткая дисциплина смены входных сигналов, ужесточающая механизм взаимодействия автомата с внешней средой (обычно – соседние переходы). Это ограничение ставило под вопрос возможность композиции асинхронных автоматов и в конце концов стало тормозом на пути широкого практического использования подхода. Неясным оставался также вопрос, как парировать последствия нестабильности элементов реализаций автоматов (равным образом не решенный и для синхронных реализаций).

Решающим для формальной фазы асинхронной науки надо, очевидно, признать 1959 год, когда Д. Е. Маллер и У. С. Бартки опубликовали статью[2], в которой впервые предложили подход, связанный со схемами, поведение которых не зависит от задержек элементов, сейчас чаще называемых асинхронными или самосинхронными («speed-independent», «quasi delay insensitive», или «self-timed») схемами. Р. Е. Миллер во втором томе своей книги[3] пытался привлечь внимание к маллеровскому подходу, но особого успеха не добился. Чрезмерное увлечение формалистикой в этом подходе оттолкнуло практиков, не увидевших изящных схемных решений. Неудачей окончилась и попытка Маллера воплотить свои идеи в рамках проекта Illiac II, по-видимому, из-за недостаточно высокого уровня технологической базы того времени и слабой проработки схемотехники базовых узлов.

В начале 1970-х годов В. И. Варшавский, имевший за плечами богатый опыт теоретических и прикладных исследований в таких областях, как пороговая и мажоритарная логики, коллективное поведение автоматов, и смежных областях, по стечению обстоятельств пытался вместе с одним из своих аспирантов разобраться в казалось бы тривиальном вопросе – как формально синтезировать схему асинхронного триггера, известного как Гарвардский триггер. Странно, но тогда они не сумели этого сделать (это было сделано несколько позже). В то время Виктор Ильич, его коллеги и ученики не имели абсолютно никакого представления о работах, инициированных Маллером. Единственным выходом из затруднения для В. И. Варшавского была идея перехода из формальной стадии назад, в наивную фазу (что, как нам кажется, послужило одновременно началом критической стадии асинхронной науки). Он занялся изобретательством, что часто помогало ему в жизни. Его неимоверно развитая инженерная интуиция была тем волшебным паровозиком, который толкал его к теоретическому осмыслению решаемых задач.

Покрыв сотни листов бумаги схемами, кубиками (которые он любил использовать для представления и минимизации булевых функций) и форму-

[1] Huffman D. A. The synthesis of sequential switching circuits. – J. Franklin inst., – 1954. – Vol. 257. N 3–4.

[2] Muller D. E., Bartky W. S. *A theory of asynchronous circuits// In Proceedings of an International Symposium on the Theory of Switching*. – Harvard University Press. – 1959. – P. 204–243.

[3] Miller R. Switching theory. – Vol. 2. – Wiley, New York, 1967.

лами, В. И. Варшавский самостоятельно пришел к мысли о необходимости расщепления входных последовательностей на две фазы – активную (рабочую) и неактивную (спейсер), – тогда все переходы в последовательностях становятся монотонными. Он предложил конструкцию простейшего триггера с индикацией моментов окончания переходных процессов, названную триггероидом, который работает правильно независимо от реальных задержек его элементов. Триггероид не способен хранить записанную в него информацию при некоторых значениях входов, но в совокупности с двумя дополнительными триггероидами (без индикаторов) его использование решало задачу создания самосинхронного счетного триггера.

Виктор Ильич продолжал развивать свою идею и инициировал тотальный поиск литературы по асинхронике, в который погрузились все члены его команды. К 1975 году был не только сделан скрупулезный анализ результатов западных ученых, но и развит общий подход к построению самосинхронных схем и устройств, интегрирующий все известные результаты. Практически были созданы основы общей теории самосинхронизации, интегрирующей известные фрагментарные подходы. В 1976 году под редакцией В. И. Варшавского на русском языке вышла в свет книга[1], в которой систематически изложены проблемы и решения в области самосинхронизации. В то время на Западе эта книга не была замечена и не могла быть замечена, поскольку асинхронного сообщества как такового еще не существовало, а книга не была переведена на английский язык.

На фоне некоторого запустения теории автоматов в 60-е годы появилась и стала интенсивно развиваться новая научная дисциплина – теория сетей Петри. Развитие этой теории было относительно самостоятельным, но фактически не было автономным. Скорее всего, она должна рассматриваться как ветвь общей теории автоматов, которая занимает ранее неисследованную нишу между конечными автоматами и машинами Тьюринга. Подавляющая часть работ по сетям Петри «обслуживает» проблематику общей теории систем и параллельного программирования. Тем не менее, идейная сторона вопросов, связанных с выделением подкласса живых и безопасных сетей Петри и их исследованием, весьма близка к проблемам самосинхронизации. В первой книге[1] была продемонстрирована возможность прямой трансляции сети Петри из указанного подкласса к асинхронной схеме в обход процедуры противогоночного кодирования состояний автомата.

Дальнейшие исследования в области управления асинхронными процессами в команде В. И. Варшавского нашли отражение во второй книге[2], законченной в 1984 году и изданной в 1986 году под несколько скучным названием, навязанным редакцией. В 1990 году перевод этой книги на английский[3] стал доступен асинхронному сообществу, и она явилась предметом детального изучения в двух ведущих научных группах университетов Беркли и Стэнфорда.

В 1987 году Хельсинским технологическим университетом был опубликован цикл лекций Варшавского[4], прочитанный им в 1982–1983 годах. К сожалению, это издание малодоступно. Наконец, изданная в 1994 году последняя книга с участием Варшавского[5] более ориентирована на формальные модели описания, проверки и синтеза управляющих асинхронных устройств. Разработанные авторами идеи нашли применение в системе FORCAGE – работающей системе автоматизированного проектирования для практического анализа и синтеза самосинхронных схем.

В. И. Варшавский был уникальным схемотехником и изобретателем. Если внимательно проанализировать многие асинхронные блоки, нашедшие сейчас практическое применение, то в них можно увидеть прототипы схем, изобретенных Виктором Ильичем.

Как ему удавалось синтезировать изящные базисные, типовые схемы в стандартном базисе – такие, как триггеры, полусумматоры, автомат повторного вхождения, счетчики, буферные устройства и т. п. – это загадка. Применение формальных методов в подавляющем большинстве случаев не позволяло воспроизвести оригинал даже при одинаковой начальной спецификации. В. И. Варшавского не интересовали «некрасивые» схемы – он оттачивал их до блеска. Асинхронный фольклор донес до нас историю создания С-элемента Маллера (в нашем окружении он назывался гистерезисным триггером, или Г-триггером). Оказывается, первые две попытки Маллера (А и В) были неудачными, и только на букве С он остановился. Если пользоваться таким же подходом, то Варшавскому определенно не хватило бы множества больших и малых букв латинского алфавита и кириллицы. Во время посещения Советского Союза Нобелевский лауреат Поль Дирак прочел лекцию по философии физики, в ходе которой подошел к доске и написал: «Физические законы должны быть математически красивыми и простыми». Ту же мысль он высказал ранее в одной из своих лекций в Пристоне: «Вся моя жизнь – это написание красивых формул». Перефразируя, можно сказать, что разработка элегантных схем, наверное, являлась главной в творчестве Виктора Ильича.

В. И. Варшавский внес фундаментальный вклад во многие направления асинхроники. Многие его результаты приоритетны. Нам представляется разумным обратить внимание научного сооб-

[1] Апериодические автоматы/ Под ред. В. И. Варшавского. – М.: Наука, 1976. – 424 с.

[2] Автоматное управление асинхронными процессами в ЭВМ и дискретных системах/ Под ред. В. И. Варшавского. – М.: Наука, 1986. – 308 с

[3] Self-timed control of concurrent processes: The design of aperiodic logical circuits in computers and discrete systems, V. I. Varshavsky, Ed. Kluwer. – Academic Publishers, 1990. – 408 p.

[4] Varshavsky V. I. Hardware support of parallel asynchronous processes. – Helsinki, Finland: Digital Systems Laboratory. University of Technology. Series A: Research Report. N 2. 1987. – 236 p.

[5] Kishinevsky M., Kondratyev A., Taubin A., Varshavsky V. Concurrent hardware. The theory and practice of self-timed design. – J.Wiley, 1994. – 368 p.

щества на вклад В. И. Варшавского и предупредить, что терминология, использованная в первоисточниках, часто отличается от современной. Одна из причин этого состоит в использовании разных языков, а другая – в желании многих авторов использовать собственную терминологию и почувствовать себя первооткрывателями.

Перечислим важнейшие результаты В. И. Варшавского в асинхронике.

- Самосинхронная реализация комбинационных логических схем и конечных автоматов (парафазная и четырехфазная с встроенными индикаторами), 1976.
- Прямая трансляция управляющих спецификаций (типа параллельных асинхронных блок-схем алгоритмов и сетей Петри) в асинхронные схемы управления, 1976.
- Работы по самосинхронным кодам и реализации кодов в изменениях, 1981.
- Самосинхронные интерфейсы, использующие двух- и трехстабильные линии с избыточным кодированием или временную избыточность, 1981–1988.
- Надежные самотестируемые и саморемонтируемые архитектуры, 1982–1986.
- Конструктивное доказательство функциональной полноты двухвходовых элементов в классе полумодулярных схем, 1981–1986.
- Десинхронизация синхронных реализаций посредством замены синхронных часов асинхронным управляющим автоматом для улучшения временных свойств схем, 1994–1998.
- Асинхронные реализации FIFO-структур и схем памяти, 1988–1993.
- Проектирование асинхронных схем на основе квантовых устройств (квантовых точек, одноэлектронных транзисторов), 1995–1996.
- Инициация работ по программной поддержке методов проектирования (синтеза и верификации) асинхронных схем, 1979–1993.
- Схемы, нечувствительные к задержкам в транзисторах и проводах, 1987.
- Проектирование конвейерного управления и конвейерных схем с разной плотностью заполнения информацией (неплотных, полуплотных и плотных), 1979–1986.
- Мостиковые транзисторные реализации (в том числе двух- и трехвходовых С-элементов), 1988.

Вне асинхроники достаточно упомянуть ставшими классическими результаты в следующих областях:

- пороговая логика и искусственные нейроны, которыми Варшавский занимался в начале своей научной карьеры и вернулся в последние годы[1];
- коллективное поведение автоматов, ставшее темой докторской диссертации Виктора Ильича[2].

Интересно заметить, что в последнее время заметно вырос интерес к поведению стохастических автоматов с переменной структурой[3].

Удивительной была способность В. И. Варшавского сколачивать команду. Его окружение работало как хорошо отлаженный механизм, который никогда не давал катастрофических отказов, хотя отдельные зависания и тупики, конечно, случались. Виктор Ильич никого не отпускал в автономное плавание – он выдавал задания, обсуждал возможные пути их решения и постоянно был в курсе их выполнения. Такой контроль не вызывал обычно отрицательных реакций, потому что он осуществлялся по-дружески, без дистанции, обычно существующей между боссом и подчиненными. Матерые сотрудники иногда втихомолку роптали – у них были собственные идеи, но прессинг они считали нормальным, потому что репутация босса была хорошим прикрытием от неприятных внешних воздействий.

На всех этапах своей жизни Варшавский не переставал учиться и учить. Во многом его мировоззрение сформировалось на зимних школах под Ленинградом, которые он сам создал в начале 1960-х годов. Ему удалось привлечь к сотрудничеству в этих школах таких выдающихся ученых и мыслителей, работающих в разных областях науки, как М. Л. Цетлин, М. М. Бонгард, Л. И. Розоноер, В. С. Гурфинкель, С. М. Осовец, Д. А. Поспелов, Я. А. Альтман и др. Школа питала мыслями и идеями всех ее участников более 10 лет.

В. И. Варшавский принимал активное участие в работе школ М. А. Гаврилова, общепризнанного пионера в области теории автоматов в России, который всегда активно поддерживал работы команды В. И. Варшавского.

Ему редко представлялась возможность побывать и поработать за границей в условиях железного занавеса. Но иногда это удавалась. Иностранные гости АН СССР, работающие в области теории автоматов, обязательно включали в программы своих поездок посещение В. И. Варшавского. Ему принадлежит и инициатива объединения «могучей кучки» эстонских ученых. Ежегодные зимние совместные семинары с участием ленинградской команды были названы «встречами на Эльби» по имени местечка под Пярну, где они проводились. Варшавский был ключевой фигурой и в подготовке ядра эстонских национальных кадров высшей квалификации в области теории автоматов и ее приложений. Можно упомянуть также множество постоянно действующих семинаров, которыми он руководил.

В. И. Варшавский был экстраординарной фигурой и в общечеловеческом плане. Он имел громадное количество друзей и знакомых, которые любили и почитали его. Он охотно поддерживал контакты и был всегда готов помочь советом или действом.

---

[1] Avedillo M. J. VLSI implementation of threshold logic: a comprehensive survey. – IEEE Trans. Neural Networks, 14(5), Sept. 2003.

[2] Варшавский В. И. Коллективное поведение автоматов. – М.: Наука, 1973. – 408 с. (German edition: Warshawski W. I. Kollektives Verhalten von Automaten. – Akademie-Verlag-Berlin, 1978).

Варшавский В. И., Поспелов Д. А. Оркестр играет без дирижера. – М.: Наука, 1984 (English edition: Varshavsky V., Pospelov D. Puppets without strings. – Moscow: Mir Publishers, 1988. – 294 p.)

[3] Economides A. A., Kehagias A. The STAR automaton: expediency and optimality properties. – IEEE Transactions on Systems, Man and Cybernetics, Part B. – Vol. 32. № 6. Dec. 2002. – P. 723–737).

Никто и не пытался сосчитать число его аспирантов, число защит, на которых он с блеском выступал в качестве оппонента, или число статей и книг, которые он отрецензировал (часто – с сарказмом и сногсшибательной критикой).

В. И. Варшавский обладал неистребимым чувством юмора. Анекдоты хлестали из него, как из пожарного брандспойта. Его розыгрыши, мгновенная реакция на происходящее, острые ремарки, убийственно меткие характеристики людей и событий вошли в научный фольклор. Он был незаменимым тамадой на банкетах, и многие его друзья, которые сами по себе были замечательными спичмейкерами, в его присутствии слегка сникали.

При полном отсутствии музыкального слуха, Виктор Ильич очень любил петь, причем, будучи человеком самобытным, он предлагал свои собственные интерпретации песен, основанные не столько на мелодике, сколько на рваном ритме (может быть, поэтому он пришел в асинхронику?). Интересно, что его хоровое исполнение всегда вызывало катастрофические отказы – соисполнители сначала сбивались, а потом выбывали из хора. Правда, Виктора Ильича это нисколько не смущало... Виктор Ильич с детства в полной мере впитал в себя колорит ленинградского послевоенного фольклора и знал огромное количество дворовых песенок и уникальных вариантов всенародно известных песен, например «Раскинулось море широко».

В студенческие годы Виктор Ильич занимался классической борьбой и достиг значительных успехов, став кандидатом в мастера спорта. Начав работать, он забросил борьбу, но сохранил неистребимое желание выигрывать. Спортивные интересы переместились не только в область работы, но и в область шахмат. Он не занимался ими серьезно, но обожал блиц, часто доводя коллег до отчаяния, потому что любой разговор начинался и кончался десятком партий.

К сожалению, Виктор Ильич всю жизнь, с раннего детства, курил, дымя как паровоз, а в короткие промежутки времени, когда он под прессингом своей жены Натальи пытался завязать с курением, всегда сосал сигарету, засунутую в рот нештатным концом. Наверное, именно курение оборвало его яркую жизнь, но работать столь эффективно без сигарет он вряд ли бы смог.

В. И. Варшавский никогда не был карьеристом. Он, естественно, не мог недооценивать себя, но никогда не рвался к власти. Позиции профессора и заведующего лабораторией его вполне устраивали, поскольку ему не хотелось тратить свою жизнь на дело, с его точки зрения, пустое... Его призванием была работа, и он явно относился к разряду трудоголиков. Того же он требовал от своего окружения.

Может показаться странным, но во многом интуитивный отбор кадров в подавляющем большинстве случаев не подводил его. Он не всегда с первого захода принимал результаты и мнения своих коллег и учеников. Его надо было сломить или, по крайней мере, выждать некоторое время. Возможно, это происходило оттого, что он беспрестанно генерировал новые и новые идеи, развитием которых хотел озадачить свое окружение.

В. И. Варшавский оставил множество последователей, которые развили его идеи и обогатили науку и практику новыми идеями и разработками. Если ранее они работали только в пределах советской России и коммунистических странах Восточной Европы, то к настоящему времени многие из них иммигрировали в более дальние страны и с честью влились в асинхронное сообщество, которое, по нашему общему мнению, в основном продолжает пребывать в формальной фазе своего развития. Критическая фаза по-прежнему только начинается, в чем она проявится и чем завершится – остается только ждать. Возможно, В. И. Варшавский наметил некоторые пути развития критической стадии асинхроники как любимого его детища, которые заложены в идеях, связанных с построением схем, нечувствительных к задержкам как в транзисторах, так и проводах, а особенно в десинхронизации. 20–25 лет тому назад казалось, что асинхроника должна вот-вот победить синхронный подход. Этого не произошло, наверное, потому, что синхронная компьютерная технология развивалась куда более быстрыми темпами, чем асинхроника. Комбинация обоих подходов, возможно, станет реально продуктивной.

Мы выражаем благодарность журналу «Информационно-управляющие системы» за предоставленную возможность опубликовать неизвестные в России статьи В. И. Варшавского.

Память о нашем друге, учителе, коллеге навсегда останется в наших сердцах.

Л. Розенблюм, М. Кишиневский,
А. Кондратьев, О. Маевский, Ю. Мамруков,
В. Мараховский, Н. Стародубцев, А. Таубин,
А. Яковлев, А. Шалыто

УДК 681.3

СИСТЕМНОЕ ВРЕМЯ И СИНХРОНИЗАЦИЯ СИСТЕМ[1]

В. И. Варшавский,
доктор техн. наук, профессор
университет Айдзу (Япония)

В статье проводится анализ концепции причинной обусловленности как основы правильного асинхронного поведения. Предлагается декомпозировать систему на процессорный стратум и синхро-стратум, функционирующий подобно распределенным асинхронным часам. Универсальный синхро-стратум обеспечивает возможность прямого перехода от синхронного прототипа к асинхронной системе. Приводятся несколько типов спецификаций и аппаратной реализации синхро-стратума вместе с их сравнительными характеристиками.

In the paper, the concept of causal conditionality is analyzed as the basis for correct asynchronous behavior. It is suggested to decompose the system to Processors Stratum and Synchro-Stratum which acts like a distributed asynchronous clock. A universal Synchro-Stratum provides the possibility of direct transition from the synchronous prototype to an asynchronous system. Several types of specification and hardware implementation of Synchro-Stratum are given with their characteristics compared.

Введение

Время является одним из наиболее сложных и плохо понимаемых объектов Природы, хотя большинство из нас полагают, будто отлично знают, что это такое. История, психология, искусство, микро- и макрофизика, космология и ряд других наук занимаются разными аспектами Времени и имеют на него, вообще говоря, разные точки зрения. Предметом этой статьи будет Время в искусственных системах, или Искусственное Время. Специфика этого предмета требует, по-видимому, специального рассмотрения. Прежде чем перейти к изложению основного содержания статьи, я позволю себе некоторые спекуляции на эту тему.

Начиная, вероятно, с Исаака Ньютона, достаточно точно сформулировавшего это понятие, Время рассматривается как независимая глобальная физическая переменная (Физическое Время). Все определения Физического Времени, с которыми нам удалось познакомиться, так или иначе связаны с процедурой измерения Времени[2] и понятием одновременности. Измерение, в свою очередь, всегда связано с его точностью, и одновременность является абстракцией в пределах точности измерительного прибора (часов).

Теория относительности ввела в рассмотрение понятие события и зависимость интервала измеренного времени между событиями от позиции наблюдателя. При этом, хотел ли этого Эйнштейн или нет, было введено понятие информационного обмена (сигнала о событии). Более того, событие по своей природе дискретно и возникает естественный вопрос, дискретно или непрерывно физическое время? С точки зрения базового определения, оно непрерывно. С другой стороны, оно гранулируется, по крайней мере, точностью измерительного прибора. Поскольку любой измерительный процесс требует расхода энергии и является вмешательством в измеряемый процесс, то соотношение неопределенности в его форме, связанной со временем, может дать информацию о минимальном кванте времени и, надо полагать, о возможной достижимой точности измерения[3]. Соотношение неопределенности в форме, связывающей энергию и время, имеет вид $\Delta E \Delta t \geq \hbar$. Это означает, что энергию взаимодействия между измерительным прибором и исследуемой системой можно учесть только с точностью до $\hbar / \Delta t$. Хотя сегодня микроэлектроника весьма далека от этого ограничения, но для FED[4], использование ко-

[1] **Varshavsky V.** System Time and System Timing. C. Nehaniv & M. Ito, eds., Algebraic Engineering, World Scientific Press, 1999. – P. 38–57.

[2] Это естественно, как для любой физической переменной. Измерительным прибором для Времени являются Часы.

[3] Честно говоря, на мои вопросы к физикам о смысле этой формы соотношения неопределенности я не получил вразумительного ответа. Возможно, я спрашивал не тех людей.

[4] Future Electron Device, таких как Single Electron Transistor Wave Function Rearrangement Devices Quantum Dot Cell и т. д.

торых ожидается во второй декаде следующего века, время смены энергетического уровня (состояния) на величину ΔE распределено по Пуассону с параметром $\tau = \hbar / \Delta E$ [1].

Как это ни парадоксально, но определение Времени, связанное с последовательностью дискретных событий, было дано значительно раньше, еще Аристотелем. Согласно Аристотелю, Время – это способ упорядочивания событий, отражающий причинно-следственные отношения между ними. С этой точки зрения Время есть логическая абстракция (Логическое Время) и суть его отражается в гениальной фразе Аристотеля: «Время отсутствует, если ничего не происходит».

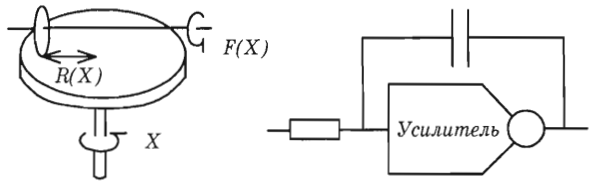
Оба определения, как физического, так и логического времени были сделаны применительно к естественному времени, связанному с независимыми от человека процессами, протекающими в Природе. Начиная с самых ранних периодов истории, у человека возникла необходимость в создании искусственных систем (устройств), поведение и использование которых было связано со временем. Во-первых, это часы. Функционирование всех известных часов связано с процессами, протекающими в непрерывном физическом времени. В большинстве случаев часы осуществляют преобразование непрерывного физического времени в последовательность дискретных событий. Во-вторых, это системы, устанавливающие логический порядок событий, как например, календари. Однако по-настоящему проблема времени в искусственных системах возникла при создании средств моделирования процессов, как аналитических, так и технических.

Аналитические непрерывные модели, такие как дифференциальные уравнения, включают непрерывное физическое время как независимую переменную без каких-либо специальных проблем, тем более, что и само определение независимого физического времени было в той или иной мере связано с созданием дифференциального исчисления.

Аналитические модели дискретных процессов, такие как уравнения в конечных разностях или конечные автоматы, оперируют с дискретным временем, вернее, с номерами шага процесса, которые в рамках этих моделей, на первый взгляд, никак не связаны с естественным временем. Но это справедливо только для изолированных систем, и как только мы попытаемся организовать или описать[2] взаимодействие системы с внешней средой, мы будем вынуждены заняться взаимодействием формального и естественного времени.

[1] При этом $\tau \approx 6 \times 10^{-4}$ пс/эВ. Для продвинутых микроэлектронных устройств характеристические энергии переключения имеют порядок FJ (1 FJ ≈ 600 эВ) и Δt имеет порядок долей пикосекунды без учета энергии перезаряда паразитных емкостей.

[2] В рамках аналитических моделей это, по-видимому, одно и то же.



■ *Рис. 1. Схемы интеграторов*

Косвенно с этим связаны трудности перехода от абстрактной к структурной модели автомата и определения асинхронного автомата в классической структурной теории автоматов[3].

Как только мы переходим к техническим системам, моделирующим те или иные процессы, мы с неизбежностью сталкиваемся с тем фактом, что физические устройства, реализующие техническую систему, функционируют в естественном физическом времени, а аналитические процессы, которые они моделируют, заданы в искусственном формальном времени. Таким образом, проблема введения времени в искусственную систему (Системное Время) и проблема организации взаимодействия между системным и физическим временем становится одной из проблем проектирования.

Впервые эта проблема в явном виде возникла в начале века при создании и использовании в режиме реального времени механических дифференциальных анализаторов (МДА). Одним из базовых элементов МДА является дисковый интегратор (рис. 1)[4].

Если маленький диск радиуса r перемещается относительно центра большого диска на величину $R(X)$ и угол поворота оси большого диска при этом равен X, то угол поворота оси маленького диска равен $F(X) = \int_0^X \frac{R(X)}{r} dX$. Как нетрудно видеть, поведение интегратора не зависит от времени и в качестве независимой[5] может быть использована любая переменная. Если же мы хотим работать с физическим временем как независимой переменной, то для этого достаточно иметь постоянную скорость вращения ω оси большого диска. При этом $F(t) = \int_0^t \frac{\omega \cdot R(t)}{r} dt$ и синхронный двига-

[3] Ничего другого, кроме как «асинхронный автомат – это автомат, переходы которого инициируются изменением состояния входа и любой переход завершается стабильным состоянием», придумано не было, но каждый, кому пришлось читать студентам лекции по структурной теории автоматов, знает, сколь трудно, а порой и невозможно ответить на вопросы студентов: «Чем же все-таки отличается синхронный автомат от асинхронного?».

[4] С его помощью при наличии следящей системы в цепи обратной связи реализуется и операция дифференцирования. Запаздывание в цепи обратной связи сказывается только на точности вычислений.

[5] Или зависимой.

тель, обеспечивающий постоянную скорость вращения, играет роль общих для системы часов. Ситуация принципиально изменяется при переходе к электронным дифференциальным анализаторам. Их базовым компонентом является интегрирующий усилитель (рис. 1, *б*), поведение которого в явном виде зависит от физического времени, и значения всех переменных являются функцией времени. Теперь нам опять требуются часы для того, чтобы сопоставить значения различных переменных в один и тот же момент времени[1]. Уже здесь наметились некоторые проблемы, связанные с тем, что усилитель обладает собственной задержкой и изменения распространяются по анализатору с некоторой задержкой. Хотя в большинстве применений этим можно было бы пренебречь, однако для моделирования очень быстрых процессов было необходимо прибегать к масштабированию времени[2].

История возникновения дискретных вычислительных средств начинается еще в древности[3]. Однако вопрос о времени, в котором функционируют соответствующие технические средства, возник значительно позже. Тем более, что значительно позже возникли формальные модели поведения таких устройств. Механические вычислительные устройства, такие как машины Паскаля, Лейбница, Однера, аналитическая машина Беббиджа, имели в качестве единицы системного времени однократный поворот ведущего вала[4], который хотя и протекал в реальном физическом времени, но в процессе работы это фактически никак не учитывалось. Введение в начале XX века электрического привода[5] никак не изменило ситуацию. Жесткость механического привода и зубчатых передач исключали ощутимую задержку передачи информации и, например, в зависимости от длины переноса при суммировании изменялась не задержка, а момент на ведущем валу[6]. В калькуляторах, где были автоматизированы операции умножения и деления, последовательность шагов вычисления задавалась последовательностью оборотов ведущего вала (системными часами).

Заметим как исторический курьез, что идея формирования шагов системного времени по готовности данных (то, что позже было названо «архитектурой потока данных») была впервые использована в XVIII веке в вычислительном бюро Прони, которое занималось составлением навигационных таблиц. Группа высококвалифицированных математиков, в которую входил Лагранж, разрабатывала методы вычислений (алгоритм) и систему таблиц, выходные данные которых были результатом одной арифметической операции над входными (программа). Оператор-вычислитель получал исходные таблицы и заполнял одну или несколько выходных таблиц, которые поступали следующему оператору. Оператор начинал очередной шаг вычислительного процесса, когда к нему поступали полные исходные данные для этого шага[7].

Настоящие проблемы взаимодействия физического и системного (логического) времени возникли, по-видимому, с появлением и широким использованием релейных устройств в промышленной автоматике и системах сигнализации и блокировки на железных дорогах. Эти проблемы были связаны с опасностью возникновения гонок и состязаний, характер которых зависел от вариаций времен срабатываний контактов реле. Именно борьба с гонками и состязаниями впервые поставила задачу организации логического поведения, инвариантного к поведению компонент в физическом времени. Вызванные запросами практики работы Шестакова и Гаврилова (СССР), Шеннона (США), Накамуры (Япония) привели к созданию теории релейных устройств и структурной теории конечных автоматов (Мур, Мили, Хаффмен и др.). Появление электронных ламп и транзисторов до поры до времени не изменяло общую ситуацию.

Модель конечного автомата $S(t+1) = F(S(t), X(t+1))$; $Y(t+1) = \Phi(S(t), X(t+1))$ – модель Мили; $Y(t) = \Psi(S(t))$ – модель Мура – включала в себя параметр t как номер шага процесса, который не был связан с физическим временем и был уже, по сути, системным искусственным временем[8]. Именно попытки исключить параметр t из автоматных уравнений привели к понятию асинхронного автомата, для которого текущее состояние определяется устойчивым решением уравнения $S = F(S,X(t))$. При этом параметр t сохраняет свое значение нумератора последовательности состояний входа. Коль скоро модель асинхронного автомата была предложена и изучалась применительно к задачам реального проектирования, то сразу же возникли вопросы связи логического поведения автомата с физическим поведением его компонент, ответы на которые искались на логическом уровне (противогоночное кодирование и др.).

[1] Опять заметим, что одновременность достижима лишь в пределах точности часов.

[2] Масштабированное время есть не что иное, как искусственное системное время.

[3] Вспомним камешки, зарубки на дереве, узелки, применявшиеся для счета и, наконец, счетные доски и абак. Заметим также, что математическое мышление древних людей было алгоритмично – в древнем Египте математическая задача считалась решенной, только если был найден алгоритм соответствующих вычислений на абаке.

[4] Или однократное перемещение рычага в машине с пропорциональным рычагом.

[5] Калькуляторы Reinmetall, Mercedes, Facit, Fenix и т. д.

[6] Что могло быть компенсировано запасом мощности и изменяло не единицу системного времени, а ее длительность в реальном физическом времени.

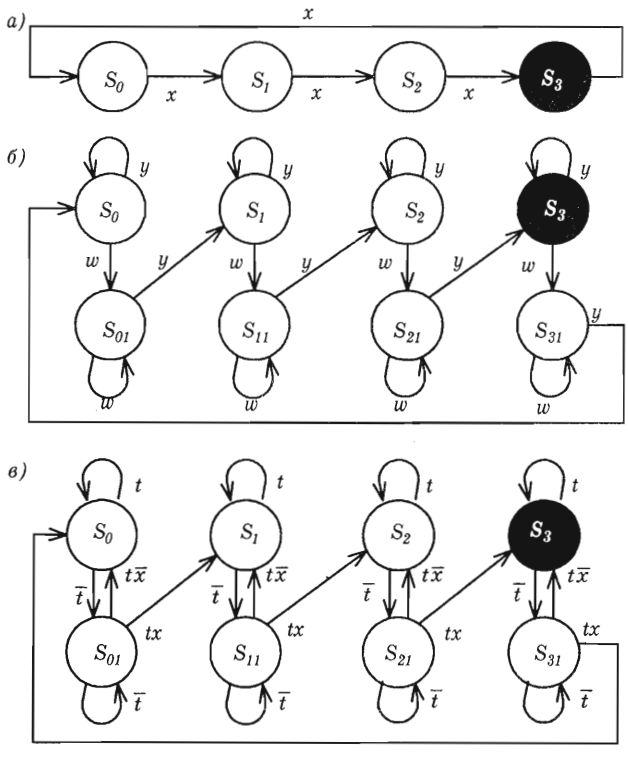
[7] Чем не мультипроцессорная data-flow computer system?

[8] Обратим внимание на содержательную близость модели конечного автомата и разностного уравнения.

Теория алгоритмов (нормальные алгоритмы Маркова, машины Тьюринга и Поста и др.) и абстрактная теория автоматов (регулярные события Клини и др.) развивались, также игнорируя реальное время. Системное время в этих моделях рассматривалось как шаг алгоритма или номер позиции символа во входном слове. Однако уже здесь стали очевидными некоторые противоречия между структурной и абстрактной теорией автоматов.

Действительно, например, по регулярному событию $xxxx(xxxx)^*$, используя прямой алгоритм МакНотона-Ямады, можно построить представляющий это событие конечный автомат (рис. 2, *а*)[1]. С точки зрения абстрактной теории, все это абсолютно корректно, но, с точки зрения структурной теории, такой автомат является не счетчиком по модулю 4, а генератором. Для обеспечения корректной работы автомата мы должны ввести либо пробел (спейсер), разделяющий два последовательных символа x, либо сигнал смены символа t. Тогда, чтобы обеспечить введение системного времени и корректную работу автомата, мы должны изменить входной алфавит и регулярное событие, представление которого есть счет по модулю 4.

Пусть $y = x \& t$, $w = \overline{x} + \overline{t}$ (синхронный вариант) или $y = x$, $w = \overline{x}$ (асинхронный вариант) и соответствующее регулярное событие имеет вид $ywywywyw(ywywywyw)^*$. Изменение регулярного события приводит к изменению распознающего это событие конечного автомата (рис. 2, *б*)[2].



■ ***Рис. 2.*** *Введение времени в граф автомата*

[1] Счетчик по модулю 4.

Несмотря на значительное число работ по асинхронным автоматам и несомненные успехи теории, наиболее простым способом введения системного времени[3] все-таки на много лет стало использование внешних часов (синхронная реализация). При этом для рассмотренного примера реализация конечного автомата по принципу работы «основной–вспомогательный» задавалась графом, приведенным на рис. 2, *в*.

Использование внешних часов или, точнее говоря, сигнала от внешних часов в качестве дополнительной глобальной переменной, формирующей шаги процесса смены дискретных состояний устройства, имеет простую структуру (рис. 3, *а*).

Часы представляют собой периодический калибровочный процесс, который делит процессы, протекающие в непрерывном физическом мировом времени в среде и устройстве, на дискретные шаги. При этом должно действовать соглашение о том, что и в среде, и в устройстве длительность фазовых переходов из одного дискретного состояния в другое не превосходит[4] длительности периода работы часов. В рамках приведенной структуры часы представляют собой не что иное, как калиброванную задержку. Важно при этом отметить, что события в часах не имеют причинной связи ни с событиями в среде, ни с событиями в устройстве. Все три объекта имеют, среди прочих, только одну общую причину происходящих в них событий – течение мирового физического времени. Заметим на будущее, что отсутствие причинно-следственных отношений между указанными событиями или, другими словами, нарушение причинной обусловленности поведения является одним из источников неприятностей, с которыми сталкивается синхронная реализация.

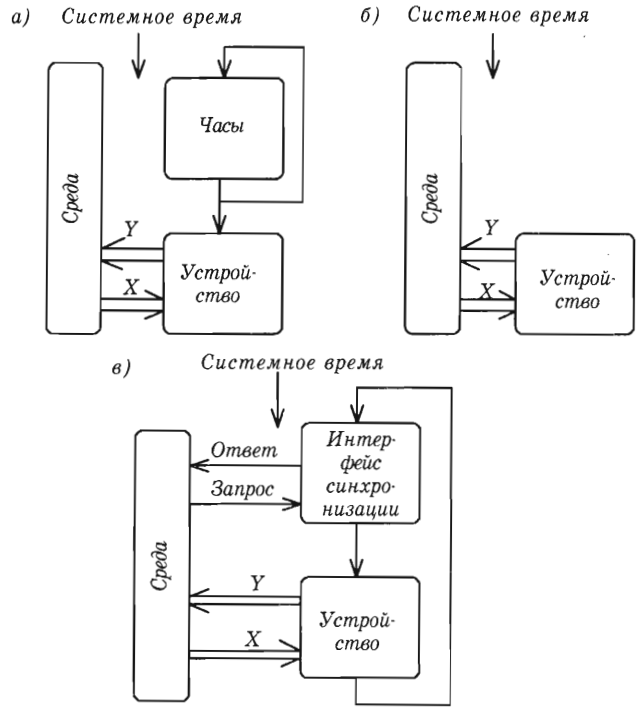
В классической асинхронной реализации (рис. 3, *б*) инициатором переходного процесса в устройстве является внешняя среда[5]. При этом, опять-таки, поведение во времени внешней среды в значительной мере причинно независимо от поведения устройства. В тех случаях, когда внешняя среда является частью системы, а часто так оно и есть, требуется постановка задачи интерфейса между системным и естественным временем и ее решение. Од-

[2] Мы получим тот же самый автомат, если осуществим подстановку yy^* и ww^* вместо y и w соответственно, но это создает лишь иллюзию асинхронности в рамках языка регулярных событий.

[3] Или, точнее говоря, преобразования физического времени в системное.

[4] Более корректно, в пределах точности поддержания периода калибровочного процесса, строго меньше.

[5] Изменение состояния входного вектора X. Заметим при этом, что и в синхронной модели часы фактически являются для устройства частью внешней среды и сигнал времени T может быть включен во входной вектор – $X' = \{X, T\}$.



■ *Рис. 3. Взаимодействие устройства и среды во времени*

нако, если внешняя среда или существенная ее часть является искусственной системой, мы вправе попытаться ввести в интерфейс между ними причинную обусловленность. Эту задачу решает так называемая «согласованная реализация» (рис. 3, *в*).

Суть согласованной реализации, предложенной в пионерских работах Д. Маллера, состоит в том, что внешняя среда навязывает ритм смены состояний устройству, а устройство – внешней среде. При этом, по крайней мере, один выход устройства изменяется только после завершения очередного переходного процесса в нем. Развитие модели Маллера привело к созданию нового класса устройств с причинно обусловленным поведением – самосинхронных (не зависящих от скорости, нечувствительных к задержкам) устройств.

Модель Маллера [5, 6][1]. Причинно-обусловленные устройства

Принципиальным в подходе Маллера к изучению поведения асинхронных устройств явился отказ от детерминированной модели. Действительно, поскольку асинхронное поведение в общем случае связано с непредсказуемыми вариациями длительностей процессов перехода из одного состояния в другое в физическом времени, состояние асинхронного устройства в любой фиксированный момент физического времени не определено. В ряде случаев мы имеем достаточно полное представление о вероятностном характере вариаций задержек и о параметрах соответствующих распределений вероятностей[2]. Однако никакой особой пользы вероятностный подход к изучению логического поведения не приносит. В действительности нас интересует ответ на вполне определенный вопрос: «*Как и при каких условиях из поведения, недетерминированного в физическом времени, получить поведение, детерминированное в логическом времени?*». Для получения ответа на этот вопрос в модели Маллера изучается полная совокупность логических возможностей.

Полная схема, или схема Маллера[3], определяется как конечное множество S из N объектов $a, b, c, ...$, называемых состояниями схемы. Поведение схемы задается множеством допустимых последовательностей состояний:

$$\begin{Bmatrix} a(1), a(2), \ldots \\ b(1), b(2), \ldots \\ \ldots\ldots\ldots\ldots \end{Bmatrix},$$

обладающих следующими свойствами:

$$a(i) \neq a(i+1) \ \forall i;$$

каждое состояние $a \in S$ является началом, по крайней мере, одной допустимой последовательности;

если $a(1)$, $a(2)$, $a(3)$, ... – допустимая последовательность, то $a(2)$, $a(3)$, ..., $a(2)$ – также допустимая последовательность;

если $a(1)$, $a(2)$, ... и $b(1)$, $b(2)$, ... – допустимые последовательности и $a(2) = b(1)$, то $a(1)$, $b(1)$, $b(2)$, ... – допустимая последовательность[4].

Схема может быть задана ориентированным графом возможных соседних переходов из одного состояния в другое. Состояния $a \in S$ и $b \in S$ находятся в отношении aRb, если существует логическая возможность непосредственного перехода из состояния a в состояние b, т. е. если на графе существует дуга, ведущая из a в b. Отношение R порождает множество окрестностей a $Q(a) \in S$, таких, что $b \in Q(a)$, если и только если aRb. Последовательность $a(1)$, $a(2)$, ..., $a(m)$ называется R-последовательностью, если $a(i)Ra(i+1)$ для $1 \le i \le m-1$. Каждому пути на графе соответствует R-последовательность состояний[5].

[1] Изложение результатов Маллера дается по работе [7, гл. 10]. Ряд утверждений приводится без доказательств, которые можно найти там же.

[2] Как, например, в случае смены квантовых состояний или выхода электронного арбитра из метастабильного состояния.

[3] Или просто схема.

[4] Последние два свойства формируют свойство марковости допустимой последовательности – вся предыстория поведения содержится в текущем состоянии.

[5] Но не каждая R-последовательность является допустимой.

Мы будем говорить, что состояние a предшествует состоянию b (aFb – отношение следования), если существует R-последовательность (путь на графе), ведущая из a в b. Отношение F рефлексивно (a, a есть R-последовательность) и транзитивно (из aFb и bFc следует aFc). Если $a \neq b$ и aFb, то a и b принадлежат допустимой последовательности.

Состояния a и b называются эквивалентными (aEb), если aFb и bFa, т. е. если в графе имеется цикл, содержащий a и b. Отношение эквивалентности разбивает все множество состояний S на классы эквивалентности $A, B, C, \ldots$. Отношение следования имеет место и для классов эквивалентности AFB, если существуют два состояния $a^* \in A$ и $b^* \in B$, такие, что aFb. И, если AFB, то $\forall(a \in A)\&(b \in B)\ aFb$. Если AFB и BFA, то $A = B$.

Отношение F для классов эквивалентности определяет частичное упорядочение классов[1]. Любой допустимой последовательности состояний соответствует единственная конечная последовательность классов эквивалентности и для любой допустимой последовательности состояний существует последний класс эквивалентности $A(m)$, называемый заключительным. В частичном порядке на классах эквивалентности существует один или несколько максимальных классов. Если заключительный класс для некоторой допустимой последовательности не является максимальным, то он называется псевдомаксимальным. Согласно определению Маллера, схема называется независимой от скорости[2] относительно состояния u, если все допустимые последовательности, начинающиеся с u, имеют один и тот же заключительный класс.

Теорема [7, 10.2.5]. Схема C не зависит от скорости относительно u, если и только если класс эквивалентности $U(u \in U)$ предшествует только одному максимальному классу K и не предшествует ни одному псевдомаксимальному классу.

Определенная выше схема C является абстрактным объектом. Структурная теория предусматривает более детальное описание, в частности, представление состояний сигналами, а сигналов – логическими элементами. Структуризация требует изменения[3] определения схемы.

Как обычно в структурной теории автоматов, схема определяется как совокупность n логических элементов, соединенных между собой. Каждый логический элемент имеет k_i входов, $k_i \leq n$, и один выход x_i. Выходы логических элементов называются узлами схемы и n-местный набор значений переменных в узлах схемы $X = \{x_0, x_1, \ldots, x_{n-1}\}$, является ее состоянием[4]. Каждый вход логического элемента соединен[5] только с одним выходом и никакие два выхода не соединены между собой. С каждым логическим элементом[6] связана логическая функция $x_i' = f_i(X)$, где x_i' называется следующим значением x_i. Если x_i' явно зависит от x_i, то логический элемент есть элемент памяти. В противном случае мы имеем комбинационный элемент. Выход логического элемента в некотором состоянии X_j называется стабильным, если $x_i \oplus f_i(X_j) = 0$, и возбужденным, если $x_i \oplus f_i(X_j) = 1$. В возбужденном состоянии входы логического элемента *уже* изменились[7], а выходная переменная *еще* не изменилась. Если в некотором состоянии X_j возбуждено m переменных, то существует 2^m логических возможностей для смены состояний, порождающих граф полной схемы Маллера. Если каждой структурированной схеме соответствует абстрактная схема, то все определения и результаты, полученные для абстрактной модели[8], могут быть распространены на структурированную модель. При этом, однако, специфика структурированной модели оправдывает рассмотрение других подходов.

В соответствии с определением возбужденного и стабильного состояний логического элемента в бинарной модели Маллера каждая переменная может принимать четыре значения: $\{0,1\}$ – стабильные значения и $\{0^*,1^*\}$ – возбужденные значения. Диаграмма состояний при этом называется диаграммой Маллера. Пример диаграммы Маллера для схемы

$$x_0 = x_2 + \bar{x}_3;\ x_1 = \bar{x}_3;\ x_2 = x_1\bar{x}_3;$$

$$x_3 = x_0x_2 + x_3(x_0 + x_1)$$

относительно начального состояния $0^*0^*0\,0$ приведен на рис. 4, a[9]. Заметим, что, несмотря на четырехзначное кодирование, схема имеет не 4^n, а 2^n возможных состояний, из которых в нашем примере четыре состояния недостижимы из рассмотренного начального состояния. Для состояний a и b диаграммы Маллера естественным образом определяется отношение непосредственного следования aRb и понятие R-последовательности. Каждой R-последовательности $X(1)$, $X(2)$, ...,

[1] Заметим, что это упорядочение является «хронологическим» (временным в логическом времени).

[2] Независимой от длительностей процессов смены состояний в физическом времени.

[3] Детализации.

[4] Как правило, рассматриваются бинарные переменные, хотя нет никаких специальных ограничений на значность переменных. Так, в работе [19] рассматриваются многозначные самосинхронные коды.

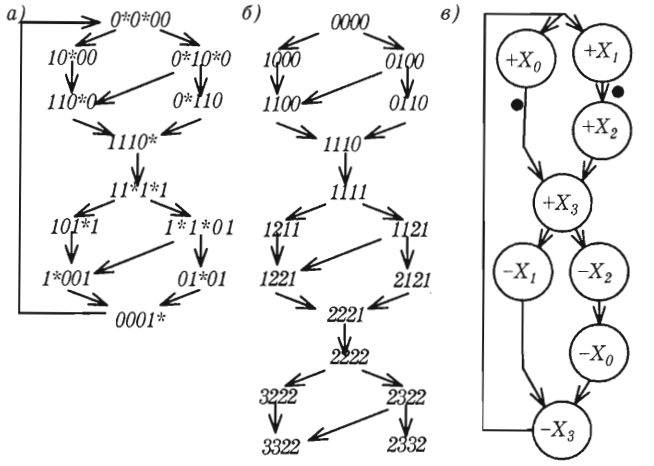
[5] Функционально связан.

[6] Кстати, уже здесь мы можем сделать предварительное замечание об отличии асинхронной модели от синхронной. В синхронной модели логическая функция элемента определяется как $x_i(t+1) = f_i(X(t))$, где t – специально выделенная переменная, сигнал логического времени.

[7] Возникла причина изменения состояния выхода.

[8] Полной схемы Маллера.

[9] Поскольку, как мы уже отмечали выше, одновременность в физическом времени недостижима, то в диаграмме отмечены только соседние переходы.



■ ***Рис. 4.*** *Диаграмма Маллера (а), кумулятивная диаграмма Маллера (б), сигнальный граф (в)*

$X(m)$ можно поставить в соответствие последовательность кумулятивных состояний (К-состояний) $\alpha(1)$, $\alpha(2)$, ..., $\alpha(m)$. Кумулятивное состояние $\alpha(j)$ есть вектор $\alpha(j)=[\alpha_0(j),\alpha_1(j),\ \ldots,\ \alpha_{n-1}(j)]$ с целочисленными компонентами $\alpha_i(j)$, такой что $\alpha_i(j)=0$ и $\alpha_i(j+1)=\alpha_i(j)+[x_i(j)\oplus f_i(X(j))]$. Иными словами, компонента кумулятивного состояния $\alpha(j)$ есть число изменений переменной x_i от начального состояния $\alpha(1)$ до состояния $\alpha(j)$. Кумулятивная диаграмма для примера рис. 4, *а* приведена на рис. 4, *б*. В кумулятивной диаграмме $\alpha\le\beta$, если $\alpha_i\le\beta_i\ \ \forall i$; $\gamma=\alpha\vee\beta$, если $\gamma_i=\max(\alpha_i,\beta_i)$; и $\gamma=\alpha\wedge\beta$, если $\gamma_i=\min(\alpha_i,\beta_i)$. При частичном упорядочении $\alpha\le\beta$ кумулятивная диаграмма есть структура. Маллером показано, что, если эта структура полумодулярна, то поведение соответствующей схемы не зависит от скорости и схема называется полумодулярной.

Диаграмма Маллера для полумодулярных схем имеет простое локальное свойство. Пусть $Q(X)$ – окрестность состояния X в диаграмме Маллера – $XRY\ \ \forall Y\in Q(X)$; $D(X(1))$ – множество состояний[1], достижимых из $X(1)$. Тогда для полумодулярности соответствующей схемы (кумулятивной диаграммы) необходимо и достаточно, чтобы, если $x_i=0^*(1^*)\,\forall X\in D(X(1))$, тогда $y_i\ne 0,1^*(1,0^*)\ \ \forall Y\in Q(X)$. Иными словами, для полумодулярности схемы необходимо и достаточно, чтобы каждый раз, когда выход логического элемента возбуждается (возникает причина для изменения выхода), возбуждение снимается только путем изменения выхода (причина сохраняется до тех пор, пока не реализуется ее следствие). Это свойство мы будем называть причинной обусловленностью.

Объем информации, содержащийся в диаграмме Маллера, вообще говоря, избыточен. Вместо графа смены состояний может быть использовано событийное описание поведения – частичный порядок на событиях, состоящих в смене значений переменных. Языками такого описания являются сигнальные графы (СГ), диаграммы изменений (ДИ), маркированные сети Петри (МСП) и др. [8, 9, 10]. Все эти языки обладают одинаковыми изобразительными возможностями. На рис. 4, *в* приведен СГ (ДИ)[2] для примера на рис. 4, *а*. Маркеры на дугах обозначают текущее состояние схемы. Для ДИ аналогично диаграмме Маллера может быть построена кумулятивная диаграмма и известен набор алгебраических свойств, обеспечивающих полумодулярность соответствующих схем. По ДИ однозначно восстанавливается диаграмма Маллера. Логические функции переменных задаются диаграммой Маллера следующим образом: если X – состояние рабочего цикла диаграммы Маллера[3], то если $x_i=0(1)$, тогда $f_i(X)=0(1)$, и если $x_i=0^*(1^*)$, тогда $f_i(X)=1(0)$ (это задание, вообще говоря, неоднозначно, так как функция может быть определена произвольно на состояниях, не входящих в рабочий цикл). Последнее обеспечивает задачу синтеза полумодулярных схем, спецификация которых задана ДИ или диаграммой Маллера. Задача синтеза, однако, этим не ограничивается по следующим причинам.

1. При переходе от ДИ к диаграмме Маллера в последней могут встретиться противоречивые состояния, т. е. состояния, в которых при одних и тех же булевых значениях возбуждены разные переменные. В этом случае для снятия противоречий необходимо введение дополнительных переменных, развязывающих противоречия. Естественно, что введение дополнительных переменных не должно нарушать свойство полумодулярности.

2. Синтез реальных схем всегда осуществляется в некотором функциональном базисе. Получающиеся в результате формального синтеза логические функции могут не входить в функциональный базис и должны быть представлены как суперпозиции базисных функций. При этом, опять-таки, схема должна сохранять свойство полумодулярности относительно выходов всех логических элементов.

Интересно отметить связь алгебраических свойств кумулятивной диаграммы Маллера и логических свойств соответствующих схем. Например, если кумулятивная диаграмма полумодулярна и дистрибутивна, то элемент 2И-НЕТ или 2ИЛИ-НЕТ является функционально полным в классе полумодулярных схем. Для недистрибутивных схем[4] функционально полный набор содержит элементы 2И-НЕТ и 2ИЛИ-НЕТ.

[1] Множество $D(X(1))$ иногда называют рабочим циклом.

[2] СГ являются подмножеством ДИ.

[3] Множество состояний, достижимых из начального.

[4] Содержательно недистрибутивность соответствует возникновению очередного возбуждения переменной более чем в одном состоянии.

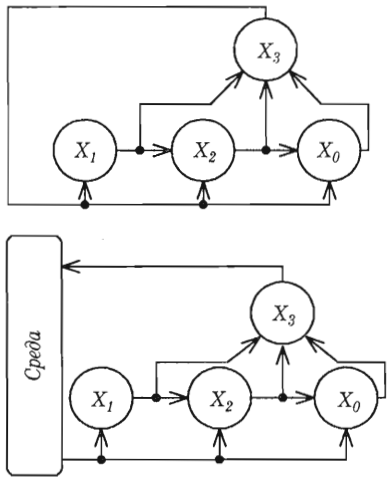
Как мы уже говорили выше, логическое время определяется частичным порядком на событиях, отражающим причинно-следственные отношения между ними. Для кумулятивной диаграммы Маллера момент логического времени в каждом состоянии может быть определен, в зависимости от семантики специфицируемого поведения, по крайней мере, тремя способами, а именно:

1) $T(\alpha)=\sum_{j=0}^{n-1}\alpha_j$, в случае, когда каждое событие, состоящее в смене значения переменной, формирует шаг логического времени;

2) $T(\alpha)=\max_j \alpha_j$, в случае, когда шаг логического времени формируется сменой значения «опережающей» переменной;

3) $T(\alpha)=\alpha_k$, в случае, когда шаг логического времени формируется сменой одной выделенной переменной[1].

Для ДИ и ее кумулятивной развертки определение моментов логического времени не столь очевидно. События ДИ определяют смену состояний и, следовательно, моменты логического времени естественно связать с дугами (маркерами). Комбинация маркеров однозначно определяет текущее состояние и, если сопоставить рис. 4, *а*, *б* и *в*, то нетрудно видеть, что для случая 1 событие $+X_0$ (возбужденное состояние 0*) присутствует в двух моментах логического времени. Таким образом, если в ДИ маркер на дуге $+X_1 \to +X_2$ имеет значение логического времени t, то маркер на дуге $+X_0 \to +X_3$ «размазан» по моментам времени $t \to t+1$. Таким образом, понимание логического времени для состояний и событий, вообще говоря, различно. Логическое время для состояния единственно для всей схемы в силу того, что в каждый момент физического времени схема находится в одном и только одном состоянии[2]. Логические времена для различных событий могут быть различны в один и тот же момент физического времени, и одно и то же логическое время для разных событий может быть в различные моменты физического времени. Интерпретация событийного логического времени зависит от соглашения, которое мы примем для этой интерпретации[3]. Например, если события $E_1(t_1)\&\ldots\&E_k(t_k)$ являются причиной события $E_m(t_m)$, мы можем считать, что $t_m=\max(t_1,\ldots,t_k)+1$ (напомним, что моменты логического времени маркируют выходные дуги соответствующих событий и нарушение непосредственного следования моментов времени на входных и выходных дугах одного события не является чем-то необычным). Для случая 3 в нашем примере, если выделена переменная x_3, то в каждом состоянии маркировок дуг все маркеры имеют одно и то же значение логического времени. Если же в качестве переменной, формирующей значения логического времени, выбрана переменная x_1 или x_2, то ситуация становится более сложной.



■ ***Рис. 5.*** *Схема Маллера для спецификации на рис. 4*

До сих пор мы рассматривали модели автономных схем и их поведение. Однако в реальных ситуациях любое устройство должно взаимодействовать и взаимодействует с внешней средой. Рассмотрим простейшую возможность организации такого взаимодействия.

На рис. 5, *а* приведена схема, специфицированная диаграммой Маллера (ДИ) на рис. 4. Согласно определению схемы Маллера, значения переменных x_i отнесены к выходам логических элементов и значения каждой переменной на входах других элементов идентичны в любой момент физического времени. Это приводит к соглашению Маллера о задержках: *все задержки приведены к выходу логического элемента и разбросом задержек в проводах после их разветвления можно пренебречь*[4]. Понятие независимости от скорости состоит в том, что поведение схемы в смысле сохранения частичного порядка на событиях не изменится, если последовательно выходу любого

[1] Выделенная переменная при этом может интерпретироваться как синхронизирующая и является в некотором смысле аналогом синхросигнала в системе с внешними часами.

[2] Это утверждение не совсем точно в силу того, что при смене состояния переменная проходит непрерывное множество значений и граница между значениями 0 и 1 размыта. Анализ режима переключения выходит за рамки этой статьи, однако сказанное может быть использовано с требуемой для наших целей степенью точности.

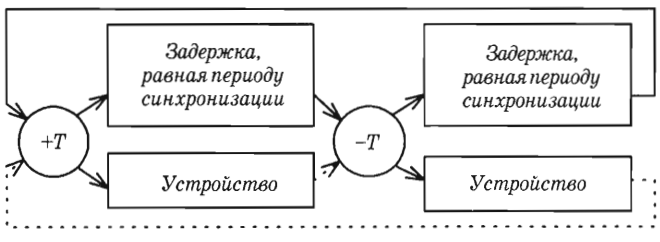
[3] В действительности интерпретация в значительной мере зависит от семантики описываемых процессов.

[4] Соглашение Маллера о задержках выполняется не всегда и, вообще говоря, зависит от разницы длин проводов после разветвления и конкретной топологии. Вопросы теории схем, не чувствительных к задержкам в проводах, выходят за рамки этой статьи и для нашего дальнейшего изложения соглашение Маллера является вполне удовлетворительным.

элемента включить произвольную задержку. В качестве такой задержки может быть включена внешняя среда (рис. 5, *б*). При этом мы можем ввести дополнительное обозначение для выходного сигнала среды e_i и, если среда включена последовательно выходу логического элемента x_i, то интерфейс между схемой и средой прост и очевиден $e_i = x_i$, $x_i = \overline{e}_i$. Моделью схемы с точки зрения среды является инвертор с произвольной последовательной задержкой. Сигнал e_i может рассматриваться как инициирующий (синхронизирующий) вход схемы, а сигнал x_i – как сигнал о завершении переходных процессов в схеме. Последнее, вообще говоря, не совсем точно. Если в качестве интерфейсного сигнала выбрана, например, переменная x_2, то, как следует из рис. 4, *в*, при $x_2 = 1$ может быть не завершено переключение переменной x_0, а при $x_2 = 0$ – переменной x_1. Завершение этих переключений проверяется в следующем цикле работы композиции схема–среда, и это открывает ряд новых возможностей, при которых часть внутренних переменных схемы переключается параллельно работе внешней среды[1].

Поскольку моделью разомкнутой полумодулярной схемы является инвертор с последовательной задержкой, то моделью разомкнутой полумодулярной схемы с последовательным инвертором является задержка. Это соображение открывает возможность композиции полумодулярных схем и их блочного синтеза, при котором разомкнутые полумодулярные блоки с последовательными инверторами включаются как задержки, последовательные выходам полумодулярной схемы, координирующей поведение этих блоков.

Синхронная схема с общими часами также может быть условно специфицирована ДИ (рис. 6). При этом, во-первых, моделью часов является калиброванная задержка и, во-вторых, если задержка, равная периоду синхронизации, заведомо больше любой возможной длительности переходных процессов в устройстве[2], то связями от событий в устройстве к событиям $\pm T$ можно пренебречь.



■ ***Рис. 6.*** *Сигнальный граф для случая синхронизации от внешних часов*

[1] Использование этих возможностей тесно связано с семантикой поведения.

[2] А это – непременное условие корректности поведения при синхронизации от общих часов.

Стратегия распределенной синхронизации

Мы должны отчетливо понимать, что система синхронизации поведения устройства от внешних часов включает в себя не только собственно часы, но и систему доставки сигналов от часов в точки их взаимодействия с устройством. Не будет преувеличением сказать, что, как только мы вводим в рассмотрение не только часы, но и систему доставки сигналов времени[3], мы переходим к рассмотрению проблемы пространства-времени. Эта проблема для искусственных систем имеет свою специфику[4]. В VLSI and VLSI-системах мы фактически имеем дело с проводной метрикой, заданной на графе связей между модулями. Два модуля A и B являются соседними относительно сигнала x_i, если существует провод, передающий сигнал x_i, от модуля A к модулю B. Расстояние[5] от модуля A до модуля B относительно сигнала x_i $D(AB_{x_i})$ есть время, в течение которого сигнал x_i распространяется от модуля A до модуля B. Заметим при этом, что если сигналы x и y распространяются от A к B по разным проводам, то $D(AB_x) \neq D(AB_y)$.

Задержка распространения сигнала по проводу определяется двумя факторами: во-первых, распределенными параметрами RC (постоянная времени $\tau_1 = RCl^2/2 \approx 100$ пс/мм)), во-вторых, ограничениями на плотность тока в проводе. Плотность тока[6], приводящая к миграции атомов, для алюминия равна 10^5 А/см^2 = 1 мА/μ^2 и время заряда емкости провода постоянным током[7] определяется как $\tau_2 = VCl/I \approx 100$ пс/(мм·мА). Если доминирующим фактором является плотность тока, т. е. если $\frac{\tau_1}{\tau_2} = \frac{Rll}{2V} << 1$, то поверхность провода можно считать эквипотенциальной обкладкой конденсатора. В этом случае говорят об эквипотенциальной или эквихронной зоне. Задержки, вообще говоря, существенно больше, так как провод нагружен на входы транзисторов, входные емкости которых на единицу площади в 50–100 раз больше, чем у провода. Оценка размеров эквихронной зоны достаточно сложна и зависит от принятых гипотез и критериев, однако по общим оценкам эти размеры имеют порядок 1 мм. Из понятия эквихронной зоны, независимо от ее размеров, следуют два важных вывода:

[3] Заметим, что не только времени, но и информационных сигналов.

[4] В частности, для VLSI и VLSI-систем, с которыми мы и будем далее иметь дело.

[5] Это совсем не экзотический способ измерения расстояний. Например, на дорожных картах Новой Зеландии расстояние между городами приводится в часах и минутах for driving.

[6] Это ограничение для постоянного тока. Для знакопеременных токов допустимое значение плотности может быть в несколько раз больше.

[7] При увеличении ширины провода в равной мере увеличивается его емкость и допустимый ток.

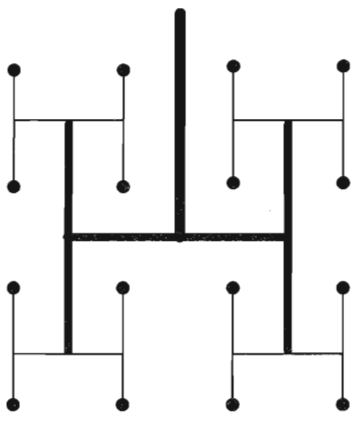
1) совокупность эквихронных зон порождает совокупность локальных времен;

2) система доставки сигналов времени должна обеспечивать взаимную координацию локальных времен у взаимодействующих эквихронных зон.

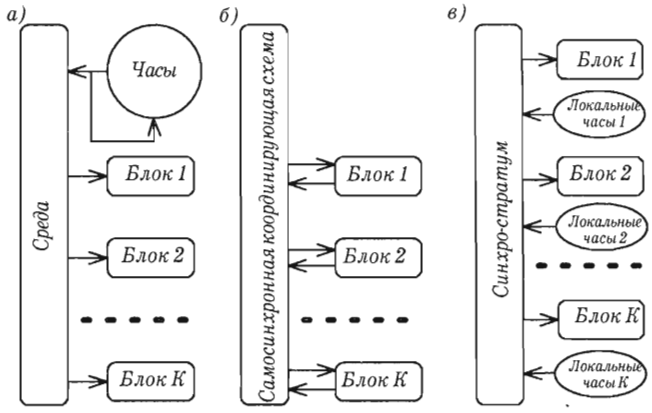
Использование общих часов для синхронизации поведения VLSI предъявляет соответствующие требования к системе доставки сигналов времени. Как правило, проводная система доставки сигналов времени от общих часов представляет собой H-дерево (рис. 7), в котором точки ввода сигнала времени в эквихронную зону равноудалены от общего источника этих сигналов. Балансировка H-дерева для современных скоростей синхронизации (2–4 ГГц) и современных размеров VLSI представляет собой весьма непростую задачу и каждый новый шаг на этом пути дается все большей и большей кровью.

Основные проблемы, возникающие при всех модификациях H-дерева, следующие: разброс физических и технологических параметров, приводящий к разбросу задержек, и большая мощность, необходимая для перезарядки проводов H-дерева[1]. Эти проблемы, вообще говоря, ограничивают число терминальных точек в H-дереве и степень дробления системы на модули, которые рассматриваются как эквихронные.

Вернемся к рассмотрению механизма работы часов. Любые известные часы, от солнечных и водяных до точнейших атомных, имеют в своей основе какой-либо периодический процесс, протекающий в физическом времени. Часы, используемые в VLSI, представляют собой генератор (clock generator) с калиброванной задержкой в цепи отрицательной обратной связи (рис. 8, *а*)[2]. Какие требования предъявляются к периоду генерации? Только одно – длительность периода должна быть достаточной для завершения переходных процессов и в рамках этого ограничения точность поддержания периода не имеет существенного значения. Более того, период вообще может быть эластичным, если мы можем управлять задержкой в зависимости от режима работы VLSI (задержка, зависящая от данных, от параметра и т. д.) или тем или иным способом фиксировать моменты окончания переходных процессов (рис. 8, *б*). Однако, как мы уже говорили выше, камнем преткновения является система распределения и доставки сигналов времени. При этом очевидно, что координироваться между собой должны только локальные сигналы времени у соседних по графу связей модулей. Это приводит нас к идее децентрализованной синхронизации (рис. 8, *в*). Другими словами, мы попытаемся показать, что могут быть созданы распределенные, асинхронные часы, в которых пассивная проводная система доставки синхросигналов заменяется активной средой. Такое устройство будем называть синхро-стратумом.



■ *Рис. 7. H-дерево*



■ *Рис. 8. Способы синхронизации*

Синхро-стратум

В качестве модели для следующего шага нашего изложения мы будем рассматривать одномерный клеточный массив из N *синхронных* автоматов Мура A_j (модулей, блоков), в котором каждый автомат A_j связан информационным обменом только с двумя своими непосредственными соседями (A_{j-1}, A_{j+1}). Если $S_j(t)$ – состояние автомата A_j в момент логического времени t, то

$$S_j(t+1) = F_j[S_{j-1}(t), S_j(t), S_{j+1}(t)]. \qquad (1)$$

Заметим[3], что если поведение массива задано системой логических уравнений (1), то известными преобразованиями пространства состояний и функций могут быть построены массивы с эквивалентным поведением:

[1] Так, при 32 эквихронных зонах и частоте 400 МГц только на перезаряд емкости проводов H-дерева требуется около 2,5 Вт.

[2] Мы пока здесь будем рассматривать изолированное (автономное) устройство. Взаимодействие с внешней средой представляет собой специальный вопрос.

[3] И это нам будет полезно в дальнейшем.

$$S_j(t+1) = F_j[S_{j-1}(t+1), S_j(t), S_{j+1}(t)];$$
$$S_j(t+1) = F_j[S_{j-1}(t), S_j(t), S_{j+1}(t+1)];$$
$$S_j(t+1) = F_j[S_{j-1}(t+1), S_j(t), S_{j+1}(t+1)]. \quad (2)$$

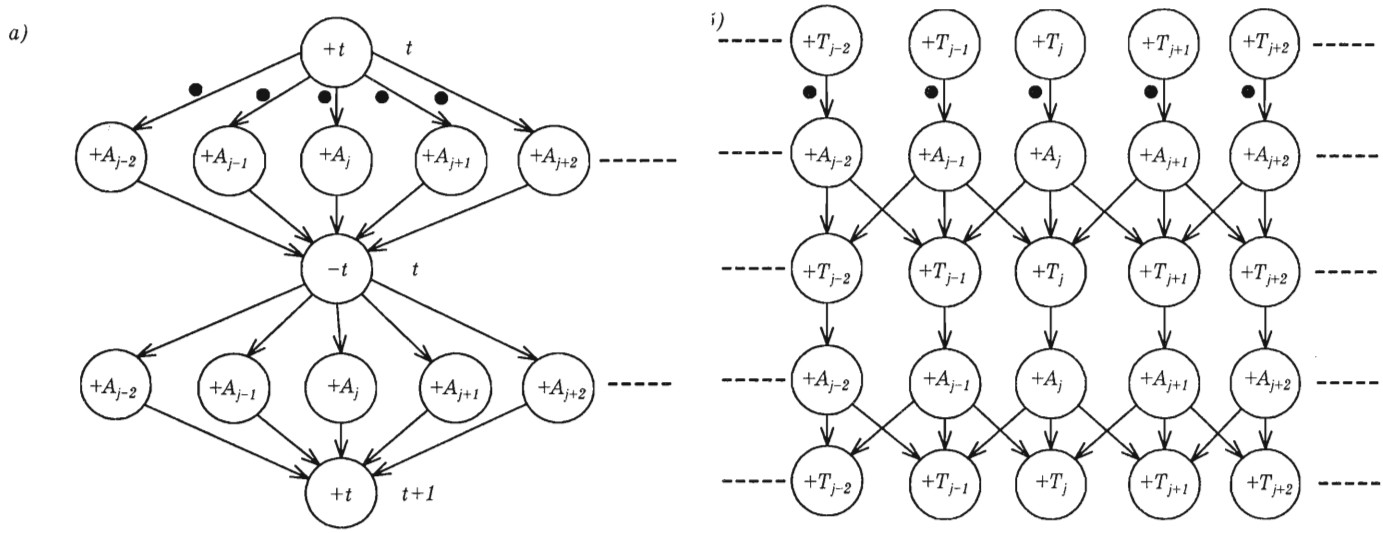
В уравнениях (1) и (2) t – целочисленная переменная, представляющая глобальное (общее для всего массива) логическое время. Введем понятие локального логического времени T_j, представляющего значение глобального логического времени на входе автомата A_j. При этом уравнение (1) имеет вид

$$S_j(T_j+1) = F_j[S_{j-1}(T_{j-1} = T_j), S_j(T_j), S_{j+1}(T_{j+1} = T_j)]. \quad (3)$$

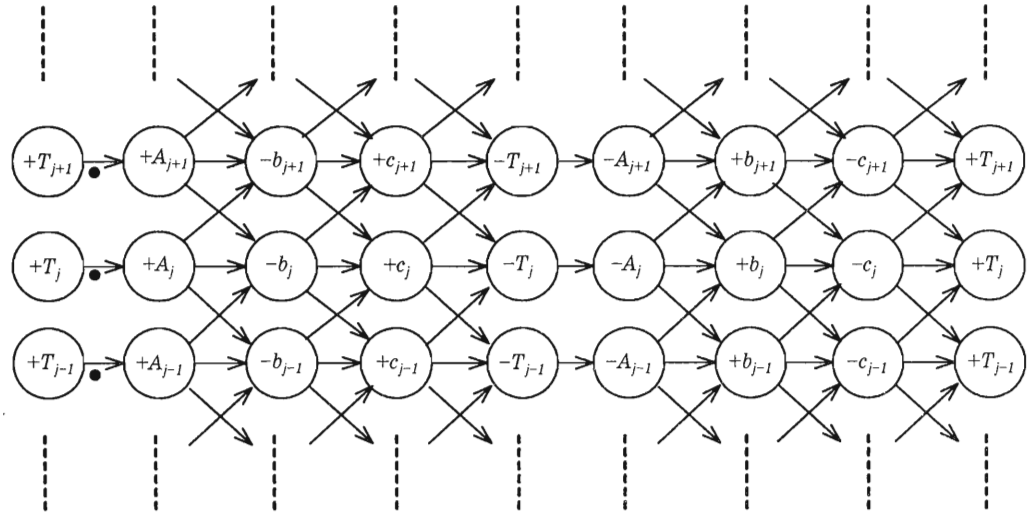
Из уравнения (3) с очевидностью следует, что для обеспечения корректности временного поведения массива нам достаточно координировать значения локальных логических времен только у ближайших соседей. На рис. 9, *а* приведена ДИ для глобальной синхронизации массивов от общего синхросигнала t, а на рис. 9, *б* – ДИ для эквивалентной координации локальных времен.

ДИ на рис. 9, *б* корректна, однако ей не соответствует никакая схема, так как соответствующая ДИ диаграмма Маллера содержит противоречивые состояния. Для снятия противоречий в ДИ должны быть введены дополнительные переменные, как, например, на рис. 10.

Синтезированная по этой спецификации схема имеет вид



■ ***Рис. 9.*** *Диаграмма изменений для синхро-стратума*



■ ***Рис. 10.*** *Диаграмма изменений для реализации синхро-стратума на рис. 9*

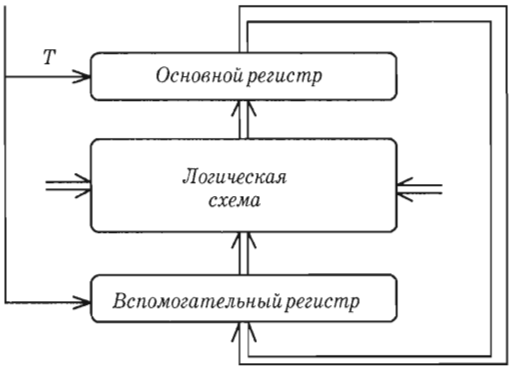
$$T_i = \overline{c_{i-1}c_ic_{i+1} + b_i(c_{i-1} + c_i + c_{i+1})},$$
$$c_i = \overline{b_{i-1}b_ib_{i+1} + T_i(b_{i-1} + b_i + b_{i+1})}, \quad (4)$$
$$b_i = \overline{a_{i-1}a_ia_{i+1} + c_i(a_{i-1} + a_i + a_{i+1})},$$

что, собственно, принципиально решает задачу о возможности реализации синхро-стратума. Однако может быть приведен ряд аргументов в пользу более детального анализа реализации синхро-стратума.

Во-первых, схема (4) достаточна сложна. Схема для каждой точки синхронизации содержит 42 транзистора в CMOS реализации[1] и, что более важно, она требует шесть входящих проводов от соседних точек синхронизации и столько же исходящих проводов к ним[2].

На практике используются различные системы синхронизации, включающие несколько синхросигналов (синхросерий), и каждый синхронный прототип приводит к своей структуре синхро-стратума.

Основная идея синхронизации сигналами от часов связана тем или иным образом с организацией поведения блоков схемы по принципу «основной–вспомогательный». Стратегии синхронизации мы будем рассматривать на модели массива из клеточных автоматов, автоматы которой являются автоматами Мура, построенными на основе двухрегистровой схемы, работающей по принципу «основной–вспомогательный», или «ведущий–ведомый» (рис. 11). При одном значении сигнала T от часов автомат изменяет свое текущее состояние путем записи нового состояния с выходов логической схемы в основной регистр (рабочая фаза). При другом значении T текущее состояние не изменяется, а только переписывается из основного регистра во вспомогательный (пассивная фаза). Такой принцип работы называется двухфазным с разнополярным управлением. В рабочей фазе неизменным является состояние вспомогательного регистра, хранящего предыдущее состояние автомата, а в пассивной фазе неизменно состояние основного регистра, хранящего текущее состояние.



■ *Рис. 11. Структура автомата с двухфазным разнополярным управлением*

Существенным фактором для временно́й координации поведения является неизменность выходов соседних автоматов (входов данного автомата) в рабочей фазе. В работах [11–15] описано несколько реализаций синхро-стратума. Здесь, в дополнение к схеме (4), мы приведем только одно новое решение. Рассмотрим фрагмент ДИ на рис. 12.

Схема, синтезированная по этой ДИ, имеет вид

$$T_i = \overline{T_{i-1}T_{i+1}(C_{i-1} + C_{i+1})},$$
$$C_i = \overline{C_{i-1}A_iC_{i+1}}. \quad (5)$$

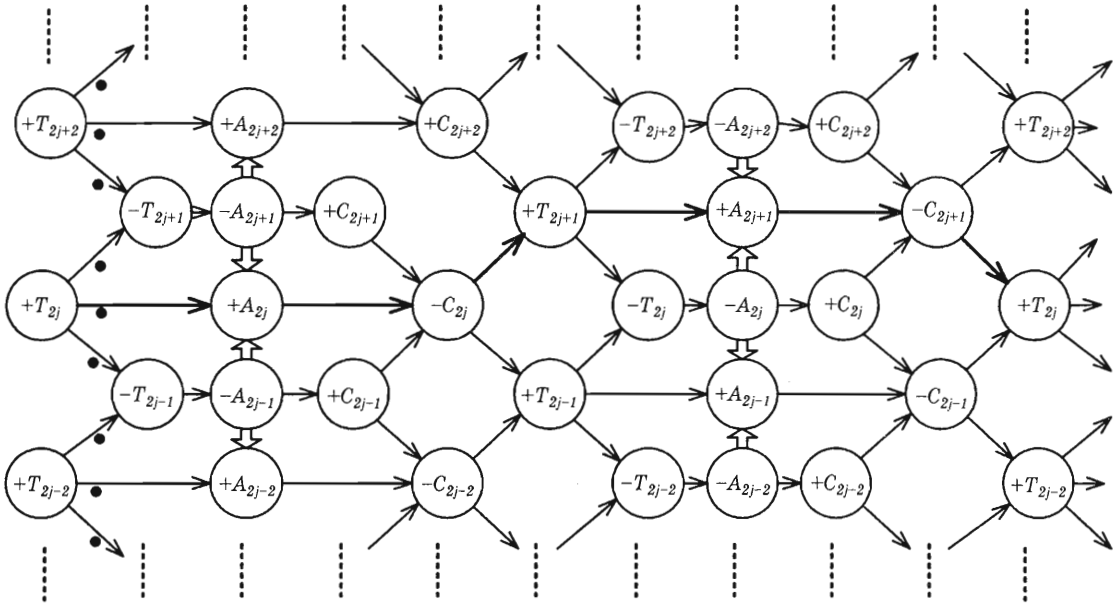
Из схемы (5) следует, что этот синхро-стратум имеет сложность 14 транзисторов на точку синхронизации в CMOS реализации, четыре входящих и четыре выходящих провода для интерфейса между ячейками синхро-стратума. По этим параметрам он проигрывает лучшему из известных [15] (см. рис. 5) синхро-стратумов[3]. Однако схема (5) обладает одним замечательным свойством – предельным быстродействием. Действительно, если мы обратимся к структуре автомата «основной–вспомогательный», то заметим, что длительности переходных процессов в разных фазах, вообще говоря, различны. В рабочей фазе длительность переходного процесса складывается из задержки логической схемы и задержки записи в регистр. В пассивной фазе переходный процесс содержит только запись в регистр. Пусть в ДИ на рис. 12 рабочий цикл в автоматах реализуется при $T_j = 1$ (знаки $\Uparrow$ и $\Downarrow$ в ДИ обозначают время и направление передачи информации с основных регистров автоматов A_{j-1} и A_{j+1} на логическую схему автомата A_j) и пусть задержка в логической схеме $\tau_{logic} = 2\tau_{gate}$. Тогда полный цикл синхронизации определяется путем, выделенным на рис. 12 жирными стрелками. Нетрудно видеть, что дополнительная задержка, вносимая синхро-стратумом, равна $4\tau_{gate}$ на полный цикл изменения T_j (заметим, что для минимальной схемы [15; рис. 5] дополнительная задержка, вносимая синхро-стратумом, равна $10\tau_{gate}$ и задержка для схемы (5), по-видимому, минимальна).

Мы будем считать, что полный цикл смены значений локального сигнала времени $\rightarrow -T_j \rightarrow +T_j$ представляет собой один шаг локального логического времени, и введем обозначения $1(k)$ и $0(k)$ для значений $T_j(k) = 1$ и $T_j(k) = 0$ соответствен-

[1] Дополнительная задержка, вносимая синхро-стратумом, равна 6τ на полный цикл синхронизации, где τ – задержка одного вентиля.

[2] С ростом числа соседей пропорционально возрастает число проводов и соответственно число транзисторов.

[3] Десять транзисторов, один входящий и один выходящий провод для внутристратумного интерфейса.



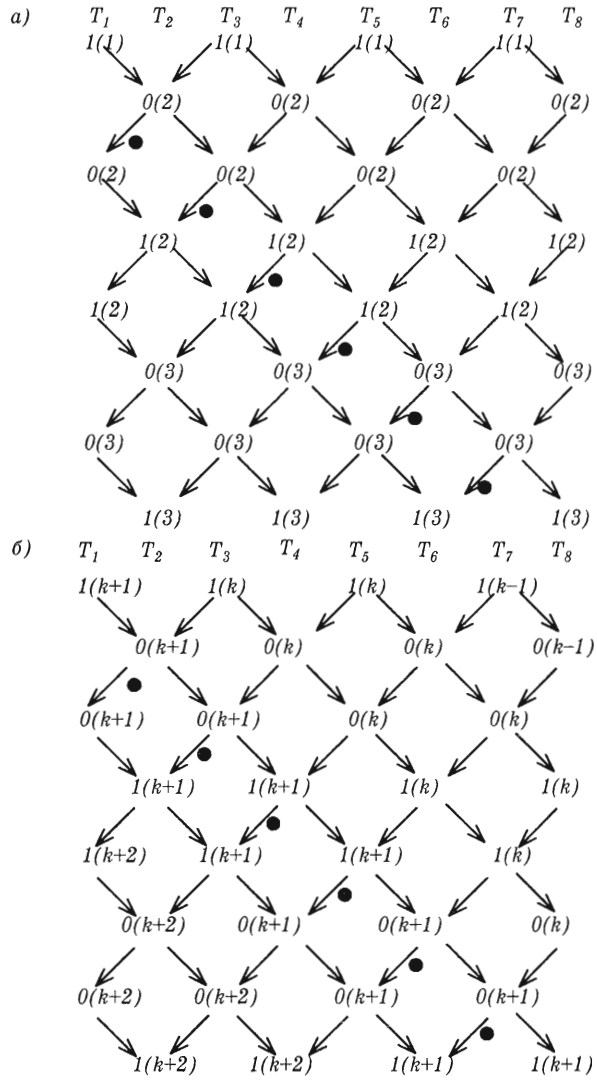
■ ***Рис. 12.*** *Диаграмма изменений для быстрого синхро-стратума*

но. Пусть в начальный момент времени $T_j(1)=1$ для любого j. Тогда проекция кумулятивной развертки ДИ, представленной на рис. 12, имеет вид рис. 13, *а*. При этом автоматные уравнения различны для четных и нечетных автоматов и имеют вид

$$S_{2j}(t+1)=F_{2j}[S_{2j-1}(t),S_{2j}(t),S_{2j+1}(t)],$$
$$S_{2j+1}(t+1)=F_{2j+1}[S_{2j}(t+1),S_{2j+1}(t),S_{2j+2}(t)]. \quad (6)$$

Размышления над диаграммой на рис. 13, *а* позволяют получить ряд любопытных соображений о временном поведении системы. Если мы зафиксируем автомат A_1 в состоянии 1(1), то система[1] придет в устойчивое состояние с распределением логических времен: 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, ... Из этого следует, что как в устойчивом состоянии, так и в процессе функционирования[2] в разных точках синхро-стратума могут существовать разные логические времена. Более того, если мы можем управлять ритмом работы одного из автоматов или соответствующей ячейкой синхро-стратума, то эта точка массива будет генератором ритма для всего массива и от нее по массиву будут распространяться «волны времени». Мы можем попытаться организовать поведение так, чтобы в устойчивом состоянии и на фронте «волны времени» значение логического времени было одинаковым для всех автоматов. Такую возможность открывает диаграмма на рис. 13, *б*. В этой диаграмме для нечетных автоматов шаг логического времени формируется циклом $\to +T_j \to -T_j$, а для четных – циклом $\to -T_j \to +T_j$. При этом, как

[1] Как это видно из распределения маркеров на рис. 13, *а*.
[2] При вариациях длительностей переходных процессов у разных автоматов.



■ ***Рис. 13.*** *Локальное распределение логического времени*

это видно из рис. 13, *б*, автоматные уравнения имеют период 4:

$$S_{4j}(t+1) = F_{4j}[S_{4j-1}(t), S_{4j}(t), S_{4j+1}(t)],$$
$$S_{4j+1}(t+1) = F_{4j+1}[S_{4j}(t), S_{4j+1}(t), S_{4j+2}(t)],$$
$$S_{4j+2}(t+1) = F_{4j+2}[S_{4j+1}(t), S_{4j+2}(t), S_{4j+3}(t)],$$
$$S_{4j+3}(t+1) = F_{4j+3}[S_{4j+2}(t), S_{4j+3}(t), S_{4j+4}(t)]. \quad (7)$$

Таким образом, меняя вид автоматных уравнений, можно переходить от параллельной к волновой синхронизации.

Переход от одномерного к двумерному массиву связан только с увеличением числа соседей и уравнения (5) приобретают вид

$$T_{i,j} = \overline{T_{i-1,j}T_{i,j-1}T_{i+1,j}T_{i,j+1}(C_{i-1,j} + C_{i,j-1} + C_{i+1,j} + C_{i,j+1})},$$

$$C_{i,j} = \overline{C_{i-1,j}C_{i,j-1}A_{i,j}C_{i+1,j}C_{i,j+1}}. \quad (8)$$

Синхро-стратум может быть синтезирован и для произвольного графа связей. Логические функции для его элементов при этом обобщаются естественным образом на случай произвольного графа. Пусть V_j – множество индексов окрестности автомата A_j на графе связей, т. е. $V_j = \{i\}$ для любых i таких, что A_i имеет информационную связь с A_j. Тогда функции элементов сихро-стратума имеют вид:

$$C_j = \overline{A_j \prod_{i \in V_j} C_i}\ ; T_j = \overline{\prod_{i \in V_j} T_i \prod_{i \in V_j} C_j}\ . \quad (9)$$

Для организации взаимодействия типа[1] рис. 12 (9) необходимым и достаточным условием корректности поведения синхро-стратума является бихроматичность графа связей [12]. Однако для синхро-стратума типа рис. 10 (4) такого ограничения на граф связей нет и функции элементов имеют вид[2]

$$T_j = \overline{\prod_{i \in V_j,\ i=j} c_i + b_j \bigcup_{i \in V_j,\ i=j} c_i},$$

$$c_j = \overline{\prod_{i \in V_j,\ i=j} b_i + T_j \bigcup_{i \in V_j,\ i=j} c_i},$$

$$b_j = \overline{\prod_{i \in V_j,\ i=j} a_i + c_j \bigcup_{i \in V_j,\ i=j} a_i}. \quad (10)$$

[1] Равно как и для всех других, описанных ранее типов синхро-стратума.

[2] Дополнительной проблемой здесь, как и в случае (9), является отображение – представление функции как суперпозиции базисных элементов, что выходит за рамки этой статьи. Заметим только, что для случая (9) эта задача принципиально проще.

Заключение

Теперь уместно задать себе несколько вопросов. Первый из них: «Что дало нам введение синхро-стратума?» Известно, что декомпозиция как таковая не может открыть никаких новых функциональных возможностей. Действительно, мы всегда можем рассматривать композицию автомата A_j со множеством внутренних состояний $\{S_{ij}\}$ и ячейки синхро-стратума T_j со множеством внутренних состояний $\{X_{ij}\}$ как один автомат W_j со множеством внутренних состояний $\{Y_{ij}\} = \{S_{ij}\} \times \{X_{ij}\}$ и соответствующей функцией переходов. Более того, используя изощренные процедуры размещения состояний и минимизации автоматов, мы можем вообще завуалировать наличие синхро-стратума в исходном представлении. Однако при этом для каждого массива клеточных автоматов мы должны решать задачу заново. Весь смысл декомпозиции системы на автоматный стратум и синхро-стратум состоит в создании возможности разделения задач организации функционального и временного поведений[3]. Именно такое разделение позволи́ло нам фактически доказать[4], что по любому синхронному прототипу массива клеточных автоматов единообразным путем может быть построен его асинхронный аналог. Это инженерный аспект решаемой задачи. С концептуальной точки зрения синхро-стратум является носителем логического времени и создает в системе поле локальных логических времен. Понятие синхро-стратума открывает широкие и еще до конца не использованные возможности для размышлений и спекуляций на тему искусственного системного времени.

Второй, не менее важный вопрос: «Является ли наличие синхронного прототипа и синхро-стратума достаточным для построения асинхронного аналога?» На этот вопрос с определенностью можно ответить: «Нет!» Действительно, асинхронное поведение должно быть причинно-обусловленным, а это означает, что синхронизируемые блоки (автоматы) должны иметь интерфейс с синхро-стратумом, обеспечивающий прямое или опосредованное (косвенное) взаимодействие типа «запрос–ответ» (см. рис. 8). В этом интерфейсе сигнал локального логического времени может и должен рассматриваться как сигнал «запрос» и, получив его, блок (автомат) должен формировать сигнал «ответ», разрешающий синхро-стратуму формировать следующий шаг локального времени. Как и какие дополнения или изменения должны быть сделаны, чтобы блок синхронного прототипа мог быть использован во взаимодействии с синхро-стратумом, – это вопрос конкретного проектирования. Проектировщик может

[3] Такое разделение является одной из привлекательных черт синхронных моделей.

[4] Хотя такое утверждение в явном виде в статье не содержится.

использовать широкий спектр возможных решений, от самосинхронизации до параллельной встроенной задержки. Важно одно: задача может быть решена децентрализовано независимыми и, быть может, различными способами для разных блоков. Такая свобода в решении задачи организации локального временного интерфейса позволяет наряду с известными системами синхронизации: FS – полностью синхронная (общие часы), FA – полностью асинхронная (самосинхронная и др.), LAGS – локально асинхронная – глобально синхронная, GALS – глобально асинхронная – локально синхронная, выделить еще одну: GALA – глобально асинхронная – локально произвольная (система с синхро-стратумом).

И, наконец, последний вопрос: «Может ли любая задача, решаемая синхронным массивом из клеточных автоматов, быть решена соответствующим массивом с синхро-стратумом?».

Ответ на этот вопрос связан с трактовкой и пониманием понятия системного времени. Одна из трудных задач для асинхронного массива клеточных автоматов – это задача взаимодействия фронтов волн, распространяющихся во встречных направлениях, при реализации волновых алгоритмов. Хотя эта задача в принципе решаема[1], при попытках ее прямого решения в структуре асинхронных массивов[2] авторам не всегда удается избежать конфликтных[3] ситуаций. Введение синхростратума снимает проблему асинхронного проектирования и, если алгоритм взаимодействия реализуем в синхронном прототипе, то он автоматически реализуем в асинхронном варианте. Однако, поскольку в синхронном варианте одновременность в физическом времени понимается как одновременность в логическом времени[4], то, по-видимому, задачи, так или иначе связанные с синхронизацией в физическом времени, в принципе не могут быть перенесены из синхронных массивов на асинхронные. Типичным примером является задача Дж. Майхилла о синхронизации цепи стрелков [16, 17]. Для синхронного массива клеточных автоматов задача формулируется следующим образом.

Имеется одномерный массив из автоматов Мура с двухсторонним взаимодействием. Каждый из N автоматов имеет n внутренних состояний (n не зависит от N) и в начальный момент времени $T = 0$ все автоматы находятся в пассивном состоянии S_0. В момент времени $T = 1$ на крайний автомат массива поступает внешний инициализирующий сигнал. В момент времени $T = 2N$ все автоматы должны одновременно перейти в финальное состояние[5] S_f.

Все известные решения этой задачи базируются на последовательном делении отрезков массива пополам путем сравнения скоростей распространения сигналов. Поскольку скорости распространения сигналов (задержки сигналов в автоматах) измеряются в единицах логического времени, то переход от синхронного прототипа к асинхронному массиву прост и очевиден. Однако момент синхронизации (перехода каждого автомата в финальное состояние) происходит при значении его локального времени, равном $2N$, а мы уже отмечали выше, что в асинхронном массиве одни и те же значения локального логического времени могут существовать в различные моменты физического времени, и задача теряет свой первоначальный смысл.

Приведенная задача теряет свой первоначальный смысл для внешнего наблюдателя, являющегося частью внешнего физического мира с единым физическим временем. Однако, если внешним наблюдателем является искусственная система со своей логической организацией, то могут быть найдены осмысленные интерпретации асинхронного в физическом времени решения этой задачи. Так или иначе проблема синхронизации цепи стрелков демонстрирует те трудности, которые возникают или могут возникнуть при организации в реальном времени интерфейса между искусственной системой и внешним физическим миром. Но это уже тема другой статьи.

Автор пользуется приятной возможностью выразить свою искреннюю признательность профессору К. Неханиву (Университет Айдзу, Япония) за приглашение прочитать на рабочем семинаре «Полугруппы и инженерная алгебра» лекцию, которая легла в основу этой статьи, доктору Т.А. Чу (Acorn Networks Inc., USA) и профессору В. Мараховскому (Университет Айдзу, Япония), с которыми автор в течение нескольких лет работал над проблемами синхро-стратума, а также В. Смоленскому (Университет Айдзу, Япония) за огромную помощь в подготовке этой статьи.

[1] И это следует из нашего изложения (см. также [15]) .
[2] Как, например, в случае counter-flow архитектуры [18].
[3] Арбитражных.
[4] С точностью до длительности синхротакта.
[5] Вся цепь стрелков должна одновременно выстрелить.

Литература

1. **Reichenbach H.** The Direction of Time / Ed. M. Reichenbach. – University of California Press, Berkeley and Los Angeles, 1958.
2. **Reichenbach H.** The philosophy of space & time. – New York, 1958.
3. Von Wrigt G.H. Explanation and Understanding. – London, 1971.
4. **Yang Z., Marsland T.** Global States and Time in Distributed Systems. – IEEE Computer Society Press, Los Alamos, Ca., 1994.
5. **Muller D. E.** A theory of asynchronous circuits // Report of University of Illinois. – 1955. – N 66.
6. **Muller D. E., Bartky W. C.** A theory of asynchronous circuits. I. IL // Report of University of Illinois. 1956. – N 75; 1957. – N 78.
7. **Miller R. E.**, Switching Theory. Sequential circuits and machines. Vol. II. – John Wiley & Sons. Inc., 1966.
8. **Varshavsky V.** Hardware support of parallel asynchronous processes. – Helsinki University of Technology, Digital Systems Laboratory, Series A: Research Reports. – N 2, Sept. 1987.
9. Self-timed control of concurrent processes / Ed. V. Varshavsky. – Kluwer Academic Publishers, 1990.
10. **Kishinevsky M., Kondratiev A., Taubin A., Varshavsky V.** Concurrent hardware. – John Wiley & Sons, 1994.
11. **Varshavsky V., Chu T. A.** Self-timing – tools for hardware support of parallel, concurrent and event-driven process control // Proceedings of the Conference on Massively Parallel Computing Systems (MPCS'94), May 1994. – P. 510–515.
12. **Varshavsky V., Marakhovsky V., Chu T. A.** Logical timing (Global synchronization of asynchronous arrays) // Parallel Algorithm/Architecture Synthesis International Symposium, Aizu-Wakamatsu, Japan. IEEE CS Press. – March 1995. – P. 130–138.
13. **Varshavsky V.** asynchronous interaction in massively parallel computing systems // IEEE First ICA3PP, Proceedings of IEEE First International Conference on Algorithms and Architectures for Parallel Processing, Vol. 2. – Brisbane, Australia. – P. 951–953.
14. **Varshavsky V., Marakhovsky V., Chu T. A.** Asynchronous timing of arrays with synchronous prototype// Proceedings of the Second International Conference on Massively Parallel Computing Systems (MPCS'96), May 1996. – P. 47–54.
15. **Varshavsky V., Marakhovsky V.** Global synchronization of asynchronous arrays in logical time // Proceedings of the Second Aizu International Symposium on Parallel Algorithm / Architecture Synthesis, Aizu-Wakamatsu, Japan. IEEE CS Press, March 1997. P. 207–215.
16. **Goto E.** A minimum time solution of the firing squad problem // Dittoed course notes for applied mathematics 298. – Harvard University, May 1962.
17. **Varshavsky V., Marakhovsky V., Peschansky V.** Synchronization of Interacting Automata, Math. System Theory. – 1970. – Vol. 4. – N 3.
18. **Sproull R. F., Sutherland I. E., Molnar C. E.** Counter flow pipeline processor architecture // Technical Report SMLI TR-94-25, SUN Microsystems Laboratories Inc. CA 94043, April 1994.
19. **Varshavsky V.** Logic design and quantum challenge // Proceedings of International Workshop on Physics and Computer Modeling of Devices Based on Low-Dimensional Structures. Aizu-Wakamatsu, Japan. IEEE CS Press. – Nov. 1995.

УДК 681.3

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ И КВАНТОВЫЙ ВЫЗОВ[1]

В. И. Варшавский,
доктор техн. наук, профессор
Университет Айдзу (Япония)

*Чистая наука решает задачи, которые **можно** решить так, как их **нужно** решать;*
*прикладная наука решает задачи, которые **нужно** решить, так, как их **можно** решать.*
Математический фольклор

Готова ли методология проектирования логических и вычислительных структур к эффективному использованию новых функциональных возможностей квантовых устройств? Обеспечивает ли ряд существующих и предлагаемых квантовых устройств эффективное проектирование логических и вычислительных структур? На основе примеров обсуждается возможность получения положительных ответов на эти вопросы за счет объединенных усилий физиков, технологов и экспертов в вычислительной технике. Статья указывает возможные направления, где такое взаимодействие может оказаться продуктивным.

Is the design methodology of logical and computing structures ready to efficiently use new functional possibilities of quantum devices? Does the range of existing and suggested quantum devices provide effective design of logical and computing structures? With some examples, we discuss the possibility of obtaining positive answers to these questions uniting the efforts of physicists, technologists and experts in computer engineering. The paper indicates some directions where such collaboration may prove to be productive.

Прогресс в вычислительной технике в значительной степени связан с беспрецедентными успехами VLSI-технологии. Одной из основных компонент прогресса VLSI-технологии является постоянное уменьшение размеров конструктивных элементов VLSI. Однако при размерах конструктивных элементов меньше 0,1 мкм в их поведении начинают проявляться квантовые эффекты, что, в свою очередь, изменяет количественные и качественные характеристики их поведения. Естественно, что изменение физического поведения функциональных компонент приводит к изменению логических возможностей их использования. С одной стороны, возникают новые логические возможности, а с другой стороны, реализация некоторых традиционных подходов становится затруднительной и неэффективной. Развитие нанотехнологии и очевидные успехи в создании и исследовании квантовых функциональных устройств [1–10][2] сделали практическое использование этой технологии и этих устройств в вычислительной технике ближайшей реальностью. История развития вычислительной техники показывает, что появление новых технологий и использование новых физических принципов обычно порождает эйфорию (как правило, оправданную) у их создателей. Первые устройства создаются обычно самими технологами. Кажется, что новые возможности неисчерпаемы. Действительно, переход от микротехнологии к нанотехнологии позволяет увеличить сложность VLSI в десятки и сотни раз. Однако вспомните достаточно популярный в 70-е годы среди технологов лозунг – «Кремния не жалеть!». Отрезвление пришло до-

[1] **Varshavsky V.** Logic Design and Quantum Challenge», Proceedings of the International Workshop on Physics and Computer Modeling of Devices Based on Low-Dimensional Structures, November 7–9, 1995; Aizu-Wakamatsu, Japan, P. 119–131.

[2] Библиография не претендует на полноту; выбор ссылок достаточно случаен и они лишь иллюстрируют изложение.

вольно быстро и практика реального проектирования потребовала применения достаточно изощренных методов логического проектирования. Простой пример: при 10 %-ном выходе годных кристаллов уменьшение площади на 10 % за счет применения продвинутых методов логического проектирования увеличивает выход годных кристаллов примерно в 1,25 раза. Возникают два естественных вопроса:

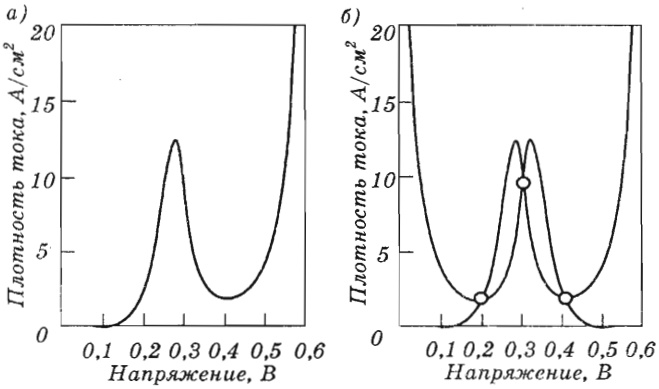
Готова ли методология проектирования логических и вычислительных структур к эффективному использованию новых функциональных возможностей квантовых устройств?

Обеспечивает ли спектр существующих и предлагаемых квантовых устройств эффективное проектирование логических и вычислительных структур?

С определенной степенью уверенности на оба эти вопроса можно ответить и «да», и «нет». Обсуждению необходимости и путей объединения усилий специалистов по логическому проектированию и специалистов по технологии и физике для увеличения уверенности в утвердительном ответе посвящена эта работа. Рано или поздно реальная практика проектирования потребует адаптации методов логического проектирования к новым технологиям. Лучше начать эту работу рано, чем поздно. К сожалению, ниже будет больше вопросов, чем ответов на них, но мы находимся только в начале того пути, который должны пройти вместе. В качестве примеров я буду использовать три типа наноустройств: резонансные туннельные диоды и транзисторы (RTD и RTT) [1–3], одноэлектронные транзисторы (SET) [4, 5] и устройства на квантовых точках (QDD) [6–10], не касаясь молекулярных устройств, а также устройств, базирующихся на технологии ядерной сборки, и ряда других, имеющих свою специфику.

Типичным отличием резонансных туннельных диодов и транзисторов от обычных микронных и субмикронных устройств является наличие падающего участка на вольт-амперной характеристике (рис. 1, *а*) [11], что с очевидностью изменяет характер их функционирования.

Так, простейшая пара последовательно соединенных RTD образует трехстабильную ячейку памяти (рис. 1, *б*). Соединение нескольких RTD позволяет получить композиционную вольт-амперную характеристику с несколькими падающими участками, что создает предпосылки к использованию таких устройств в схемах многозначной логики. Я не буду касаться здесь дискуссионных вопросов целесообразности использования многозначных систем, проблем их помехоустойчивости и поддержания требуемой точности работы элементов. Удовлетворимся словами поэта: «Если звезды зажигают, то это кому-нибудь нужно». При прочих равных условиях, по крайней мере, использование многозначных элементов памяти и многозначных соединительных линий выглядит достаточно привлекательным с точки зрения уменьшения их числа для запоминания и передачи того же объема информации. Вопрос о предпочтительности схем многозначной логики по сравнению с двоичными схемами не столь очевиден, и ответ на него в большой степени зависит от используемого функционального базиса.



■ *Рис. 1. Вольт-амперная характеристика для AlGaAs/GaAs RTD [11] (а); вольт-амперная характеристика и точки равновесия для пары RTD (б)*

Традиционно, так или иначе, логические построения в многозначной логике базируются на алгебраической системе Поста [12, (1921)]. Исходная система Поста модифицируется в многозначный аналог нормальных дизъюнктивных форм (MVNDF). Базовыми операциями в MVNDF являются: $\max(x, y)$; $\min(x, y)$ и x^σ $(0 < x, y, \sigma < k-1)$, где σ – константа;

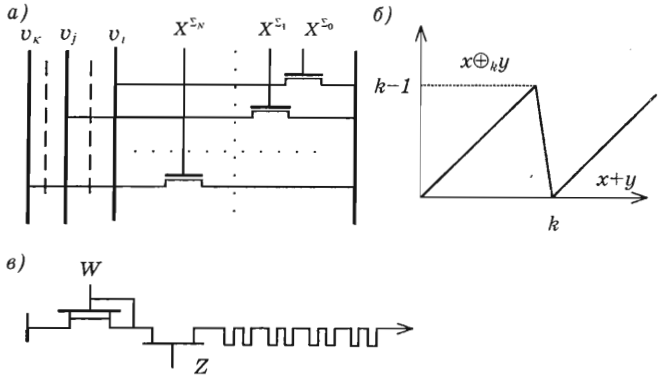
$$x^\sigma = \begin{cases} 0, & \text{если } x \neq \sigma, \\ k-1, & \text{если } x = \sigma. \end{cases}$$

Введем обозначение $X^\Sigma = \min(x_0^{\sigma_0}, x_1^{\sigma_1}, \ldots, x_{n-1}^{\sigma_{n-1}})$, где X и Σ – векторы $(x_0, x_1, \ldots, x_{n-1})$ и $(\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ соответственно. Тогда произвольная функция k-значной логики может быть представлена в виде

$$F(X) = \max_\Sigma \left(\min(F(\Sigma), X^\Sigma)\right). \qquad (1)$$

Доказательство этого утверждения тривиально. Действительно, функция X^Σ принимает максимально возможное в k-значной логике значение $(k-1)$ на одном и только на одном наборе значений переменных, а именно, на наборе $X = \Sigma$. При всех других значениях переменных $X^\Sigma = 0$. Функция $\min(F(\Sigma), X^\Sigma) = F(\Sigma)$ при $X = \Sigma$ и равна нулю во всех остальных случаях. Взятие максимума по всем Σ, за исключением, быть может, Σ, для которых $F(\Sigma) = 0$, от термов $\min(F(\Sigma), X^\Sigma)$ и порождает функцию $F(X)$.

Посмотрим на MVNDF с точки зрения схемной реализации. Функция (1) является многозначным аналогом нормальной дизъюнктивной формы в булевой алгебре. Операция x^σ – операция сравнения, многозначная по входу и бинарная по выходу. В бинарном случае ее аналогом является $x^\sigma = x \oplus \sigma$, т. е. $x^1 = x$ и $x^0 = \bar{x}$. Функция X^Σ оперирует фактически с двоичными переменными и яв-



■ *Рис. 2. Выходная ступень многозначной ПЛМ (а); сумма по модулю k в k-значной логике (б) и логический элемент с немонотонной характеристикой (в) [10]*

ляется не чем иным, как двоичной функцией AND от переменных x^σ. Функции X^{Σ_i} и X^{Σ_j}, $i \neq j$, взаимно ортогональны и для получения значений функции $F(X)$ достаточно реализовать «проводное или» между транзисторами, коммутирующими выходной провод с опорными напряжениями, представляющими значения функции (рис. 2, *а*). Нетрудно видеть, что прямая реализация MVNDF приводит нас к структуре бинарной программируемой логической матрицы (ПЛМ) с очень незначительными изменениями: на первую ступень ПЛМ подаются не значения x и $\bar{x}$, а значения x^σ (для N наборов значений переменных таких входных сигналов $k\log_k N$), во второй ступени коммутируются не 2, а k значений напряжения. Сложность реализации функции, заданной на N наборах значений переменных, $R_k(N)$ определяется сложностью формирования сигналов x^σ и рангом конъюнкции $X^\Sigma(\log_k N)$, т. е. $R_k(N) = N\log_k N + r(k)\, k\log_k N$, где $r(k)$ – сложность реализации операции x^σ. В бинарном случае $r(2)$ равно половине сложности реализации инвертора (x^0 – инвертор, x^1 – провод). Эффективность использования многозначной системы по сравнению с бинарной при таком подходе $(R_k(N) / R_2(N) = (N + r(k)k)/((N+1)\times \times\log_2 k)$ зависит от многих факторов. Возникает естественный вопрос: можно ли увеличить эффективность реализации за счет использования специальных свойств туннельно-резонансных устройств и операций, в которых в полной мере используется многозначность?

Весьма популярными в многозначных логических построениях являются операции суммирования по модулю k. Это многозначные аналоги полиномов Жегалкина [13] (в западной литературе – канонические представления Рида–Мюллера), функция Вебба и оригинальная система Поста, использующие операцию $(1+x)_{\mathrm{mod}\, k}$ и т. д. Заметим, что функция суммирования по модулю имеет падающий участок (рис. 2, *б*) и его наличие на вольт-амперной характеристике туннельного резонансного диода (или транзистора) позволяет построить схемы с падающим участком на логической характеристике. Пример такого устройства приведен в работе [11] (рис. 2, *в*). Однако логические функциональные возможности таких элементов и методы логического синтеза с их использованием требуют специальных исследований.

Коль скоро во многих функциональных построениях используется суммирование, по крайней мере, как составляющая операции суммирования по модулю, то естественно рассмотреть возможность использования операции арифметического суммирования как компоненты логических построений. Это, в свете вышесказанного, тем более интересно, что операция арифметического суммирования является реально многозначной. Нетрудно доказать, что операция арифметического суммирования-вычитания с насыщением $S(\Sigma x_i - \Sigma y_k)$, или $S(x_1+x_2-y)$ в простейшем случае, где

$$S(Z) = \begin{cases} 0, & \text{если } Z < 0, \\ Z, & \text{если } 0 \le Z \le k-1, \\ k-1, & \text{если } k-1 < Z \end{cases}$$

(рис. 3, *а*) и константы образуют в логике произвольной значности функционально полную систему[1].

В качестве примера рассмотрим реализацию сумматора в этом фукциональном базисе. Пусть $x_i + y_i + c_{i-1} = Z_i + kC_i$ (Z_i – сумма, C_i – перенос). Тогда (рис. 3, *б–д*)

$$Y_{1i} = S(x_i + y_i + c_{i-1});\ \ C_i = S(Y_{1i} - k + 1);$$

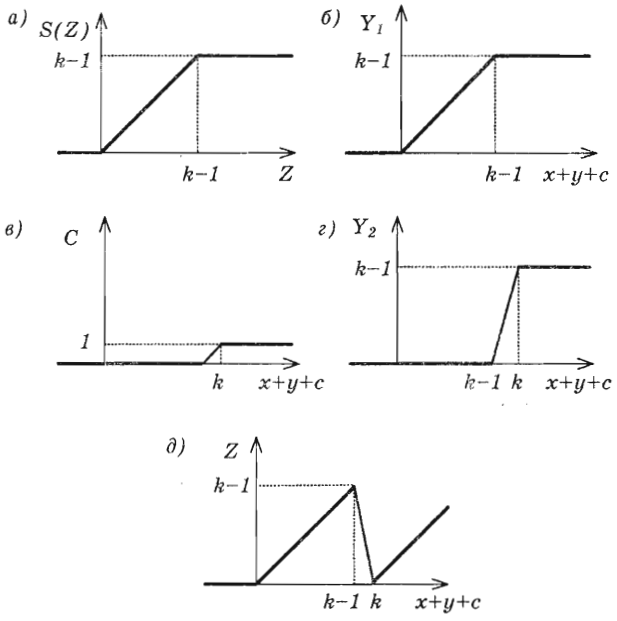
$$Y_{2i} = (k-1)C_i;\ \ Z_i = S(x_i + y_i + c_{i-1} + Y_{2i}).$$

Наличие падающего участка на вольт-амперной характеристике может быть использовано и в бинарном случае. Например, на одном многоэмиттерном транзисторе RTT возможна реализация немонотонных булевых функций, таких как $\varphi_2(x,y) = \overline{xy} \vee xy$ для двухэмиттерного RTT (рис. 4, *а*) и $\varphi_3(x,y,z) = \overline{xyz} \vee xyz$ для трехэмиттерного RTT (рис. 4, *б*) [3].

[1] Доказательство этого утверждения сводится к представлению операций любой функционально полной системы суперпозицией функций $S(x_1+x_2-y)$.

Функция $S(X)$ фактически является обобщением бинарных мажоритарной и пороговой функций. Мажоритарная функция имеет два эквивалентных представления: как булева функция $\mathrm{maj}(x, e, z) = xy \vee xz \vee yz$ и как пороговая функция $\mathrm{maj}(x, e, z) = = \mathrm{sign}(x + y + z - 2)$. Прямое расширение булевского представления на многозначный случай дает функцию медианы $\mathrm{med}(x, e, z) = \max(\min(x, y), \min(x, z), \min(y, z))$, позволяющую строить системы трехканального мажоритарного резервирования в системах любой значности, в том числе и аналоговых [24].

Для случая симметричного представления переменных ($-a < x < +a$) функционально полной является система, $S(x+y)$, инверсия ($\bar{x} = -x$) и константы.



■ *Рис. 3. Поведение функции S(Z) (а) и поведение полного k-значного сумматора (б–д)*

Вопрос о том, как включить подобные операции в состав базисных для регулярного метода синтеза, требует специального исследования. Однако для метода последовательного исключения переменных некоторые предварительные соображения могут быть высказаны уже здесь.

Пусть функция $F(x, Y)$ принадлежит к подклассу функций, для которых остаточные функции в разложении Шеннона по переменной x ортогональны, т. е.

$$F(x,Y) = \bar{x}F(0,Y) \vee xF(1,Y) \text{ и } F(0,Y)F(1,Y) \equiv 0.$$

Будем искать представление функции $F(x, Y)$ в виде

$$F(x,Y) = \varphi_3(x,\alpha(Y),\beta(Y)) = \bar{x}\bar{\alpha}(Y)\bar{\beta}(Y) \vee x\alpha(Y)\beta(Y).$$

Для этого достаточно решить систему уравнений:

$$\begin{cases} \bar{\alpha}(Y)\bar{\beta}(Y) = F(0,Y); \\ \alpha(Y)\beta(Y) = F(1,Y). \end{cases}$$

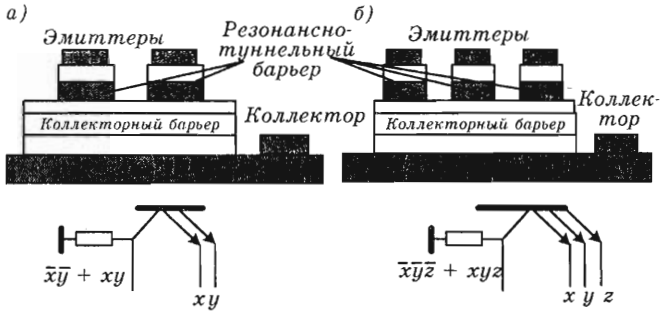
Одним из решений этой системы с учетом ортогональности остаточных функций[1] является:

$$\alpha(y) = F(1,Y),\ \beta(Y) = \bar{F}(0,Y) \text{ и}$$

$$F(x,Y) = \varphi_3(x,\alpha(Y),\beta(Y)).$$

Для произвольной булевой функции можно воспользоваться разложением Рида–Мюллера

[1] Из ортогональности $F(0,Y)$ и $F(1,Y)$ следует $F(0,Y)\ \bar{F}(1,Y) = F(0,Y)$ и $\bar{F}(0,Y)\ F(1,Y) = F(1,Y)$.



■ *Рис. 4. Двухэмиттерный (а) и трехэмиттерный RTT (б)*

$$F(x,Y) = F(0,Y) \oplus \frac{\partial F(x,Y)}{\partial x},$$

где

$$\frac{\partial F(x,Y)}{\partial x} = F(0,Y) \oplus F(1,Y).$$

Тогда

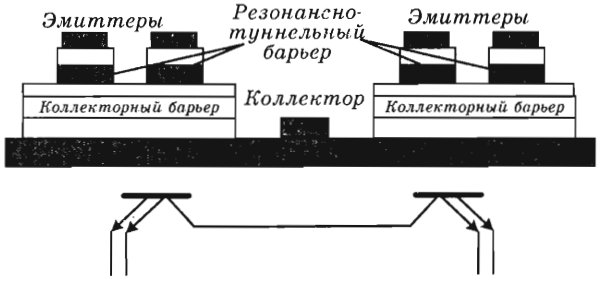
$$F(x,Y) = \varphi_2(\bar{F}(0,Y),\varphi_3(x,\frac{\partial F(x,Y)}{\partial x},1))$$

и для исключения одной переменной из произвольной булевой функции требуется один двухэмиттерный и один трехэмиттерный RTT. Дальнейшее упрощение реализации может быть осуществлено за счет представления

$$F(x,Y) = \varphi_2(\bar{F}(0,Y),\varphi_3(x,\alpha(Y),\beta(Y)),$$

где $\alpha(Y)\beta(Y) = \dfrac{\partial F(x,Y)}{\partial x}$ и $\bar{\alpha}(Y)\bar{\beta}(Y) = 0$.

Я абсолютно уверен, что объединение усилий специалистов по логическому проектированию, физиков и технологов позволит целенаправленно создавать новые резонансно-туннельные структуры, обладающие необычными и высокоэффективными логическими возможностями. Не являясь специалистом ни в физике, ни в технологии, я осмелюсь обратить внимание на гипотетический многоэмиттерный элемент с общим коллектором (рис. 5) с реализуемой логической функцией



■ *Рис. 5. Два двухэмиттерных транзистора с общим коллектором*

$$f(x_1,x_2,x_3,x_4)=(\bar{x}_1\bar{x}_2\vee x_1x_2)(\bar{x}_3\bar{x}_4\vee x_3x_4).$$

Перспективным элементом как для бинарного, так и для многозначного случая мог бы, например, оказаться многоэмиттерный (или многоколлекторный) коммутатор, перераспределяющий токи между эмиттерами (коллекторами) в зависимости от напряжения на базе или распределения напряжения вдоль базы.

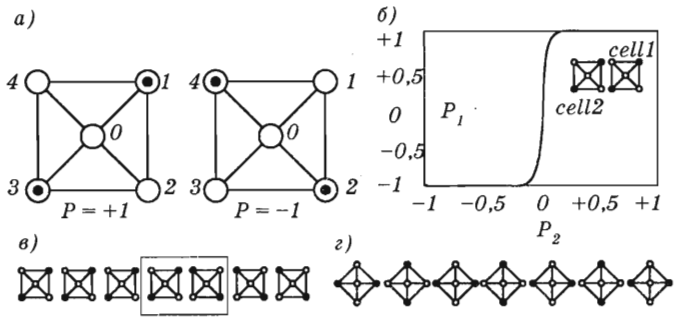
Наиболее интригующими квантовыми устройствами являются, безусловно, устройства с квантовыми точками (QDD) [6–10], обладающие, с точки зрения внешнего наблюдателя, рядом мистических свойств.

QDD, предложенные К. С. Лентом и его соавторами, представляют собой мезоскопические квантовые структуры. Элемент с квантовыми точками (QDC) – это структура из пяти квантовых точек, которая содержит два электрона и может находиться в двух устойчивых состояниях (рис. 6, *а*). Взаимодействие между ячейками осуществляется за счет кулоновых сил. Поляризация P элемента определяется как некоторая величина, измеряющая направления, в которых плотность заряда располагается вдоль одной из двух диагональных осей, как это показано на рис. 6;

$$P=\frac{(\rho_1+\rho_3)-(\rho_2+\rho_4)}{\rho_1+\rho_3+\rho_2+\rho_4},$$

где ρ_i – плотность вероятности нахождения электрона в точке i. В устойчивых состояниях $P=\pm1$.

Внешняя поляризация переключает состояние QDC (рис. 6, *б*). Рассмотрим простейшую линейную композицию ячеек QDC (линия, передающая информацию). Такая структура [8] может быть двух типов (рис. 6, *в, г*). Если к крайней QDC приложено внешнее поле, переключающее ее состояние, то элементы линии начинают переключаться и линия переходит в новое состояние с минимальной энергией. По мере удаления границы смены состояний от источника внешнего поля влияние последнего на процесс переключения уменьшается. На границе смены состояний, при идеальных условиях (все геометрические размеры выдержаны абсолютно точно), в силу симметрии кулонового взаимодействия, две соседние QDC имеют нулевую внешнюю поляризацию от совокупности правых и левых соседей (выделенный контур на рис. 6, *в*), и сдвиг границы определяется только воздействием от внешнего источника $U(r)$, где r – расстояние от источника внешнего сигнала. При этом время переключения пропорционально $\frac{1}{\varepsilon}>\frac{\hbar}{\Delta U(r)}$ [14], где $\Delta U(r)$ – энергия, разделяющая два состояния (ε – вероятность смены состояния за единицу времени). При локальном сдвиге границы $\Delta U(r)\overset{r\to\infty}{\to}0$ и задержка распространения сигнала по линии с ростом n становится непрогнозируемо большой[1].



■ ***Рис. 6.*** *Две QDC в двух состояниях (а), зависимость поляризации элемента cell1 от внешней (cell2) поляризации (б), возможные линейные конфигурации из QDC (в, г)*

Возможно, такой вывод неправилен в силу некорректности применения использованного подхода к квантовому объекту. Если рассматривать всю линию как единое целое, то время ее переключения имеет порядок $\hbar/\Delta E$, где ΔE – энергия переключения крайней QDC линии. Однако если последнее верно, то мы приходим к совершенно поразительному выводу для логических схем на QDC (о них чуть ниже).

Пусть мы реализуем логическую функцию от n переменных. Сложность реализации растет[2], как $2^n/n$, а инжектируемая энергия растет линейно от числа переключаемых входных переменных. Тогда при росте сложности схемы как $2^n/n$ ее быстродействие *растет* (!)[3], как n. И уж совсем неясно, за счет каких сил может быть преодолен локальный энергетический барьер при наличии разветвлений линии (энергетический барьер должен возникать также при предложенном в работе [8] пересечении проводов в силу большего расстояния до предшествующей QDC, чем до последующей). Так или иначе, справедливы ли приведенные выше соображения или нет, нужно подчеркнуть: 1) вопрос о динамике поведения QDD требует специального исследования; 2) для улучшения динамических характеристик поведения и преодоления локальных энергетических барьеров в точках разветвления линий передачи сигналов необходима подкачка энергии (использование усилителей). Это следует из общих представлений об энергетике информационных процессов [13] и понятия произведения мощности на скорость. Такие усилители, по-видимому, могут

[1] Что, по-видимому, и позволило Ленту с соавторами сделать следующее утверждение: «вычисления на границах не требуют ни энергии ни информации для передачи непосредственно внутрь элементов» [6].

[2] Асимптотически по n.

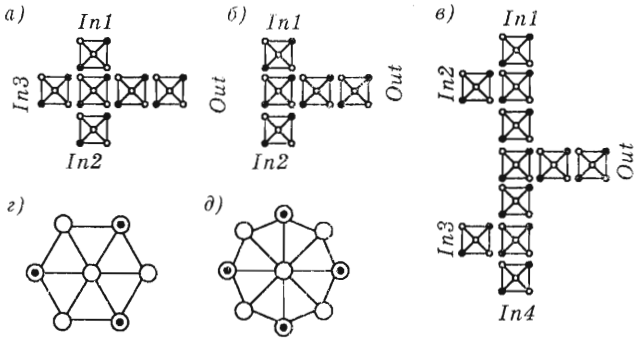
[3] Воистину «торжество науки над здравым смыслом».

				x_1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1
				x_2	−1	−1	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1	1	1
				x_3	−1	−1	−1	−1	1	1	1	1	−1	−1	−1	−1	1	1	1	1
p_1	p_2	p_3	p_Y	Y	−1	−1	−1	−1	−1	−1	−1	−1	1	1	1	1	1	1	1	1
1	1	1	1	F_1	−4	−2	−2	0	−2	0	0	2	−2	0	0	2	0	2	2	4
1	1	0,5	1	F_2	−3,5	−1,5	−1,5	0,5	−2,5	−0,5	−0,5	1,5	1,5	0,5	0,5	2,5	−0,5	1,5	1,5	3,5
1	1	0	1	F_3	−3	−1	−1	1	−1	−1	−1	1	−1	1	1	1	−1	1	1	3
1	1	−0,5	1	F_4	−3,5	−1,5	−1,5	0,5	−3,5	−0,5	−0,5	1,5	−0,5	1,5	1,5	3,5	−0,5	1,5	1,5	3,5
1	1	0,5	1	F_5	−2,5	−0,5	−0,5	1,5	−2,5	−0,5	−0,5	1,5	−0,5	1,5	1,5	3,5	−0,5	1,5	1,5	3,5
1	1	1	0,5	F_6	−3,5	−1,5	−1,5	0,5	−1,5	0,5	0,5	2,5	−3,5	−1,5	−1,5	0,5	−1,5	0,5	0,5	2,5

быть созданы на базе одноэлектронных транзисторов [4, 5], управляемых квантовыми точками [15].

Рассмотрим совокупность логических ячеек QDC (рис. 7). Общая идея построения логических QDD состоит в суммировании воздействий на QDC ее соседей и в ее переключении в зависимости от суммарного внешнего воздействия. Базовым логическим элементом является трехвходовый мажоритарный элемент (рис. 7, *а*). Заметим, что, как уже отмечалось выше, в силу симметрии кулоновских взаимодействий влияние соседней QDC, являющейся выходной, ничем не отличается от влияния входных QDC. Поэтому в действительности логический элемент на рис. 7, *а* реализует логическую функцию не трех, а четырех переменных. Влияние выходной QDC является обратной связью по состоянию, и логический элемент (при соответствующем проектировании) обладает памятью.

В таблице x_i – входы логической QDC, Y – ее выход, p_i – коэффициент влияния входной QDC (уменьшаемый, например, удалением от логической QDC), p_3 со знаком (±) – постоянное внешнее поле. В таблице приведены значения суммарной внешней поляризации в зависимости от поляризации входных и выходной QDC для различных коэффициентов влияния p_3.



■ ***Рис. 7.*** *Мажоритарный элемент из [8]* (а), *двухвходовый С-элемент Маллера (б), четырехвходовый С-элемент Маллера (в), возможные (?) многоэлектронные QDC (г, д)*

Функция F_1 соответствует «мажоритарному элементу» на рис. 6, *а*. Логическая функция задана на наборе своих значений и значений входных переменных {−1,+1}. Формально, при значениях внешней поляризации, равных нулю[1], функция не определена[2]. Предполагается, что при этих значениях состояние логической QDC будет переключаться в направлении общего состояния схемы с минимальной энергией. В этом и только в этом случае устройство реализует мажоритарную функцию. Однако опять, как и в случае линии из QDCов, возникает вопрос о времени достижения финального состояния. Удаление одной из входных QDC (уменьшение влияния) делает логическую QDC всюду определенной, но изменяет реализуемую логическую функцию. Из таблицы следует, что F_2–F_5 представляют одну и ту же функцию и достаточно рассмотреть F_3 (рис. 7, *б*). Вопреки утверждению [7], F_3 представляет собой не элемент AND(OR), а двухвходовый С-элемент Маллера с собственной функцией $F_3 = x_1x_2 \vee F_3(x_1 \vee x_2)$. Многовходовой С-элемент Маллера строится очевидным образом (рис. 7, *в*) и имеет функцию $Y = \&_i x_i \vee Y(\vee_i x_i)$. С-элемент обладает памятью, весьма важен во многих практических применениях (синхронизаторы, конвейеры и т. д.), но он вместе с инвертером не образуют функционально полной системы элементов.

Для исключения неопределенных состояний (состояний с нулевой внешней поляризацией) возможны, по крайней мере, два пути:

[1] Обеспечение равенства нулю суммарной внешней поляризации, безусловно, требует очень высокой точности поддержания геометрических размеров; ошибки в геометрических размерах могут сдвигать суммарное внешнее воздействие в любую сторону, создавая, как, впрочем, и в случае линии, локальный энергетический барьер.

[2] Это связано, в частности, с тем, что для четного числа переменных мажоритарная функция не определяется.

1) исключение влияния выходной QDC для нечетного числа входных QDC, например, включением усилителя; в этом случае трехвходовый логический элемент действительно реализует мажоритарную функцию без неопределенных состояний (F_6 в таблице);

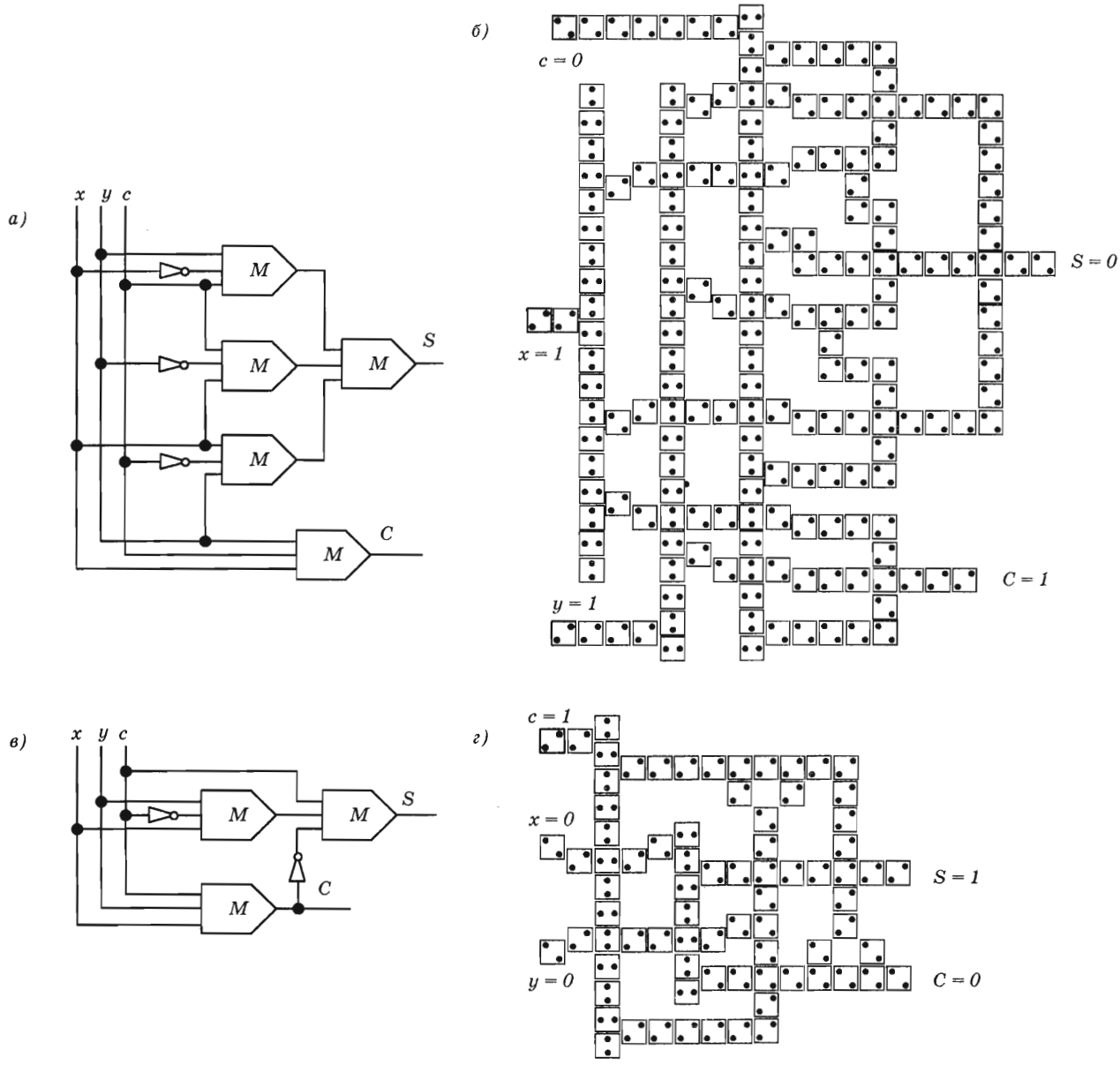
2) использование элементов с четным числом входов; в этом случае, с учетом влияния выходной QDC, внешнее воздействие всегда имеет определенное ненулевое значение. Что, в частности, принципиально ослабляет требования к точности геометрических размеров.

Здесь возникает вопрос к физикам: «Возможно ли создание многоточечных элементов, имеющих $2n$ (или $2n+1$) квантовых точек и n электронов, аналогичных 4(5)-точечных устройств (например, типа рис. 7, *в, г*)?»

Если такие устройства могут быть созданы и могут эффективно управляться за счет суммирования поляризаций их соседей, то справедливо следующее:

1) $(2k-1)$-входовый $2n$-точечный элемент ($2n$QDC), $k < n$, при исключенном влиянии выходной QDC реализует мажоритарную функцию $(2k-1)$ переменной – $Y = S_{2k-1}^{k-1}$;

2) $2(k-1)$-входовой $2n$QDC с учетом влияния выходной QDC является H-триггером [16], т. е. схемой с памятью и логическим уравнением – $Y = S_{2k}^{k} \vee Y S_{2k}^{k-1}$, где



■ ***Рис. 8.*** *Логическая структура и QDD полного сумматора [8] (а, б), логическая структура и QDD нового полного сумматора (в, г)*

$$S_n^k = \begin{cases} 0, & \text{если } \sum x_i \le k, \\ 1, & \text{если } \sum x_i > k, \end{cases}$$

– монотонная симметрическая функция n переменных.

При этом весьма привлекательным выглядит четырехвходовый 6QDC с уравнением

$$Y = x_1x_2x_3 \vee x_1x_2x_4 \vee x_1x_3x_4 \vee x_2x_3x_4 \vee$$
$$\vee Y(x_1x_2 \vee x_1x_3 \vee x_1x_4 \vee x_2x_3 \vee x_2x_4 \vee x_3x_4),$$

обладающий памятью и богатыми функциональными возможностями. В частности, при разветвлении выхода (наличии двух выходных QDC) – это трехвходовый С-элемент без энергетического барьера на выходном разветвлении, а при $x_4 = 1$ – мажоритарный элемент с индикацией спейсера.

В отсутствие памяти QDD носят в большинстве публикаций название «клеточные автоматы» лишь по недоразумению. Фактически это обычные логические схемы. Так, в качестве примера QDD в большом числе публикаций приводится полный сумматор (рис. 8, *а*, *б*). Если гипотезы о работоспособности основных принципов, положенных в основу его реализации, справедливы, то взгляд на него как на обычную логическую схему позволяет стандартными методами логического проектирования получить гораздо более простое решение (рис. 8, *в*, *г*).

«Природа коварна, но не злонамеренна». Свойства объектов, с которыми мы работаем, не бывают плохими или хорошими, полезными или вредными. Есть свойства, которые мы умеем использовать, и свойства, которые мы использовать не умеем. В этом смысле симметрия кулоновского взаимодействия является свойством, эффективное использование которого ждет своего исследования. Мы уже отметили, что симметрия взаимодействия позволяет вводить память в логические элементы. Кроме того, деление внешних полюсов QDD на входы и выходы условно и определяется только наличием или отсутствием внешнего воздействия (при отсутствии встроенных усилителей, делающих взаимодействие однонаправленным). Именно здесь кроются, с моей точки зрения, совершенно уникальные возможности по созданию действительно клеточных автоматов. Однако пока это не более, чем мысли вслух и добрые пожелания. С другой стороны, мне хотелось бы еще раз акцентировать внимание на проблемах динамики и информационно-энергетических соотношений. Мне кажется, что без введения усилителей, обеспечивающих подкачку энергии, однонаправленность передачи информации (особенно в схемах с не кулоновскими, а с информационными обратными связями) и повышающих определенность смены состояний, перспектива создания компьютера на QDD кажется достаточно сомнительной. Более того, композиция QDD и SET (одноэлектронных транзисторов), если она осуществима технологически, обещает поразительные функциональные возможности.

Одной из ключевых проблем создания VLSI новых поколений является проблема синхронизации. Традиционно эта проблема решалась и решается путем использования внешних часов, сигналы от которых инициируют очередной шаг работы устройства и маскируют переходный процесс. При этом интервал между сигналами времени должен перекрывать возможные вариации длительностей переходных процессов. По мере уменьшения размеров конструктивных элементов и роста сложности VLSI такой подход начал сталкиваться с непрерывно возрастающими трудностями. Эти трудности связаны как с построением собственно системы доставки сигналов времени ко всем точкам синхронизации (точность доставки, связанная с собственными задержками проводов в системе доставки; мощность, необходимая для транспортировки сигналов времени с нужной скоростью, и т. д.), так и с увеличивающейся ролью задержек в соединительных проводах и разбросом этих задержек. Переход к наноструктурам в значительной мере усугубляет эти проблемы. Во-первых, просто за счет увеличения сложности VLSI и увеличения быстродействия активных элементов, требующих для использования этого быстродействия более высокой точности системы синхронизации. Во-вторых, переходные процессы в устройствах, имеющих квантовую природу, по сути своей имеют случайные длительности, что делает предельно неэффективным использование для их синхронизации длительности такта, ориентированного на худший случай. Выходом из этого положения является использование методологии самосинхронизации [16, 17]. Общая идея самосинхронизации (или логической синхронизации [18]) связана с отказом от использования внешних часов и организации поведения устройств во времени за счет организации логического взаимодействия между событиями в устройствах, отражающего причинно-следственные связи между этими событиями.

При недостаточной длительности цикла синхронизации переходный процесс может не уложиться в это время. В силу неблагоприятного сочетания задержек некоторые выходы либо «не успеют» изменить свое состояние, либо изменят его на неправильное (состязания или гонки). Для предотвращения ошибок синхронизации необходимо либо выбирать период синхронизации настолько большим, чтобы все переходные процессы заведомо завершились, либо уметь определять моменты их завершения. Наличие сигнала окончания переходного процесса позволяет инициировать следующий такт работы, и длительность каждого цикла при этом адаптируется к реальным задержкам в устройстве.

Процедура определения моментов окончания переходных процессов (процессов перехода схемы

(устройства) из одного состояния в другое) связана со специальным (самосинхронным) кодированием состояний. Основа самосинхронного кодирования восходит к идеям эквидистантного кодирования, при котором расстояния от данного кода до всех кодов, достижимых из него за один переход, равны. Самосинхронные коды (ССК) для двухзначного случая изучены достаточно подробно [16, 19]. Однако, коль скоро мы коснулись выше возможности построения многозначных систем на резонансно-туннельных диодах и транзисторах, имеет смысл рассмотреть возможность построения ССК для многозначных переменных[1]. Для простоты рассмотрим применение многозначных ССК для параллельной передачи n-разрядного k-значного кода.

Мы будем называть весом кода $X = \{x_0, x_1, ..., x_{n-1}\}$ величину $W(X) = \sum_0^{n-1} x_j$, а расстоянием по Хеммингу между двумя кодами $X = \{x_0, x_1, ..., x_{n-1}\}$ и $Y = \{y_0, y_1, ..., y_{n-1}\}$ величину $H(X,Y) = \sum_0^{n-1} (x_j \oplus_k y_j)$, где операция $\oplus_k$ обозначает сумму по модулю k.

Простейшим примером ССК является парафазный код (ПК). В этом коде каждому разряду соответствуют два провода ($y_j = x_j$ и $\hat{y}_j = k-1-x_j$) и число передается (кодируется) $2n$-разрядным кодом, в котором все кодовые наборы равновесны ($W(Y) = n(k-1)$). В кодовую систему вводится дополнительный минимальный кодовый набор S_0 – спейсер, с весом $W(S_0) = 0$, где $S_0 = \{s_j = 0, \hat{s}_j = 0\}$; в равной мере в качестве спейсера может быть использован и максимальный кодовый набор $S_1 = \{s_j = k-1, \hat{s}_j = k-1\}$, $W(S_1) = 2n(k-1)$. Протокол самосинхронного обмена содержит четыре фазы (с учетом запрос-ответного взаимодействия):

$$S_0 \xrightarrow{W(S_0)=0} Ответ$$
$$\uparrow \qquad\qquad \downarrow$$
$$Ответ \xleftarrow{W(Y)=n(k-1)} Y\ .$$

Преимуществами ПК является простота кодирования и декодирования (кодирование сводится к вычислению значения идентификатора, декодирование совпадает с индикацией момента окончания переходного процесса) и возможность независимой индикации моментов окончания переходных процессов в каждой паре проводов[2]. Однако избыточность кода при этом высока и равна 1.

Снижение избыточности может быть достигнуто за счет введения более изощренного кодирования. Таким кодом, например, является оптимальный равновесный код (ОРК или код Спернера), включающий в качестве кодовых наборов все коды с $W(Y) =]kn/2[$. Асимптотическая по n и k ($n \to \infty, k \to \infty$) избыточность ОРК[3] равна $\frac{1}{2n}\log_k n$. Из-за чрезвычайно сложных процедур кодирования и декодирования ОРК может быть использован для кодирования состояний, но не может быть рекомендован для информационного обмена.

Невысокой избыточностью при достаточно простых процедурах кодирования-декодирования обладает код с идентификатором (КИ). КИ $Y = \{X, Z\}$ состоит из двух частей: собственно n-разрядного k-значного кода $X = \{x_0, x_1, ..., x_{n-1}\}$ и идентификатора $Z = \{z_0, z_1, ..., z_{m-1}\}$ – m-разрядного k-значного кода числа I такого, что

$$I = \sum_0^{m-1} k^j z_j = n(k-1) - \sum_0^{n-1} x_j.$$

Нетрудно видеть, что если вес кода вычисляется как

$$W(Y) = \sum_0^{n-1} x_j + \sum_0^{m-1} k^j z_j,$$

то КИ является равновесным кодом ($W(Y) = n(k-1)$), самосинхронным в протоколе со спейсером. Избыточность КИ имеет порядок $\frac{1}{2n}\log_k n$, более точно

$$m =]\log_k n(k-1)[,\ r = \frac{n+m}{n} - 1 = \frac{1}{n}]\log_k n + \log_k (k-1)[.$$

Существование ССК создает принципиальную основу для самосинхронизации как двоичных, так и многозначных схем и устройств[4]. Однако этого недостаточно для решения задачи самосинхронизации.

Для того чтобы протокол обеспечивал корректную индикацию завершения переходного процесса по факту достижения финального состояния перехода, необходимо также обеспечить монотонность изменения веса ССК. Для бинарного случая процедуры, обеспечивающие такую монотонность, известны и изучены. Например, использование функциональных элементов с монотонной собственной логической функцией (например, элементов NOR

[1] Насколько мне известно, вопрос о самосинхронизации в многозначных системах не изучался.

[2] Сигнал $\hat{y}$ является многозначной инверсией сигнала x и может быть эффективно использован в логических построениях.

[3] Для двоичных кодов доказано, что ОРК обладает наименьшей избыточностью среди всех ССК. Для многозначных кодов такое предположение кажется правдоподобным, но не доказано.

[4] При этом для индикации момента достижения финального состояния перехода может быть использован как четырехфазный протокол со спейсером, так и двухфазный протокол с искусственным спейсером в индикаторе устройства, в котором ССК используется для кодирования изменений в коде состояния («кодирование в изменениях» [17]).

и NAND) совместно с парафазным представлением переменных обеспечивает монотонное изменение веса индицируемого ССК в протоколе со спейсером. Однако введение в функциональный базис элементов с немонотонной функцией (например, многоэмиттерного RTT), в силу возможности возникновения функциональных состязаний, являющихся свойством не реализации, а собственной функции элемента, требует использования специальных дисциплин изменения сигналов на входах и внутри устройства. Эти проблемы усугубляются в многозначном случае. Возможно, путь к преодолению этих трудностей лежит в направлении создания протоколов с частично немонотонными переходами, при которых допустимы колебания веса индицируемого ССК (реализация, не свободная от состязаний, или реализация с многократным переключением переменной в течение переходного процесса), но пороговый вес ССК достигается только в терминальном состоянии. Примеры таких переходов могут быть построены в классе неполумодулярных схем, не зависящих от скорости, по классификации Маллера [25]. Однако общие вопросы синтеза таких устройств не изучены.

Достижение терминального состояния, вообще говоря, не означает завершения переходных процессов. Терминальным по данному переходу может быть и подмножество состояний с одним и тем же характеристическим кодом. Серьезные проблемы возникают, когда провод или отрезок провода начинает выступать как компонент устройства. Методология самосинхронизации использует гипотезу Маллера относительно задержек в проводах – *вся задержка провода приведена к выходу элемента, а разбросом задержек в проводах после разветвления можно пренебречь.* В этом случае провода вообще исключаются из рассмотрения. Нарушение гипотезы Маллера приводит к нарушению причинной обусловленности поведения, являющейся логической основой самосинхронизации. Причинная обусловленность требует, чтобы каждое событие в системе являлось причиной, по крайней мере, одного другого события (свойство индицируемости самосинхронных систем [16, 17]). В логических структурах, в отличие от систем передачи, изменение состояния отрезка провода после разветвления может не приводить к переключению логического элемента и, следовательно, не индицироваться. При этом отрезок провода начинает выступать как элемент памяти. Для борьбы с этим, т. е. для построения схем, не зависящих от задержек в проводах, необходимо использование либо специальных дисциплин переключения (что сужает класс реализуемых схем), либо использование специальных логических или топологических конструкции, как, например, изохронные разветвления [20, с. 270–272] или разветвления полем [17, с. 261], требующих введения новых гипотез или/и приемов проектирования, зависящих от технологии. Эта проблема усугубляется с ростом влияния задержек в проводах и дисперсии этих задержек (что и имеет место в квантовых структурах). Быть может, проблема синхронизации окажется ключевой для эффективного использования будущей квантовой электроники в вычислительной технике.

Конечно, я коснулся здесь лишь нескольких примеров возможных квантовых систем. Спектр обсуждаемых возможных подходов принципиально шире. В равной мере рассмотренные примеры из области логического проектирования не покрывают всего многообразия проблем. Как я уже отмечал выше, это только приглашение к диалогу. Уместно отметить здесь, что квантовая механика более полувека тому назад сформулировала свои требования к логике [21][1]. Язык и модели сетей Петри, широко используемые сегодня в вычислительной науке и технике, первоначально (в 1960–1962 гг.) создавались Карлом Адамом Петри как общая модель взаимодействия физических событий, и лишь затем были применены к событиям в автоматах [22]. Предсказанный 35 лет назад в известной лекции Феймана [23] переход к организации вычислительных процессов на квантовом уровне стоит на пороге, и наша общая задача – восполнить лакуны, которые затрудняют этот переход.

Возможно, ряд утверждений и предположений, высказанных выше, выглядят наивными с точки зрения физика. Более того, быть может, кое-что, с точки зрения физика, вообще неверно. В этих случаях автор просит прощения за неквалифицированное вторжение в чужую область. Ошибки простительны неофитам. Однако, так или иначе, но анализ логических возможностей будущих квантовых устройств ставит множество вопросов[2] и задач, отдельные из которых сформулированы выше. Вопросы друг к другу и ответы на них или совместный поиск этих ответов позволят, я надеюсь, объединить усилия физиков, технологов и логических проектировщиков для целенаправленной работы в области создания эффективных квантовых вычислительных устройств будущего. Желание обратить внимание на существование областей взаимных интересов и было основным побудительным мотивом написания этой работы.

В работе имеется ряд скептических высказываний по поводу клеточных автоматов на квантовых точках. Однако именно работы Лента и других во многом стимулировали интерес автора к этой области и желание работать в ней.

[1] Интересно отметить, что отношение импликации в системе L логик фон Неймана «определяет частичное упорядочение L». С другой стороны, частично упорядоченные системы составляют основу многих наиболее популярных моделей в логическом и архитектурном проектировании компьютеров.

[2] Говорят, что не бывает глупых вопросов, бывают глупые ответы.

Автор выражает глубокую признательность коллегам по Университету Айдзу профессорам Виктору Рыжему, инициировавшему интерес автора к проблемам логического проектирования квантовых устройств, Григорию Хренову, Рафаилу Лашевскому и Вячеславу Мараховскому за помощь, время и усилия, потраченные на обсуждение этой работы, а также Вадиму Смоленскому за помощь в подготовке рукописи.

Литература

1. **Capasso F.** Quantum transistors and integrated circuits// In nanotechnology/ Ed. B. C. Crandal, J. Lewis. – The MIT Press, 1992. – P. 171–197.
2. **Chang C. Y., Kai F.** GaAs high-speed devices. – John Wiley & Sons Inc., 1994.
3. **Yokoyama N.** et al. Present status and future prospects of resonant tunneling hot electron transistors/ / 2nd International Workshop on Quantum Functional Devices, Extended Abstracts, R&D Association for Future Electron Devices, May 23–25, 1995, Matsue, Japan. – P. 40–43.
4. **Nakazato K.** Possibility of single-electron logic devices// 2nd International Workshop on Quantum Functional Devices, Extended Abstracts, R&D Association for Future Electron Devices, May 23–25, 1995, Matsue, Japan. P. 4–5.
5. **Murase K.** Silicon single-electron transistors on SIM OX Substrate// Invited Talk, VLSI'95, August 30-September 1, 1995. Chiba, Japan.
6. **Lent C. S., Tougaw P. D., Porod W., Berbstain G. H.** Quantum cellular automata// Nanotechnology 4. 1993. P. 49–57; Jpn. J. Appl. Phys., 74 (5), 1993. – P. 3558–3565.
7. **Tougaw P. D., Lent C. S., Porod W.** Bistable saturation in coupled quantum-dot cells// Jpn. J. Appl. Phys., 74 (5), 1993. – P. 3558–3565.
8. **Tougaw P. D., Lent C. S.** Logical devices implemented using quantum cellular automata// Jpn. J. Appl. Phys. 75 (3), 1994. – P. 1818–1825.
9. **Tanamoto T., Katoh R., Naruse Y.** A novel quantum cellular automata logic with loop structure, Jpn. J. Appl. Phys., 33, 1994. – P. L1502–L1505.
10. **Lent C. S., Tougaw P. D., Porod W.** Quantum cellular automata: Computing with quantum dot molecules/ / 2nd International Workshop on Quantum Functional Devices, Extended Abstracts, R&D Association for Future Electron Devices, May 23–25, 1995, Matsue, Japan. – P. 140–143.
11. **Micheel L. J., Taddiken A. H., Seabaugh A. C.** Multiple-valued logic computation using micro- and nanoelectronic devices// Proc. Int. Symp. Multiple Valued Logic. IEEE, 1993. – P. 164–169.
12. **Post E. L.** Introduction to a general theory of elementary propositions// Amer. J. Math. – Vol. 43. – 1921. – P.163–185.
13. **Zhegalkin I. I.**// Math. Collection. – Vol. 34 (1), 1927. – P. 9–28 (in Russian).
14. **Mead C., Conway L.** Introduction to VLSI Systems. – Addison-Wesley, 1980 (Chapter 9, Physics of Computational Systems, P. 333–371).
15. **Field M.** et al. Measurement of coulomb blockade with a noninvasive voltage probe// Phys. Rev. Letters/ – 70 (9). – 1993. – P. 1311–1314.
16. Self-timed control of concurrent processes/ Ed. V. Varshavsky. – Kluwer Academic Publishers, 1990 (translation from Russian issue, 1986).
17. **Kishinevsky M., Kondratiev A., Taubin A., Varshavsky V.** Concurrent Hardware. – John Wiley & Sons, 1994.
18. **Varshavsky V., Marakhovsky V., Chu T.-A.** Logical timing (Global synchronization of asynchronous arrays)// Proceedings of First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis, Aizu-Wakamatsu, Japan, IEEE Press., March 15–17, 1995.
19. **Verhoeff T.** Delay-insensitive codes – an overview// Distributed Computing. – 1988. – 3. – P. 1–8.
20. Formal methods for VLSI design/ Ed. J. Sttaunstrup, IFIP WG 10.5 Lecture Notes, 1990.
21. John von Neumann, Quantum Logics (strict- and probability-logics)[1], Collected works, Vol. 4. – New York, 1962. – P. 195–197.
22. **Petri C. A.** Kommunication mit Automaten. Schriften fur des Rheinich-Westfalischen Inst. fur Mathematik, Univ. Bonn, 1962.
23. **Feynman R.** There's plenty of room at the bottom: an invitation to enter a new field of physics// Nanotechnology/ Ed. B. C. Crandal, J. Lewis. Перепеч. из "*Engineering and Science*", February, 1960.The MIT Press, 1992. – P. 347–363.
24. **Varshavsky V., Ivanov V., Zeitin V.** USSR Invertory Certificate. № 217043, Inventory Bulletin, 15, 26.04.1968 (in Russian).
25. **Miller R. E.** Switching Theory. Vol. 2. – John Willey & Sons, 1965.
26. **Yokoyama N.** et al. Quantum Effect Devices. Potential Application of Resonant-Tunneling Hot Electron Transistors. Proc. International Conference on Advanced Microelectronic Devices and Processing, 1994.

[1] Неоконченная рукопись (1937).

УДК 681.3

САМОСИНХРОНИЗИРУЕМЫЙ КОНЕЧНЫЙ АВТОМАТ: ОТ ПРИМЕРА К СИНТЕЗУ[1]

В. И. Варшавский,
доктор техн. наук, профессор
В. Б. Мараховский
доктор техн. наук, профессор
университет Айдзу (Япония)

В статье рассматриваются проблемы синтеза самосинхронных устройств и способы их решения. Показано, что широко используемый язык конечных автоматов может успешно применяться для проектирования таких устройств. Возникающие проблемы синтеза и методы их решения иллюстрируются на примере проектирования самосинхронного буферного запоминающего устройства типа СТЕК.

In the article the problems of self-timed devices synthesis and the methods of their solution are discussed. It is shown that the widely used language of finite automata can be successfully applied for designing such devices. The appearing problems of synthesis and the methods of their solution are illustrated on the base of the example of designing self-synchronous buffer memory of the STACK type.

Введение

Одной из ключевых проблем синтеза самосинхронных схем и устройств является спецификация взаимодействия с внешней средой. В принципе эта проблема снимается использованием стандартных приемов и стандартного запрос-ответного взаимодействия со стандартными (библиотечными) входными устройствами [1, 2]. Однако в ряде случаев задача синтеза формулируется именно относительно схем взаимодействий с внешней средой. В этом случае опять-таки могут быть использованы известные приемы синтеза с использованием известных языков спецификации поведения (сигнальные графы переходов STG [3, 4], диаграммы изменений CD [5, 6] и др.). При этом, как правило, имитация поведения внешней среды включается в общую спецификацию поведения, что, в свою очередь, требует «развертки» общей спецификации и существенно усложняет как процедуру синтеза, так и собственно процедуру спецификации.

С другой стороны, хорошо разработанный и широко используемый язык конечных автоматов позволяет без специальных проблем задавать требуемое поведение во взаимодействии с внешней средой в синхронных моделях. Естественно при этом желание объединить изученность и простоту конечно-автоматных моделей с методами спецификации и синтеза самосинхронных структур. Эта идея явно не оригинальна. Мы сами начинали изучение самосинхронных структур с таких моделей [7, 8]. Широко известны и другие работы по использованию конечно-автоматных моделей в самосинхронных устройствах, например [9, 10].

Ниже мы постараемся продемонстрировать одну из возможностей использования автоматного подхода к проектированию самосинхронных устройств. Она изложена на примере, в качестве которого взято одно из широко известных устройств – самосинхронная стековая память типа LIFO (Last-In-First-Out).

Самосинхронная память типа «стек»

Общая структура. Известно несколько подходов к построению памяти типа «стек». Все их можно разделить на два основных типа:

1) регистровые структуры с реверсивным сдвигом информации;

2) стеки на основе памяти с реверсивным сдвигом адресного маркера.

При построении самосинхронных устройств стековой памяти, как правило, рассматриваются регистровые структуры [9, 10]. Такое ограничение связано со слабой изученностью самосинхронных массивов памяти. Индикация момента за-

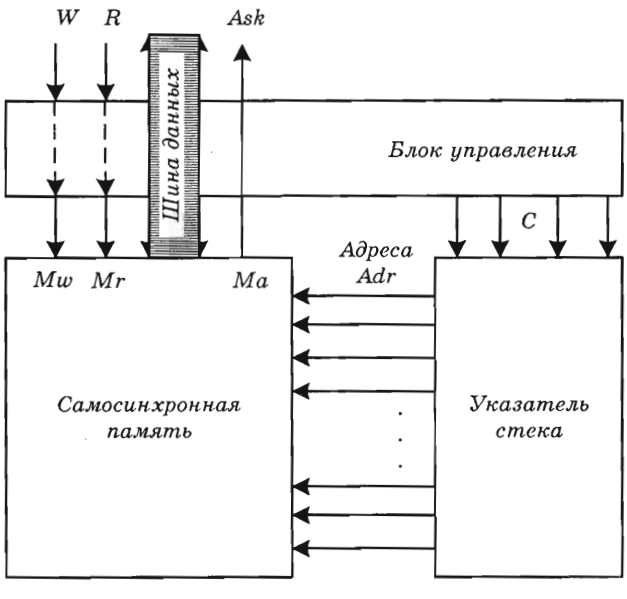
[1] STFSM – Self-Timed Finite State Machine: from Example to Synthesis, 3-d International Design Automation Workshop (Russian Workshop'93), July 19–20, Moscow, 1993.

вершения чтения, например, из обычной CMOS статической памяти при парафазных шинах чтения данных не представляет никакой проблемы. Основная задача состоит в индикации момента окончания записи. Эта задача может быть решена (и именно так решена [11, 12] для CMOS статической памяти) путем расчленения процесса записи на чтение информации и перезапись без гашения выходов. Окончание процесса записи при этом определяется совпадением считываемого и записываемого кодов. Мы здесь не касаемся тонкостей организации и схем управления памятью, поскольку эти вопросы находятся за рамками предмета этой статьи. Заметим только, что, конечно, использование такой самосинхронной процедуры приводит к замедлению работы памяти. Однако в памяти типа LIFO и FIFO (First-In-First-Out) нам известен следующий адрес или пара возможных следующих адресов, что позволяет принципиально ускорить работу такой памяти, организовав пайп-лайновое взаимодействие «входной регистр – память», обеспечивающее параллельную работу внешних регистровых портов и собственно памяти. Однако этот вопрос также выходит за рамки данного исследования.

Структура стека представлена на рис. 1. Не касаясь, как было указано выше, схем самосинхронных массивов памяти, рассмотрим схемы и поведение указателя стека, осуществляющего выработку адреса, и управляющего автомата, что, собственно, и является основной целью статьи.

В приведенной на рис. 1 структуре использованы следующие управляющие сигналы:

W – сигнал записи в стек (PUSH);



■ *Рис. 1. Структура стековой памяти*

R – сигнал чтения из стека (POP);

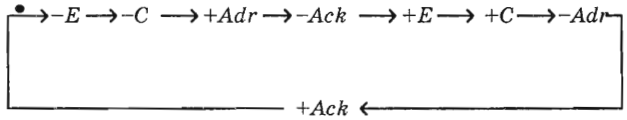
сигналы W и R поступают на управляющий автомат, определяя режим работы указателя стека, и на массив самосинхронной памяти, определяя режим работы памяти;

Ack – сигнал завершения работы памяти, поступающий из матрицы памяти во внешнюю среду;

Adr – множество адресных сигналов, поступающих с указателя стека в матрицу памяти; изменение адресного сигнала управляет работой матрицы памяти, в то время как сигналы W и R определяют режим этой работы;

C – совокупность сигналов управления, поступающих от схемы управления на указатель стека.

Введем сигнал E, обозначающий состояние входа W или входа R ($E = W \wedge R$). Тогда общая спецификация поведения стека может быть задана следующим сигнальным графом:



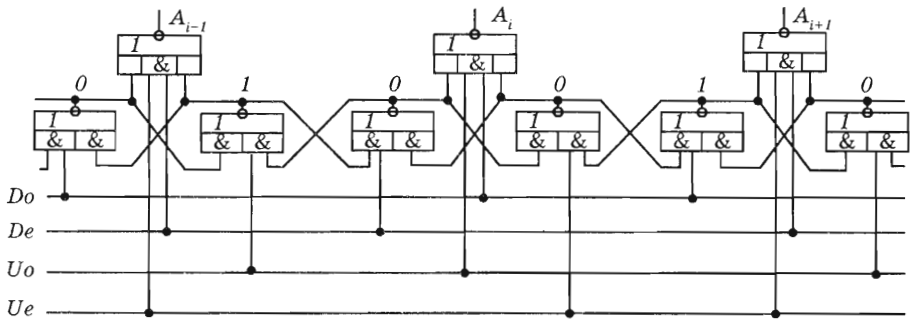
Указатель стека. При выборе или проектировании логической схемы указателя стека должно быть выполнено одно основное требование – схема должна быть максимально простой, быть может, за счет увеличения числа внешних управляющих сигналов и усложнения схем управления этими сигналами. Примером такого указателя стека является схема, приведенная на рис. 2.

Схемной основой указателя стека является многофазный триггер из вентилей ИЛИ-НЕ. В таком триггере возможны устойчивые состояния с нарушением правильного чередования состояний вентилей, например,

...0101010101010010101010101010...

Управление сдвигом пары соседних состояний 00 осуществляется следующим образом: для сдвига влево – принудительный перевод левого вентиля пары 00 в состояние 1 (00 → 10); для сдвига вправо – принудительный перевод правого вентиля пары 00 в состояние 1 (00 → 01). Для предотвращения сквозного сдвига сигналы управления сдвигами разделены на нечетные (odd) Do и Uo и четные (even) De и Ue; сигналы Do и De управляют сдвигом вниз (Down, см. рис. 1), а сигналы Uo и Ue – вверх (Up). Вентили, вырабатывающие адресные сигналы A_i, управляются теми же сигналами сдвига. (Заметим, что использование для этого самостоятельных управляющих сигналов несколько упрощает реализацию указателя стека, но мы не будем здесь рассматривать этот вопрос).

Управляющий автомат. Управляющий автомат осуществляет выработку сигналов управления указателем стека по входным сигналам W и R.



■ ***Рис. 2.*** *Указатель стека*

Задание поведения этого автомата в принципе описывает взаимодействие стека с внешней средой.

Содержательно поведение стека определяется поведением указателя стека следующим образом:

если за операцией записи следует повторная операция записи, то адрес, вырабатываемый указателем стека, смещается на одну позицию вверх;

если за операцией записи следует операция чтения, то указатель стека вырабатывает тот же адрес, что и на предыдущем шаге;

если за операцией чтения следует повторная операция чтения, то адрес, вырабатываемый указателем стека, смещается на одну позицию вниз;

если за операцией чтения следует операция записи, то указатель стека вырабатывает тот же адрес, что и на предыдущем шаге.

С учетом разбиения позиций указателя стека на четные и нечетные, поведение управляющего автомата задается автоматом Мили (рис. 3). Состояния автомата соответствуют следующим ситуациям:

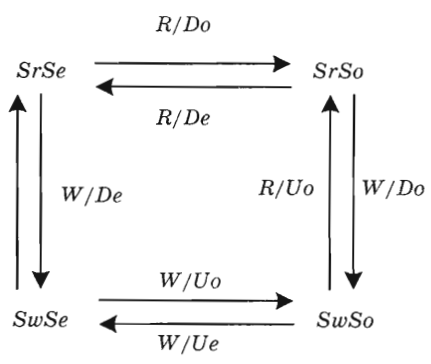
SwSo – запись на нечетной позиции;

SwSe – запись на четной позиции;

SrSo – чтение на нечетной позиции;

SrSe – чтение на четной позиции.

Обратим внимание на тот факт, что рассматриваемый автомат, с точки зрения его формальной спецификации, является синхронным и, следовательно, требует структурирования времени. Такое структурирование может быть обеспечено за счет запрос-ответного взаимодействия с внешней средой (стандартная для самосинхронной структуры четырехфазная дисциплина). Таким образом, перед нами возникает задача поиска самосинхронной реализации синхронного автомата Мили. Учитывая существенное свойство автомата Мили – выходные сигналы связаны с дугами, обозначающими смену состояний, – естественно в качестве базовой схемы, представляющей состояния автомата, использовать сложный триггер, построенный из двух простых триггеров по принципу «основной – вспомогательный».



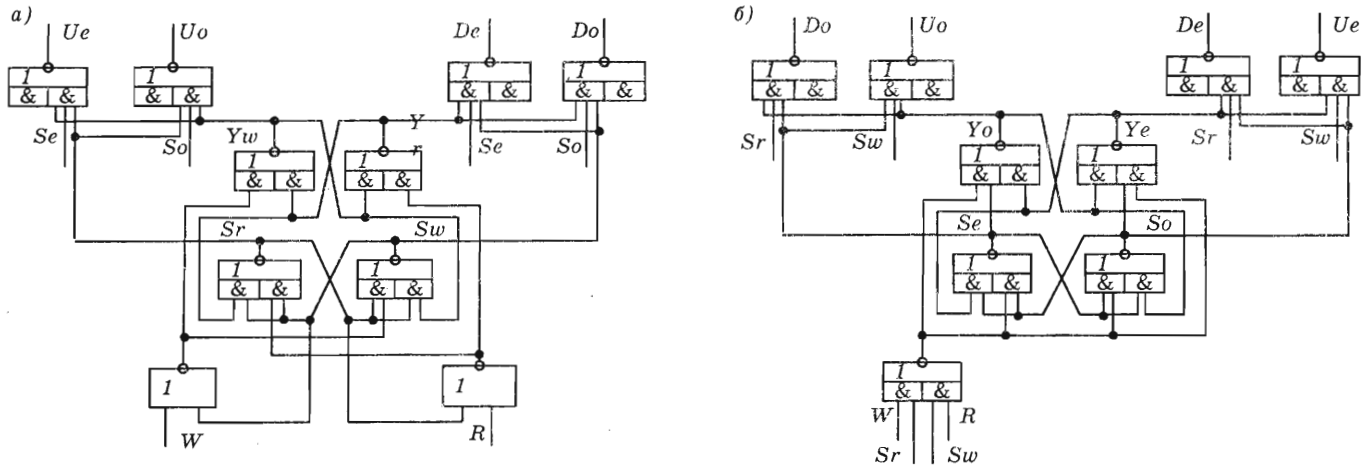
■ ***Рис. 3.*** *Граф управляющего автомата*

На рис. 4, *а* приведены схема триггера, представляющего переменные *Sw*, *Sr*, и вентили, вырабатывающие сигналы на переходах $SwSo \leftrightarrow SrSo$ и $SwSe \leftrightarrow SrSe$. В соответствии с описанным выше алгоритмом выработки адресов для этой схемы может быть построен сигнальный граф, демонстрирующий независимость поведения стека от задержек вентилей на рассматриваемых переходах автомата:

$$\dot{\longrightarrow} -W \rightarrow -Yw \rightarrow +Yr \rightarrow -De(Do) \rightarrow +Adr \rightarrow -Ack$$
$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\uparrow \quad +Ack \leftarrow -Adr \leftarrow +De(Do) \leftarrow -Sr \leftarrow +Sw \leftarrow +W$$
$$\uparrow \qquad \downarrow$$
$$\uparrow \quad -R \rightarrow -Yr \rightarrow +Yw \rightarrow -Ue(Uo) \rightarrow +Adr \rightarrow -Ack$$
$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\uparrow\leftarrow\leftarrow +Ack \leftarrow -Adr \leftarrow +Ue(Uo) \leftarrow -Sw \leftarrow +Sr \leftarrow +R$$

На рис. 4, *б* представлены схема триггера, представляющего переменные *Se*, *So*, и вентили, вырабатывающие сигналы на переходах $SwSe \leftrightarrow SwSo$ и $SrSe \leftrightarrow SrSo$. Условием этих переходов является $F = (R \vee Sw)(W \vee Sr) = 0$.

Аналогично переходам на триггере (*Sw*, *Sr*) строится сигнальный граф, демонстрирующий независимость поведения стека от задержек вентилей на рассматриваемых переходах автомата (*So*, *Se*):



■ *Рис. 4. Схема автомата управления*

$$\xrightarrow{\cdot} -F \to -Yo \to +Ye \to -De(Ue) \to +Adr \to -Ack$$
$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\uparrow \quad +Ack \leftarrow -Adr \leftarrow +De(Ue) \leftarrow -Se \leftarrow +So \leftarrow +F$$
$$\uparrow \quad \downarrow$$
$$\uparrow \quad -F \to -Ye \to +Yo \to -Do(Uo) \to +Adr \to -Ack$$
$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\uparrow\leftarrow\leftarrow +Ack \leftarrow -Adr \leftarrow +Do(Uo) \leftarrow -So \leftarrow +Se \leftarrow +F\ .$$

Объединяя выходные функции на переходах, получаем:

$$De = \overline{YeSoSw \vee YrSwSe},$$
$$Do = \overline{YoSeSw \vee YrSwSo},$$
$$Ue = \overline{YeSoSr \vee YwSrSe},$$
$$Uo = \overline{YoSeSr \vee YwSrSo}.$$

Нетрудно понять, как реализуются функции выхода на дугах, сохраняющих состояние (в нашем примере таких дуг нет).

Естественно, что реальное проектирование устройства требует некоторых дополнительных усилий, связанных с ограничениями на нагрузочные возможности выходов вентилей, оптимизации задержек и т. п. Может оказаться, что с точки зрения распределения нагрузок имеет смысл разделить сигналы U и D не на два, а на три или более выходов, однако все это находится за рамками настоящего изложения и не меняет общего подхода.

Обсуждение результатов

Методика, использованная выше, состояла в следующем: мы провели декомпозицию устройства на два блока – схему управления, описанную как конечный автомат, и собственно самосинхронное устройство. Подобный подход не является методологической новинкой. Достаточно вспомнить память и логическую схему в структуре конечного автомата, регистры и схему управления при использовании языка межрегистровых передач, операционное и управляющее устройства в процессоре и т. д. Декомпозиция не является формальной процедурой. Она базируется на наших представлениях о том, как должно работать проектируемое устройство, нашем опыте и, если хотите, наших пристрастиях к тем или иным типам устройств. Другим побудительным мотивом для использования того или иного типа декомпозиции является возможность использования в разных блоках различных языков спецификации и различных методов синтеза (проектирования), предпочтительных для нас, исходя из тех или иных внемодельных представлений.

При всех очевидных успехах в развитии формальных методов проектирования и их теории высокоэффективное проектирование было и остается искусством, использующим средства формальной поддержки. По крайней мере, системы проектирования в значительной степени используют и должны использовать искусство проектировщика. Надеемся, что читатель простит нас за эти труизмы.

Фактически при проектировании стека мы ввели устройство, согласующее интерфейс собственно самосинхронного стека и интерфейс внешней среды, и специфицировали алгоритм согласования на языке конечных автоматов. Таким образом, задача проектирования самосинхронного устройства с внешними входами сводится к спецификации интерфейсного автомата, самосинхронная реализация которого осуществляется стандартной процедурой. При этом еще раз заметим, что практически все известные процедуры синтеза самосинхронных устройств с внешними входами используют

процедуру преобразования такого устройства в автономное. Эта процедура – развертка (линеаризация) спецификации поведения на все возможные состояния интерфейса. Рассмотренная выше процедура обеспечивает компактное задание для такой спецификации.

Следует обратить внимание на следующий факт. Каждый раз, когда мы линеаризируем сигнальный граф, диаграмму изменений или диаграмму переходов, возникает вопрос о полноте такой развертки. Для того чтобы не потерять какие-то ситуации, мы должны располагать средством, обеспечивающим представление всех возможных интерфейсных ситуаций. Использованный автоматный подход, собственно, и направлен на решение этой задачи. Очевидно, что достаточным (но не необходимым!) условием является прохождение развертки через все дуги автоматного графа.

В принципе, от автоматного описания очевиден переход к стандартной поведенческой спецификации. (Специальным вопросом является упрощение такой спецификации.) Однако, коль скоро мы уже построили автоматную спецификацию интерфейса, имеет смысл использовать изложенный выше подход.

Данный подход не претендует на решение проблемы. Нашей целью было обратить внимание на возникшую возможность и продемонстрировать ее эффективность на примере.

Литература

1. **Варшавский В., Кишиневский М., Мараховский В** и др. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах/ Под ред. В. И. Варшавского. – М.: Наука, 1986. – 398 с.
2. **Varshavsky V. I.** Hardware support of parallel asynchronous processes, Helsinki University of technology, Digital Systems Laboratory, Series-A: Research Reports, – N 2. Sept. 1987.
3. **Chu T. A.** A Design Methodology for VLSI Self-Timed Systems. – Ph.D. Thesis Dep. of EECS, MIT, 1987.
4. **Chu T. A.** On the model for designing VLSI asynchronous digital systems// Integration. The VLSI Journal. – 1986. – Vol. 4. – N 2. – P. 99–113.
5. **Варшавский В.И., Кишиневский М.А., Кондратьев А.Ю.** и др. Модели для спецификации и анализа процессов в асинхронных схемах// Известия АН СССР. Техническая кибернетика. 1988. – № 4. – С. 137–142.
6. **Kishinevsky M. A., Kondratiev A. Yu., Taubin A. R, Varshavsky V. I.** Concurrent Hardware. The Theory and Practice of Self-Timed Design. – J. Wiley & Sons, 1993.
7. **Варшавский В. И.** Апериодические автоматы с самосинхронизацией// Дискретные системы: труды IFAC симпозиума. – Т. 1. – Рига, Zinatne, 1974. – С. 9–25.
8. **Астановский А. Г., Варшавский В. И., Мараховский В. Б.** и др. Апериодические автоматы/ Под ред. В. И. Варшавского. – М.: Наука, 1976. – 423 с.
9. **Chu T. A.** Synthesis of hazard-free control circuits from asynchronous finite state machine specifications// In: Proceedings of «Tau 92: 1992 Workshop on Timing Issues in the Specification and Synthesis of Digital Systems». – Princeton University, March, 1992. – 10 p.
10. **David I., Ginosar R., Yoeli M.** Implementing sequential machines as self-timed circuits // IEEE Trans, on Comput. – 1992. – Vol. 41. – N 1. – P. 12–17.
11. **Ebergen J. C., Gingras S.** An asynchronous stack with constant, response time, 1992 (unpublished manuscript).
12. **Josephs M. B., Udding J. T.** The design of a delay-insensitive stack// In: Jones G., Sheeran S. (eds.), Designing correct circuits, Workshops in Computing. – Springer-Verlag, 1990. – P. 132–152.
13. **А. с. СССР** № 1365129. Запоминающее устройство на МОП-транзисторах / В. И. Варшавский, Н. М. Кравченко, В. Б. Мараховский, Б. С. Цирлин. – 1988. Бюл. № 1.
14. **А. с. СССР** № 1474738. Запоминающее устройство / В. И. Варшавский, Н. М. Кравченко, В. Б. Мараховский, Б. С. Цирлин. – 1989, Бюл. № 15.

УДК 681.3

КМОП ПОРОГОВЫЕ ЭЛЕМЕНТЫ С ФУНКЦИОНАЛЬНЫМИ ВХОДАМИ[1]

В. И. Варшавский,
доктор техн. наук, профессор
В. Б. Мараховский,
доктор техн. наук, профессор
университет Айдзу (Япония)
И. С. Левин
канд. техн. наук, профессор
Бар-Иланский университет (Израиль)

Предлагается метод увеличения функциональных возможностей порогового элемента за счет введения так называемых функциональных входов. Каждый такой вход соответствует булевой сумме (или произведению) некоторого подмножества входных переменных. Показано, что бета-управляемый пороговый элемент с функциональными входами способен реализовать произвольные монотонные булевы функции. Предложены КМОП реализации таких элементов и методы токовой стабилизации функциональных входов. Приводятся примеры реализации сложных логических функций на основе бета-управляемых пороговых элементов с функциональными входами. Представлены также результаты SPICE-моделирования поведения предложенного элемента.

A method for increasing the functional capability of threshold elements by introducing so-called functional inputs is proposed. Each functional input corresponds to a Boolean sum (or product) of a particular subset of input variables. It is shown that introducing functional inputs enables expansion of the functional capability of beta-driven elements up to the capability to implement an arbitrary monotonic function. The CMOS based implementation of the beta-driven threshold element with newly proposed functional inputs is presented. Methods of the current stabilization of functional inputs are proposed. The paper presents examples of the SPICE simulation of the proposed threshold element behavior.

Введение

Не утихающий в течение десятилетий интерес к пороговым элементам и пороговой логике [1–6] вызван, на наш взгляд, во-первых, более широкими по сравнению с традиционными базисами (И, И-НЕ, ИЛИ, ИЛИ-НЕ и т. д.) функциональными возможностями порогового элемента и, во-вторых, тем, что пороговые элементы являются функциональной основой искусственных нейронных сетей.

Однако в утверждении, что пороговый базис обладает функциональными преимуществами перед традиционными базисами, содержится некоторое лукавство. Действительно, даже если синтез схем в пороговом базисе более эффективен, чем в традиционных, любая пороговая функция может быть реализована в базисе традиционных операций. Следовательно, прежде всего, эффективность использования порогового базиса непосредственно зависит от сложности реализации порогового элемента.

В работах [7–12] нами был предложен β-управляемый КМОП пороговый элемент, требующий только одного транзистора на функциональный вход с весом входа, определяемым шириной этого транзистора. Трудно представить себе более простую реализацию.

Основой для β-управляемой реализации явилось достаточно простое преобразование обычного аналитического представления пороговой функции

[1] Полностью статья публикуется впервые, небольшие фрагменты были опубликованы:

Varshavsky V., Marakhovsky V., Levin I. CMOS Based Beta-Driven Threshold Elements with Functional Inputs', Proceedings of thee 22-th IEEE Convention of Electrical and Electronics Engineering in Israel, Tel Aviv, December, IEEE , 2002. P. 111–113.

Varshavsky V., Marakhovsky V., Levin I. Artificial Neurons Based on CMOS Beta-Driven Threshold Elements with Functional Inputs', WSEAS Transactions on Systems, Issue 2, Vol. 3. April 2004. P. 142–148.

в форме отношения [7]. В традиционном представлении пороговой функции

$$Y = Sign(\sum_{j=0}^{n-1} \omega_j x_j - \eta),\ \ Sign(A) = \begin{cases} 1, & \text{если } A \geq 0 \\ 0, & \text{если } A < 0 \end{cases},$$

где ω_j – вес j-го входа; η – порог.

Выделим некоторое произвольное подмножество переменных S, для которого

$$\sum_{i \in S} \omega_i = \eta\ . \qquad (1)$$

Тогда

$$\sum_{j=0}^{n-1} \omega_j x_j - \eta = \sum_{k \notin S} \omega_k x_k - (\eta - \sum_{i \in S} \omega_i x_i) =$$

$$= \sum_{k \notin S} \omega_k x_k - \sum_{i \in S} \omega_i (1 - x_i) = \sum_{k \notin S} \omega_k x_k - \sum_{i \in S} \omega_i \bar{x}_i, \qquad (2)$$

откуда

$$Y = Sign(\sum_{j=0}^{n-1} \omega_j x_j - \eta) = Rt\left(\frac{\sum_{k \notin S} \omega_k x_k}{\sum_{i \in S} \omega_i \bar{x}_i}\right), \qquad (3)$$

где $$Rt(B) = \begin{cases} 1, \text{ если } B \geq 1 \\ 0, \text{ если } B < 1 \end{cases}.$$

Заметим, что в формуле (3) при физической реализации могут возникнуть проблемы, связанные с неопределенностью типа $0/0$. Для исключения такой опасности достаточно сместить значение порога на некоторую величину $0 < \delta < 1$ в исходном определении пороговой функции:

$$Y = Sign(\sum_{j=0}^{n-1} \omega_j x_j - \eta + \delta) = Rt\left(\frac{\delta + \sum_{k \notin S} \omega_k x_k}{\sum_{i \in S} \omega_i \bar{x}_i}\right), \qquad (4)$$

где $$Rt(B) = \begin{cases} 1, \text{ если } B > 1 \\ 0, \text{ если } B < 1 \end{cases}.$$

Определение (4) для $Rt(B)$, в отличие от (3), симметрично, что делает тривиальным инвертирование функции

$$\overline{Rt\left(\frac{\delta + \sum_{k \notin S} \omega_k x_k}{\sum_{i \in S} \omega_i \bar{x}_i}\right)} = Rt\left(\frac{\sum_{i \in S} \omega_i \bar{x}_i}{\delta + \sum_{k \notin S} \omega_k x_k}\right). \qquad (5)$$

Для пояснения сказанного рассмотрим численный пример:

$$Y = Sign(x_0 + x_1 + 2x_2 + 2x_3 + 4x_4 - 6) =$$

$$= x_0 x_1 x_2 x_3 + x_0 x_1 x_4 + x_2 x_4 + x_3 x_4. \qquad (6)$$

Условию (1) отвечают следующие подмножества переменных: $\{x_0, x_1, x_2, x_3\}$, $\{x_0, x_1, x_4\}$, $\{x_2, x_4\}$, $\{x_3, x_4\}$и форма отношения может быть построена по любому из них

$$Y = Sign(x_0 + x_1 + 2x_2 + 2x_3 + 4x_4 - 6) =$$

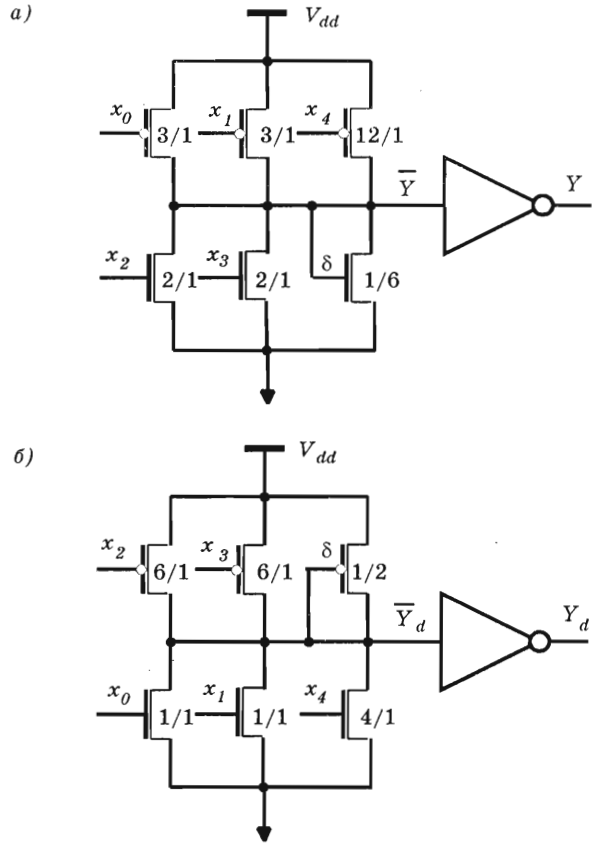
$$= Rt(\frac{\delta + 4x_4}{\bar{x}_0 + \bar{x}_1 + 2\bar{x}_2 + 2\bar{x}_3}) = Rt(\frac{\delta + 2x_2 + 2x_3}{\bar{x}_0 + \bar{x}_1 + 4\bar{x}_4}) =$$

$$= Rt(\frac{\delta + x_0 + x_1 + 2x_3}{2\bar{x}_2 + 4\bar{x}_4}) = Rt(\frac{\delta + x_0 + x_1 + 2x_2}{2\bar{x}_3 + 4\bar{x}_4}). \qquad (7)$$

Как следует из формулы (5),

$$\overline{Y} = \overline{Rt(\frac{\delta + 2x_2 + 2x_3}{\bar{x}_0 + \bar{x}_1 + 4\bar{x}_4})} = Rt(\frac{\bar{x}_0 + \bar{x}_1 + 4\bar{x}_4}{\delta + 2x_2 + 2x_3}). \qquad (8)$$

Вытекающая из представления в форме отношения β-управляемая реализация основывается на замене отношения взвешенных сумм на отношение суммарных проводимостей n- и p-каналов КМОП элемента (рис. 1, *а*)[1].



■ ***Рис. 1.*** *КМОП реализация для отношения подвижностей носителей зарядов n- и p-транзисторов 1/3: а – для функции (8); б – для функции, двойственной (8)*

[1] Смещение δ обеспечивает слабый транзистор, затвор которого подключен либо к источнику опорного напряжения, либо к выходу делителя (диодное включение).

■ *Таблица 1. Таблица истинности для схемы на рис. 1, а*

$x_0x_1 \backslash x_2x_3x_4$	000		100		010		110		001		101		011		111	
00	6	δ	6	2+δ	6	2+δ	6	4+δ	2	δ	2	2+δ	2	2+δ	2	4+δ
	0		0		0		0		0		1		1		1	
10	5	δ	5	2+δ	5	2+δ	5	4+δ	1	δ	1	2+δ	1	2+δ	1	4+δ
	0		0		0		0		0		1		1		1	
01	5	δ	5	2+δ	5	2+δ	5	4+δ	1	δ	1	2+δ	1	2+δ	1	4+δ
	0		0		0		0		0		1		1		1	
11	4	δ	4	2+δ	4	2+δ	4	4+δ	0	δ	0	2+δ	0	2+δ	0	4+δ
	0		0		0		1		1		1		1		1	

■ *Таблица 2. Таблица истинности для схемы на рис. 1, б*

$x_0x_1\backslash x_2x_3x_4$	000		100		010		110		001		101		011		111	
00	4+δ	0	2+δ	0	2+δ	0	δ	0	4+δ	4	2+δ	4	2+δ	4	δ	4
	0		0		0		0		0		1		1		1	
10	4+δ	1	2+δ	1	2+δ	1	δ	1	4+δ	5	2+δ	5	2+δ	5	δ	5
	0		0		0		1		1		1		1		1	
01	4+δ	1	2+δ	1	2+δ	1	δ	1	4+δ	5	2+δ	5	2+δ	5	δ	5
	0		0		0		1		1		1		1		1	
11	4+δ	2	2+δ	2	2+δ	2	δ	2	4+δ	6	2+δ	6	2+δ	6	δ	6
	0		0		0		1		1		1		1		1	

Функционирование схемы, представленной на рис. 1, *а*, ясно из табл. 1. Каждая клетка таблицы, соответствующая одному набору значений входных переменных, разделена на три подклетки. Верхняя левая подклетка содержит число единичных проводимостей открытых *p*-транзисторов для данного набора входных переменных. Верхняя правая подклетка содержит аналогичное число для *n*-транзисторов[1]. Нижняя подклетка содержит значение выходной функции (выхода инвертора).

Заметим, что в отличие от функции (8) перестановка *n*- и *p*-канальных частей схемы (см. рис. 1, *а*) с соответствующим изменением типов транзисторов и их ширин (см. рис. 1, *б*) не приводит к инвертированию функции. Как нетрудно видеть из табл. 2, мы получаем при этом функцию, двойственную исходной:

$$Y_1 = x_0x_2x_3 + x_1x_2x_3 + x_0x_4 + \\ + x_1x_4 + x_2x_4 + x_3x_4 = \\ = Sign(x_0 + x_1 + 2x_2 + 2x_3 + 4x_4 - 5). \quad (9)$$

Из определения двойственной функции и функции (6) следует

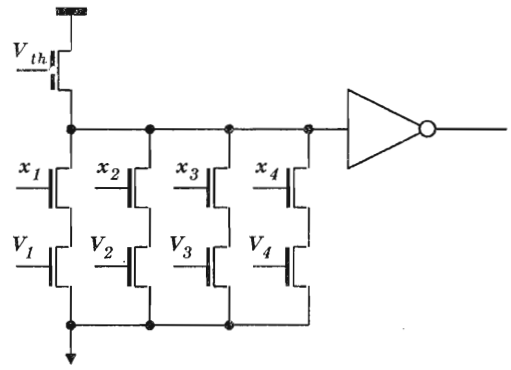
$$Y_d = \overline{\bar{x}_0\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_0\bar{x}_1\bar{x}_4 + \bar{x}_2\bar{x}_4 + \bar{x}_3\bar{x}_4} = \\ = (x_0 + x_1 + x_2 + x_3)(x_0 + x_1 + x_4)(x_2 + x_4)(x_3 + x_4) = \\ = x_0x_2x_3 + x_1x_2x_3 + x_0x_4 + x_1x_4 + x_2x_4 + x_3x_4 = Y_1. \quad (10)$$

В пороговом базисе двойственная функция сохраняет веса входов, а порог вычисляется как

$$[Sign(\sum_{j=0}^{n-1}\omega_j x_j - \eta)]_d = Sign(\sum_{j=0}^{n-1}\omega_j x_j - \sum_{j=0}^{n-1}\omega_j + \eta - 1). \quad (11)$$

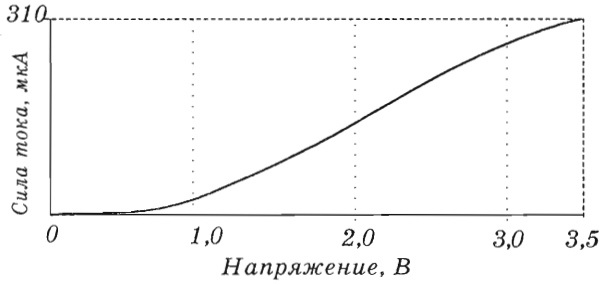
Здесь уместно также обратить внимание на то, что на рис. 1, *б* суммарная ширина *n*-транзисторов равна не пяти [значение порога в функции (9)], а

[1] Заметим, что в данном примере рассматриваются транзисторы, у которых подвижность носителей (а следовательно, и проводимость) полностью открытого *n*-транзистора в 3 раза выше, чем у *p*-транзистора при равной длине канала.



■ ***Рис. 2.*** *Реализация пороговой функции*

$$Y = Sign[\sum_{j=1}^{4} \varphi_n(V_j)x_j - \varphi_p(V_{th})]$$



■ ***Рис. 3.*** *Зависимость тока (пропорционального β) через управляемую пару транзисторов от управляющего напряжения (результат SPICE-моделирования)*

шести. Это связано с тем, что перенос слабого транзистора из n-цепи в p-цепь увеличивает порог на единицу[1].

Другой и, быть может, определяющей чертой пороговых функций является их функциональная пластичность, т. е. возможность изменения реализуемой пороговым элементом функции простым изменением весов входов и порога. И опять-таки это свойство становится значимым только в случае, если управление весами входов и порогом может быть осуществлено достаточно просто.

Поскольку проводимость транзистора монотонно зависит от напряжения на его затворе, то очевидна простота, с которой может быть осуществлено управление весом входа в β-управляемых КМОП пороговых элементах [9, 10] (рис. 2).

На рис. 3 приведена зависимость тока (пропорционального β) через пару транзисторов от напряжения на затворе управляющего транзистора V_i при пороговом напряжении на входе инвертора (1,75 В), полученная SPICE моделированием.

Заметим, что в схеме на рис. 2 мы несколько отошли от анонсированного вначале принципа β-управляемой реализации, связанного с представлением пороговой функции в форме отношения. В схеме на рис. 2 p-транзистор определяет только величину порогового тока.

Пороговый элемент реализует пороговую функцию, которая является монотонной. Однако, не все монотонные функции являются пороговыми. Уже для четырех переменных есть простейшая монотонная функция, которая не является пороговой, а именно:

$$y = x_0x_1 + x_2x_3. \qquad (12)$$

Нетрудно видеть, что это функция описывает поведение двухвходового мультиплексора – схемы, очень важной во многих применениях. Монотонными, но не пороговыми, кстати, являются все функции, описывающие поведение k-входовых мультиплексоров. В этой работе мы попытаемся показать, что, используя достаточно простые дополнительные схемотехнические приемы, можно расширить функциональные возможности β-управляемых элементов до реализации произвольных монотонных функций.

Пороговые элементы с функциональными входами

Прежде всего рассмотрим ряд функциональных и схемных примеров.

Схема на рис. 4, *а* реализует симметрическую монотонную функцию ранга 2

$$\begin{aligned} Y_1 &= x_0x_1 + x_0x_2 + x_0x_3 + x_0x_4 + x_1x_2 + \\ &+ x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 = \\ &= Sign(x_0 + x_1 + x_2 + x_3 + x_4 - 2). \end{aligned} \qquad (13)$$

Разделим n-канальную часть схемы на два канала, включив в каждый из них стабилизатор тока, ограничивающий ток в канале единичным значением (рис. 4, *б*). При этом произойдет очевидное изменение функции (13):

$$Y_2 = Sign(x_0 + z_1 + z_2 - 2) = x_0z_1 + x_0z_2 + z_1z_2, \qquad (14)$$

где $z_1 = x_1 + x_2$ и $z_2 = x_3 + x_4$, откуда

$$\begin{aligned} Y_2 &= x_0x_1 + x_0x_2 + x_0x_3 + x_0x_4 + \\ &+ x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4. \end{aligned} \qquad (15)$$

Замечательным в этом примере является то, что функция Y_2 (14) является монотонной, но не является пороговой.

Теперь рассмотрим реализацию функции, двойственной (13) (рис. 5, *а*):

$$\begin{aligned} Y_3 &= Sign(x_0 + x_1 + x_2 + x_3 + x_4 - 4) = \\ &= x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_0x_1x_3x_4 + \\ &+ x_0x_2x_3x_4 + x_1x_2x_3x_4. \end{aligned} \qquad (16)$$

Разделим p-цепь на два канала, включив в каждый из них стабилизатор тока, ограничивающий

[1] Это, казалось бы, мелкое замечание оказывается весьма полезным для увеличения границ реализуемости [7].

ток в канале единичным значением (рис. 5, *б*). При этом, как и в первом случае (см. рис. 4, *а*), произойдет изменение реализуемой функции[1]:

$$Y_4 = Sign(x_0 + z_1 + z_2 - 2) = x_0 z_1 + x_0 z_2 + z_1 z_2, \quad (17)$$

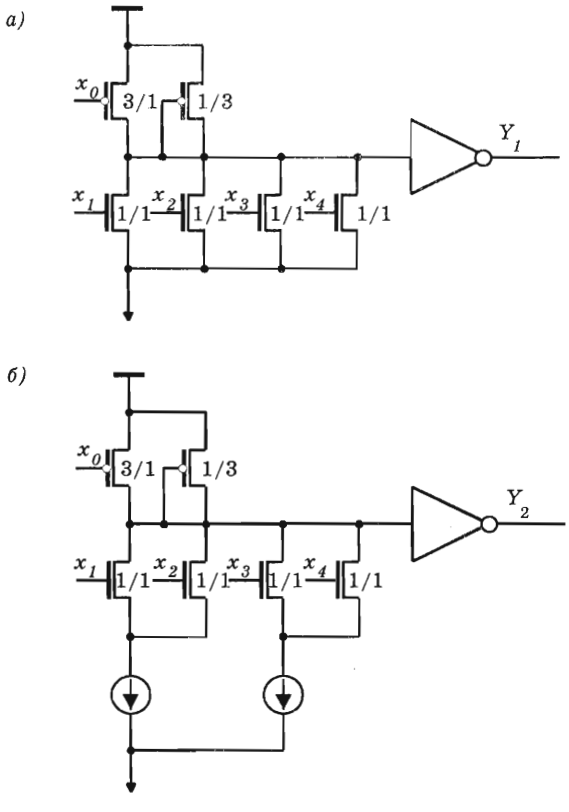
где $z_1 = \overline{\overline{x_1} + \overline{x_2}} = x_1 x_2$ и $z_2 = \overline{\overline{x_3} + \overline{x_4}} = x_3 x_4$, следовательно

$$Y_4 = x_0 x_1 x_2 + x_0 x_3 x_4 + x_1 x_2 x_3 x_4. \quad (18)$$

Опять заметим, что функция (18) монотонная, но не пороговая.

Из приведенных примеров видно, что, если выделить некоторое подмножество n-транзисторов, управляемых переменными $x_j \in R_k$ и, используя стабилизатор тока, ограничить суммарный ток через это подмножество транзисторов единичным значением, то в формировании выходной функции все это подмножество переменных будет участвовать, как одна переменная

$$z_k = \bigvee_{x_j \in R_k} x_j. \quad (19)$$



■ *Рис. 4. Реализация функции (13) (а) и реализация функции (14) (б)*

[1] Заметим, что функция (17) совпадает с (14) в силу самодвойственности мажоритарной функции для нечетного числа переменных.

С другой стороны, если выделить некоторое подмножество p-транзисторов, управляемых переменными $x_i \in R_m$ и, используя стабилизатор тока, ограничить суммарный ток через это подмножество транзисторов единичным значением, то в формировании выходной функции все это подмножество переменных будет участвовать как одна переменная

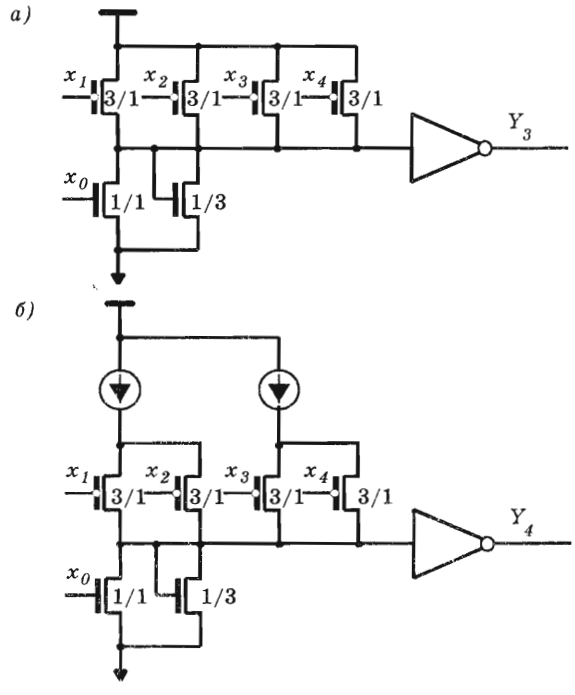
$$z_m = \overline{\bigvee_{x_i \in R_m} \overline{x_i}} = \&_{x_i \in R_m} x_i. \quad (20)$$

Подмножество параллельных транзисторов, ток через которые ограничен стабилизатором тока и подмножество входов которых принимает участие в формировании выхода как одна переменная, мы будем называть функциональным входом.

Каков при этом смысл функционального входа? Ведь тоже самое может быть реализовано при помощи дополнительного И (ИЛИ) элемента. Это действительно так, однако, используя функциональные входы, мы сокращаем глубину схемы (задержку), потребляемую мощность (число элементов) и, правда в незначительной мере, число транзисторов.

Кроме того, возникает естественный вопрос: «Может ли произвольная монотонная функция быть реализована на одном пороговом элементе с функциональными входами?»

Для ответа на этот вопрос рассмотрим монотонную функцию



■ *Рис. 5. Реализация функции (16) (а) и функции (17) (б)*

$$F(X) = \bigvee_{j=0}^{k-1} c_j = \&_{i=0}^{m-1} d_i, \qquad (21)$$

где c_j – множество конъюнкций в минимальной дизъюнктивной форме, а d_i – множество дизъюнкций в минимальной конъюнктивной форме функции[1] $F(X)$. Пусть c_j и d_i – p- и n-функциональные входы соответственно. Тогда

$$F(X) = Rt\left(\frac{\sum_{j=0}^{k-1}\overline{c}_j + \delta}{k}\right) = Rt\left(\frac{m}{\sum_{i=0}^{m-1} d_i + \delta}\right). \qquad (22)$$

Из формулы (22) следует лишь принципиальный положительный ответ на поставленный выше вопрос, не касаясь проблем реализуемости и сложности реализации. Здесь мы заметим только как преимущество предлагаемого схемного решения, что реализуемость не зависит от ранга конъюнкции (дизъюнкции), а зависит лишь от их числа в минимальной форме[2].

Стабилизация тока в функциональном входе

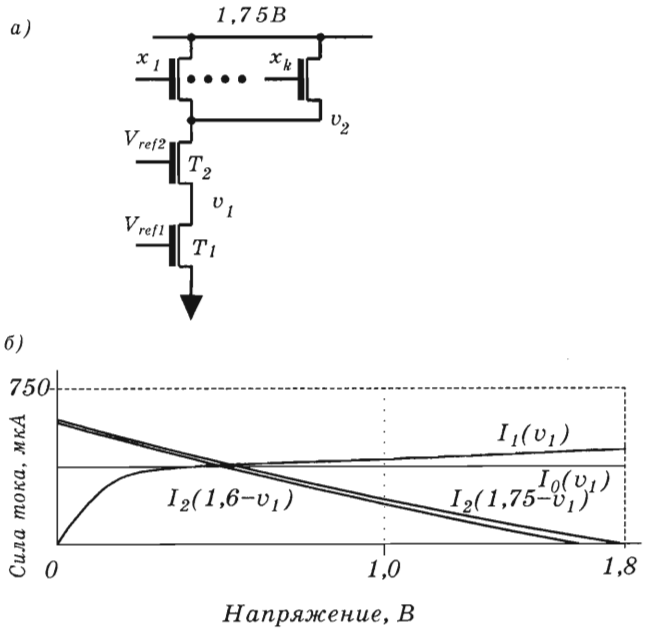
Стабилизатор порогового тока, состоящий из двух последовательно соединенных транзисторов, уже использовался нами в β-управляемых пороговых элементах [13–16]. Однако в этих работах стабилизатор тока применялся только для увеличения крутизны характеристики β-компаратора в точке переключения. Здесь же мы намерены использовать стабилизатор для формирования (ограничения) единичного рабочего тока и, следовательно, необходимо обеспечить более высокую степень стабилизации.

Казалось бы, что стабилизацию тока обеспечивает любой МОП-транзистор в режиме насыщения. Действительно, ток через транзистор T_1 (рис. 6, *а*) при $\upsilon_1 \geq V_{ref1} - V_{th}$ (режим насыщения) в соответствии с простейшим уравнением транзистора $I_1 = \frac{\beta}{2}(V_{ref1} - V_{th})^2$ не зависит от υ_1. Однако учет эффектов второго порядка опровергает это утверждение. На рис. 6, *б* приведены результаты SPICE моделирования (уровень 7) схемы (см. рис. 6, *а*) для n-транзисторов, модель которых взята из MOSIS:

«MOSIS PARAMETRIC TEST RESULTS
RUN: N82L
VENDOR: TSMC
TECHNOLOGY: SCN035H
FEATURE SIZE: 0.4 microns
DATE: 98 May 20
LOT: N82L
WAF: 97»

В эксперименте на участке насыщения $dI_1(\upsilon_1)/d\upsilon_1 \approx 0{,}06$ мкА/мВ, что для $\delta\upsilon_2 = 150$ мВ составляет 9 мкА или порядка ≈2,5% от единичного тока ($I_0 = 380$ мкА). Предположим, что падение напряжения на функциональной части (1,75 В–υ_2) при одном открытом p-транзисторе равно 150 мВ[3] и уменьшается до нуля по мере роста числа открытых транзисторов. На рис. 6, *б* приведены две кривые для тока через T_2 в зависимости от υ при напряжениях на стабилизаторе тока 1,75 В и 1,6 В [$I_2(1{,}75-\upsilon_1)$ и $I_2(1{,}6-\upsilon_1)$ соответственно]. Пересечение этих кривых с кривой $I_1(\upsilon_1)$ определяет токи, протекающие через стабилизатор при указанных напряжениях. Разброс токов при этом равен 3,8 мкА, т. е. 1 %. Результат не выдающийся, но во многих применениях вполне удовлетворительный. Заметим опять, что поскольку $\delta\upsilon_2$ зависит от ширины единичного функционального транзистора, а разброс токов зависит от $\delta\upsilon_2$ то, в принципе, этот разброс за счет потери площади может быть снижен.

В эксперименте, результаты которого приведены на рис. 6, *б*, $V_{ref1} = 0{,}855$ В и $V_{ref2} = 3{,}5$ В. Эти значения выбраны нами экспериментально, исхо-

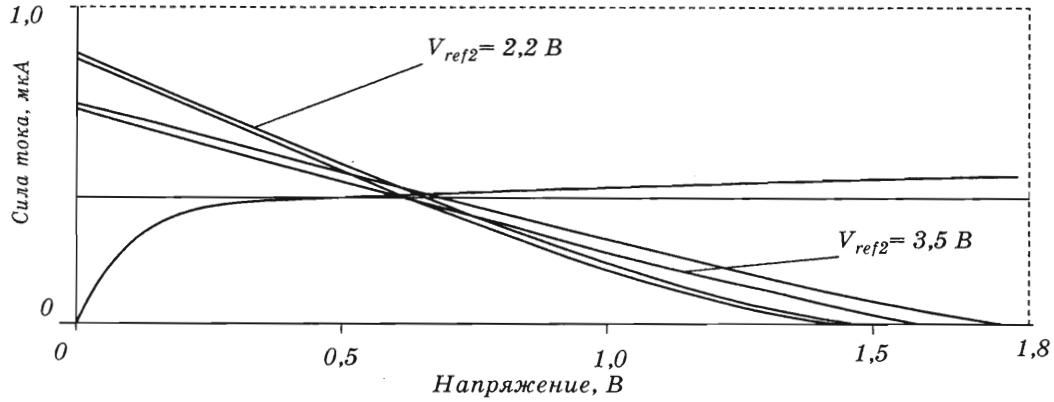


■ ***Рис. 6.*** *Стабилизатор тока: а – схема функционального входа со стабилизатором тока на двух последовательных транзисторах; б – токовые характеристики стабилизатора*

[1] Для монотонной функции обе минимальные формы единственны.

[2] Сказанное весьма напоминает проблему реализуемости для псевдо-n-МОП ИЛИ-НЕ и псевдо-p-МОП И-НЕ элементов [19].

[3] Это напряжение может регулироваться выбором ширины транзистора.



■ *Рис. 7. Поведение токового стабилизатора при $\delta\upsilon_2$ = 150 мВ, V_{ref2} = 3,5 В и V_{ref2} = 2,2 В*

дя из того, что по причинам, которые будут ясны ниже, V_{ref2} = 3,5 В. Теперь рассмотрим вопрос о выборе установочных напряжений более детально.

Ток насыщения через транзистор T_1 (и, следовательно, через стабилизатор) в рабочей точке (в режиме насыщения) должен быть равен единичному функциональному току, т. е. $\frac{\beta_1}{2}(V_{ref1} - V_{th})^2 = I_0$. Тогда из условия режима насыщения $\upsilon_1 \geq V_{ref1} - V_{th}$ следует[1]

$$V_{ref1} \leq \sqrt{\frac{2I_0}{\beta_1}} + V_{th}. \qquad (23)$$

Теперь рабочая точка определяется пересечением зависимостей токов в транзисторах T_1 $I_1(\upsilon_1)$ и T_2 $I_2(\upsilon_2 - \upsilon_1)$. Точность стабилизации при этом зависит от крутизны $I_2(\upsilon_2 - \upsilon_1)$ и $I_2(\upsilon_2 - \delta\upsilon_2 - \upsilon_1)$ и расстояния между ними. $I_2(\upsilon_2 - \upsilon_1) = 0$, если с учетом эффекта подложки

$$V_{ref2} \leq V_{th} + \upsilon_1 + \sqrt{\upsilon_1}, \qquad (24)$$

поэтому уменьшение V_{ref2} приводит к смещению начала кривой $I_2(\upsilon_2 - \upsilon_1)$ влево. Для сохранения положения рабочей точки теперь необходимо увеличение крутизны характеристики транзистора (w, β).

На рис. 7 приведены результаты SPICE-моделирования токового стабилизатора для двух значений V_{ref2} – 3,5 и 2,2 В[2]. При этом для V_{ref2} = 2,2 В разброс токов по сравнению с V_{ref2} = 3,5 В уменьшился в два раза (1,9 мкА или 0,5 % для $\delta\upsilon_2$ = 150 мВ).

Как следует из сказанного выше, характеристики токового стабилизатора при V_{ref2} = 2,2 В лучше, чем при V_{ref2} = 3,5 В, однако использование в качестве V_{ref2} напряжения V_{dd} в ряде случаев[3] может быть оправдано.

Во-первых, чем выше V_{ref2}, тем меньше может быть ширина транзистора T_2 в токовом стабилизаторе.

Во-вторых, вместо V_{ref2} может быть подана дополнительная переменная y. В этом случае поведение функционального входа в n-канальной части элемента, как это следует из формулы (19), описывается выражением

$$z_k = \left(\bigvee_{x_j \in R_k} x_j\right) \cdot y; \qquad (25)$$

и аналогично для функционального входа в p-канальной части элемента, как это следует из (20):

$$z_m = \overline{\left(\bigvee_{x_i \in R_m} \bar{x}_i\right) \cdot \bar{y}} = \&_{x_i \in R_m} x_i + y. \qquad (26)$$

Указанная модификация функционального входа не всеобъемлюща, но в ряде случаев может оказаться весьма полезной.

В том случае, если увеличение V_{ref2} по тем или иным причинам нежелательно, а изменение поведения функционального входа типа (25), (26) полезно, то для дополнительного входа y может быть использован преобразователь уровня на истоковом повторителе (рис. 8).

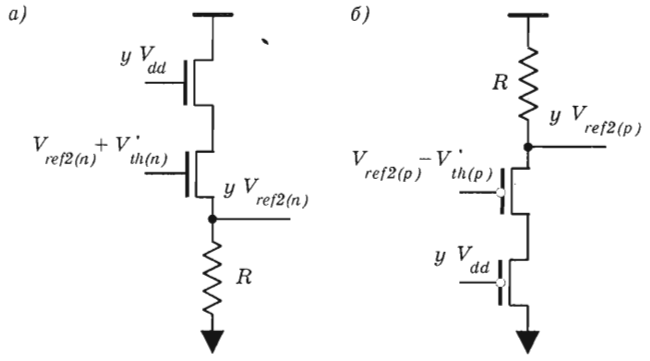
В схемах на рис. 8, *а* и *б* приведены преобразователи уровня для управляемых установочных напряжений для n- и p-функциональных входов соответственно. При этом для достаточно больших R выходное напряжение преобразователя, как это показано на рис. 8, зависит от установочных напряжений и порогов транзисторов с учетом эффекта подложки.

Вернемся, однако, к поведению функционального входа. На рис. 9 приведены результаты его SPICE-

[1] Для определения ограничений на установочное напряжение для нас вполне достаточна точность простейшего уравнения транзистора.

[2] Все остальные параметры моделирования те же, что и на рис. 6, *б*.

[3] Например, при сниженных требованиях к точности поддержания тока.



■ *Рис. 8. Преобразователь уровня сигнала: а – на n-транзисторах; б – на p-транзисторах*

моделирования. Моделировалось поведение функционального входа (ток через функциональный вход и падение напряжения на токовом стабилизаторе) в зависимости от числа открытых параллельных *n*-транзисторов при пороговом напряжении на выходе β-компаратора, равном 1,75 В. Темп включения транзисторов приведен в табл. 3.

■ *Таблица 3. Моменты включения транзисторов*

Время, нс	0	100	200	300	400	500	600
Число открытых транзисторов	0	1	3	7	15	31	63

Заметим, что в разных примерах SPICE-моделирования мы видим различные значения рабочего тока. Это естественно, так как ток через стабилизатор зависит от его параметров. С другой стороны, это сделано намеренно для того, чтобы обратить внимание на возможность вводить в пороговую функцию функциональную переменную с различными весами.

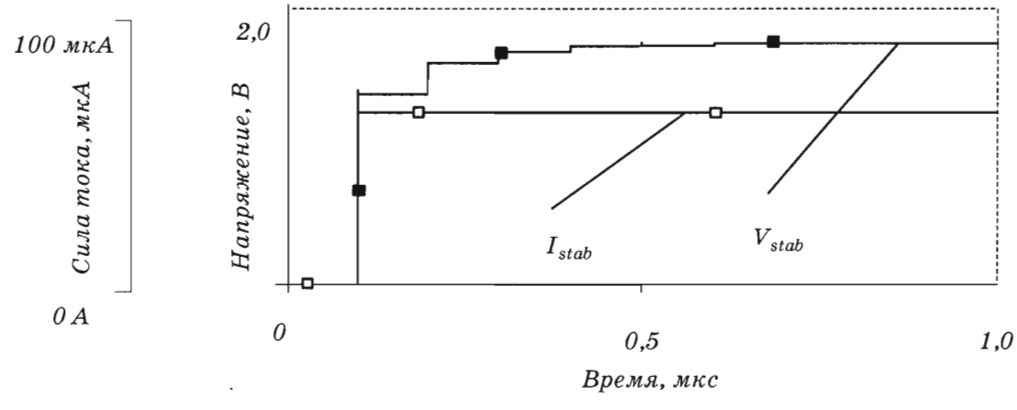
Параметрами функционального входа, определяющими протекающий через него ток, являются ширины функциональных транзисторов и согласованные с ними параметры стабилизатора тока (ширины транзисторов и установочных напряжений). При проектировании порогового элемента мы можем варьировать любые из указанных параметров. Однако, если мы хотим оперативно управлять весом функционального входа в процессе адаптации или обучения для создания искусственного нейрона с функциональными входами, то мы, по-видимому, можем эффективно воспользоваться только установочным напряжением.

Результаты моделирования (см. рис. 6, 7) наводят на мысль, что эффективное управление током через токовый стабилизатор может быть осуществлено путем изменения тока насыщения нижнего транзистора T_1 изменением V_{ref1}, с сохранением неизменной вольт-амперной характеристики верхнего транзистора T_2. Характеристики управляемого токового стабилизатора, полученные SPICE-моделированием, приведены на рис. 10.

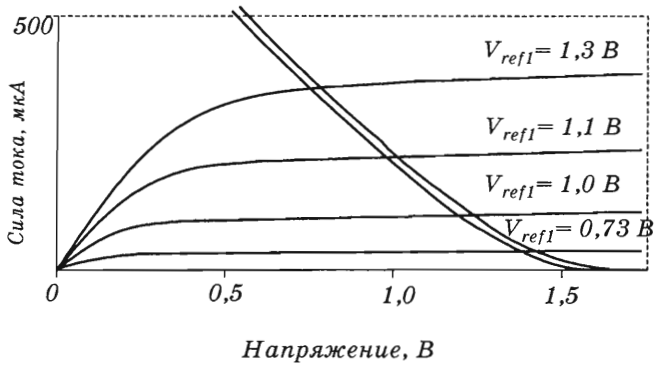
В данном эксперименте при изменении установочного напряжения от 0,73 до 1,3 В ток через токовый стабилизатор изменяется от 35 до 350 мкА (см. рис. 10). При изменении падения напряжения на токовом стабилизаторе $\delta\upsilon_2 = 200$ мВ изменение тока варьируется от 0,3 до 1,5 мкА соответственно, т. е. не превышает 1 % от текущего значения тока во всем диапазоне.

На рис. 11 приведена зависимость тока через токовый стабилизатор от управляющего напряжения для тех же параметров схемы, что и на рис. 10. При изменении установочного напряжения от нуля до 1,4 В ток изменяется от нуля до 410 мкА с максимальным отклонением (для максимального тока) 3 мкА/200 мВ.

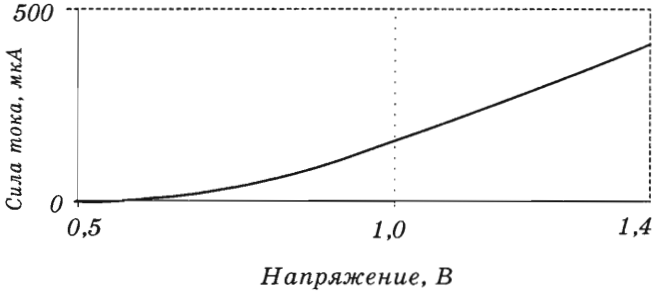
В приведенных выше примерах токовый стабилизатор включен между собственно функциональным входом и «землей». Однако, как это видно из



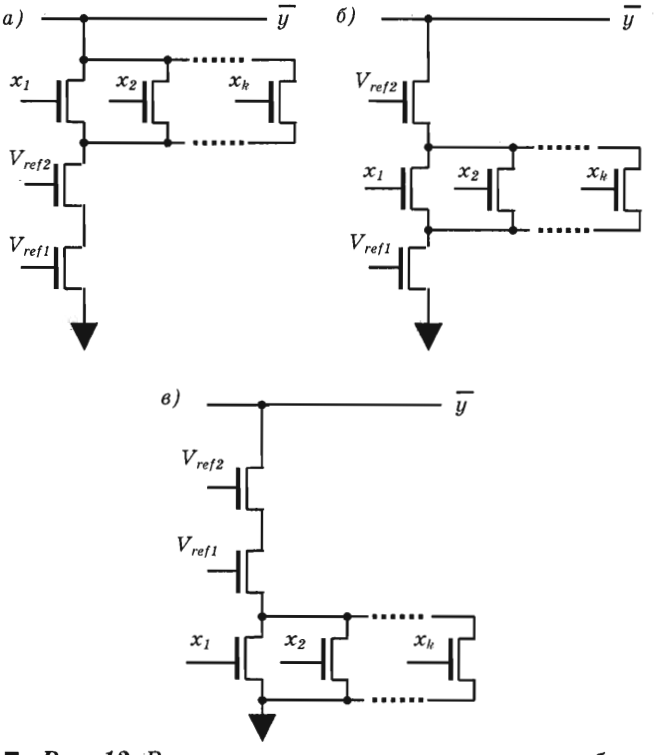
■ *Рис. 9. Ток через токовый стабилизатор и падение напряжения на нем в зависимости от числа открытых транзисторов, $\delta\upsilon_2 = 345$ мВ, $\Delta I = 50$ нА (≈ 0,1 %) : □ – ΔI; ■ – υ_2*



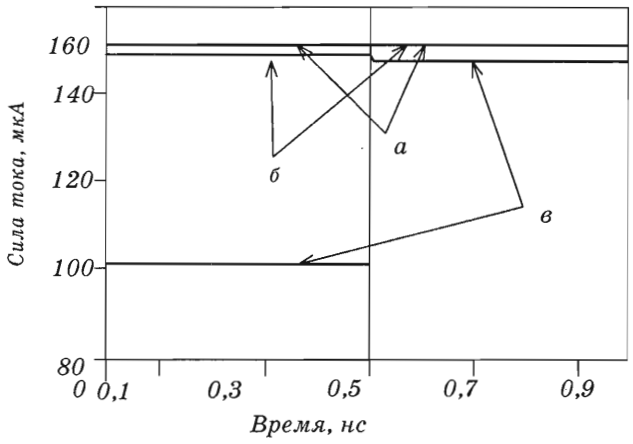
■ ***Рис. 10.*** *Вольт-амперная характеристика управляемого токового стабилизатора (результаты SPICE-моделирования)*



■ ***Рис. 11.*** *Зависимость тока через токовый стабилизатор от V_{ref1}*



■ ***Рис. 12.*** *Варианты включения токового стабилизатора*



■ ***Рис. 13.*** *Результаты SPICE-моделирования схем, представленных на рис. 12*

рис. 12, существуют три возможности включения токового стабилизатора.

Не вдаваясь в пояснение возникающих эффектов, рассмотрим результаты SPICE-моделирования поведения трех схем (рис. 13), приведенных на рис. 12. В этих схемах параметры транзисторов и установочных напряжений одинаковы.

В эксперименте в начальный момент времени в каждом из функциональных входов (см. рис. 12) открыт один транзистор с единичной проводимостью. В момент времени 500 нс включается транзистор с проводимостью 20. При этом ток через токовый стабилизатор изменяется на 0,1; 2,4 и 46 мкА для схем на рис. 12, *а*, *б* и *в* соответственно. Из эксперимента ясно, что вариант включения токового стабилизатора на рис. 12, *а*, очевидно, более предпочтителен, чем варианты на рис. 12, *б* и *в*, хотя, безусловно, соответствующим выбором параметров характеристики стабилизации тока для схем на рис. 12, *б* и *в* могут быть улучшены.

Все сказанное выше относится к стабилизации тока в *n*-канальной части β-компаратора. Легко понять, что абсолютно аналогичные построения могут быть проведены и для *p*-канальной части.

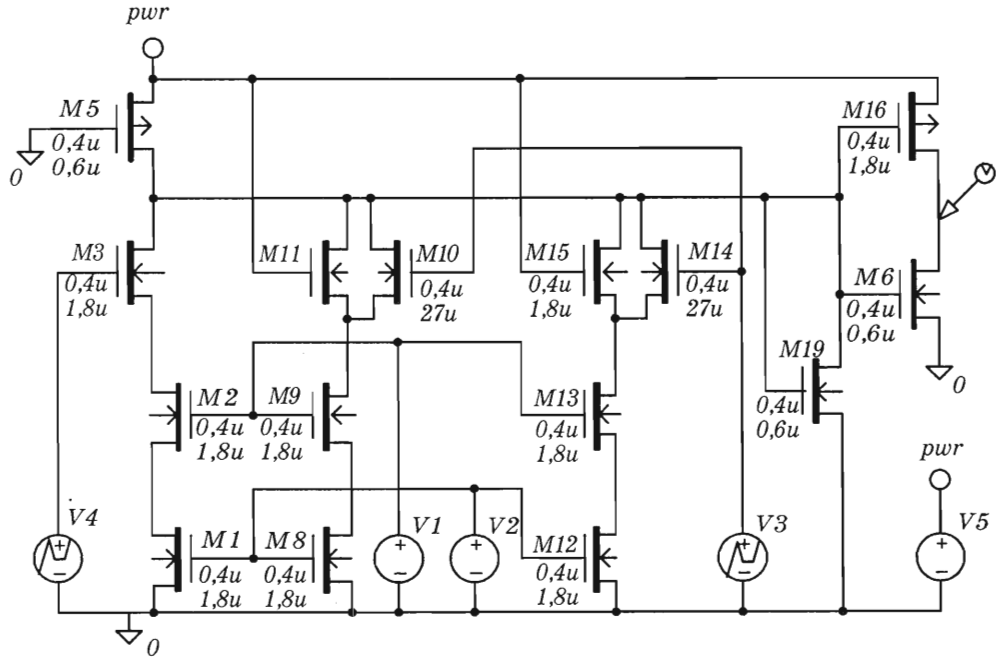
Монте-Карло SPICE-моделирование

Задачей этого раздела является демонстрация реализуемости порогового элемента с функциональными входами. При этом мы не ставим цели детального анализа, оставляя это для следующей работы, посвященной данному предмету.

Рассмотрим пример порогового элемента с функциональными входами. Пусть имеется три функциональных входа

$$z_a = \sum_{j=1}^{k_a} a_j;\ z_b = \sum_{j=1}^{k_b} b_j;\ z_c = \sum_{j=1}^{k_c} c_j; \qquad (27)$$

в пороговом элементе



■ ***Рис. 14.*** *Схема для SPICE-моделирования функции (29)*

$$Y = Sign(z_a + z_b + z_c + 3x - 3) = \overline{Rt\left(\frac{3\bar{x}}{z_a + z_b + z_c + \delta}\right)} =$$

$$= x + z_a z_b z_c = x + \left(\bigvee_{j=1}^{k_a} a_j\right) \cdot \left(\bigvee_{j=1}^{k_b} b_j\right) \cdot \left(\bigvee_{j=1}^{k_c} c_j\right). \qquad (28)$$

Решая вопрос реализуемости, нам достаточно исследовать поведение порогового элемента только на «критических» наборах значений переменных. Под «критическими» мы будем понимать наборы значений переменных, для которых минимальное изменение в наборе приводит к смене значений выхода порогового элемента. Рассмотрим критические наборы для функции (28).

Для обычной пороговой функции «критические» наборы, определяющие реализуемость порогового элемента, есть наборы, принадлежащие так называемым опорным множествам T_0 и F_0; где T_0 – множество минимальных наборов значений переменных, для которых $F(\Omega \in T_0) = 1$, F_0 – множество максимальных наборов значений переменных, для которых $F(\Omega \in F_0) = 0$. Сами наборы определяются минимальными формами функции и ее инверсии [20]. Нетрудно понять, что сказанное справедливо не только для пороговых, но и для монотонных функций.

Для пороговой функции $F(x,z_a,z_b,z_c)$ (28) опорными множествами являются $T_0 = \{1,0,0,0;\ 0,1,1,1\}$ и $F_0 = \{0,1,1,0;\ 0,1,0,1;\ 0,0,1,1\}$. Заметим, что в силу симметрии функции (28) относительно переменных z_i для анализа реализуемости достаточно рассмотреть лишь переход $(0,1,1,1) \underset{Y=1}{\Leftrightarrow} (0,0,1,1)$. Однако при наличии функциональных входов ориентировка на опорные множества недостаточна в силу того, что в зависимости от числа открытых транзисторов вклад переменных z_i в поведение β-компаратора будет различным.

Для оценки наборов значений исходных переменных a_j, b_j, c_j введем номера этих наборов в лексикографическом упорядочивании:

$$I(z_a) = \sum_{j=0}^{k_a-1} a_j 2^j \ ; \ I(z_b) = \sum_{j=0}^{k_b-1} b_j 2^j \ ;$$

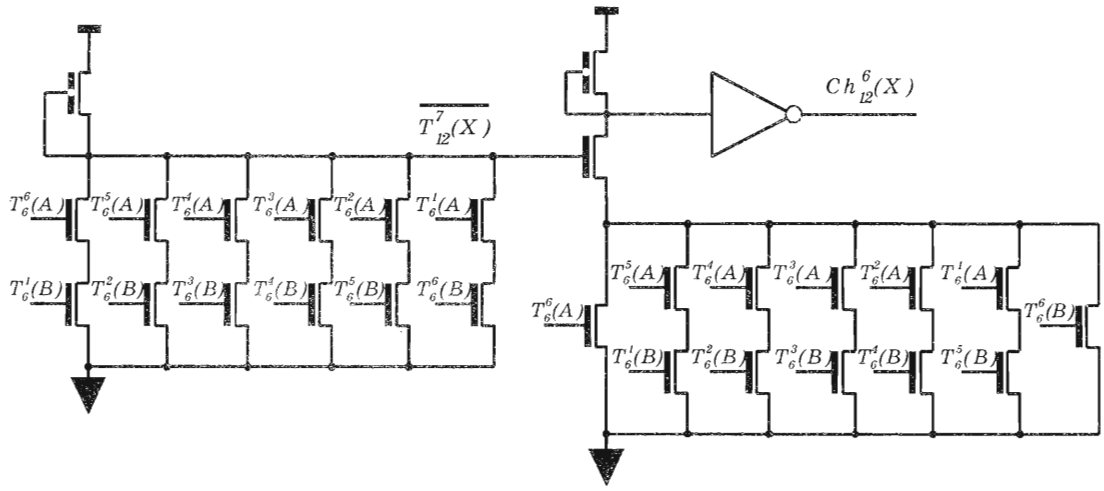
$$I(z_c) = \sum_{j=0}^{k_c-1} c_j 2^j \ . \qquad (29)$$

Полагая $k_a = k_b = k_c = k$, в силу симметрии вклада исходных переменных в соответствующие z_i нам достоточно рассмотреть поведение β-компаратора только на двух наборах[1]

$$(0,1,1,1) \underset{Y=1}{\Leftrightarrow} (0,0,2^k - 1, 2^k - 1).$$

Использованная для моделирования схема приведена на рис. 14, результаты моделирования – на рис. 15.

[1] Использованные здесь векторы соответствуют (x, $I(z_a)$, $I(z_b)$, $I(z_c)$).



■ ***Рис. 16.*** *Реализация выходных каскадов чекера «6 и только 6 из 12»*

$$Ch_{12}^{6}(x_0,\ x_1,\ \ldots,\ x_{11}) = Sign\left(\sum_{j=0}^{11} x_j - 6\right) \times$$

$$\times \overline{Sign\left(\sum_{j=0}^{11} x_j - 7\right)} = T_{12}^{6}(X)\overline{T_{12}^{7}(X)}. \qquad (32)$$

Для снижения требуемого значения порога разделим все множество переменных на два подмножества и введем обозначения

$$A = \{x_0, x_1, x_2, x_3, x_4, x_5\};$$
$$B = \{x_6, x_7, x_8, x_9, x_{10}, x_{11}\}. \qquad (33)$$

Тогда

$$T_{12}^{6}(X) = T_6^6(A) + T_6^5(A)T_6^1(B) +$$
$$+T_6^4(A)T_6^2(B) + T_6^3(A)T_6^3(B) +$$
$$+T_6^2(A)T_6^4(B) + T_6^1(A)T_6^5(B) + T_6^6(B) \qquad (34)$$

и аналогично

$$T_{12}^{7}(X) = T_6^6(A)T_6^1(B) + T_6^5(A)T_6^2(B) +$$
$$+T_6^4(A)T_6^3(B) + T_6^3(A)T_6^4(B) +$$
$$+T_6^2(A)T_6^5(B) + T_6^1(A)T_6^6(B). \qquad (35)$$

Из формул (32)–(35) следует, что для реализации чекера «6 и только 6 из 12» достаточно реализовать двенадцать шестивходовых пороговых элементов, отвечающих указанному выше ограничению на порог, два пороговых элемента с функциональными входами, инверторы и один двухвходовый вентиль «ИЛИ-НЕ».

Обратим, однако, внимание на возможность упрощения реализации этого чекера. На рис. 16 приведена схема, заменяющая пороговые элементы с функциональными входами. Правильнее было бы сказать, что на рис. 16 приведена еще одна возможность реализации порогового элемента с функциональными входами.

Мы не будем в этой работе подробно останавливаться на возникающих на этом пути дополнительных функциональных возможностях. Обратим внимание лишь на то, что такие возможности есть и они позволяют реализовать чекер «6 и только 6 из 12» с расходом оборудования, равным 136 транзисторам.

В заключение заметим, что все изложенное лишь указывает на существующие схемные возможности и реализуемость в кремнии, очевидно, потребует дополнительной работы.

Литература

1. **McCulloch S., Pitts W.** A logical calculus of the ideas imminent in nervous activity // Bulletin of Mathematical Biophysics, – 1943. – 5.– P. 115–133.
2. **Mead C.** Analog VLSI and neural systems. – Addison-Wesley, 1989.
3. **Shibata T., Ohmi T.** Neuron MOS Binary-logic Integrated Circuits: Part 1, Design Fundamentals and Soft-hardware Logic Circuit Implementation // IEEE Trans. Electron Devices. – 1993. – Vol. 40. – N 5. – P. 974–979.
4. **Ohmi T., Shibata T., Kotani K.** Four-terminal device concept for intelligence soft computing on silicon integrated circuits // Proc. of IIZUKA'96, 1996. – P. 49–59.
5. **Fakhraie S. M., Smith K. C.** VLSI-compatible implementations for artificial neural networks. – Kluwer, Boston-Dordrecht-London, – 1997.
6. **Montalvo A., Gyurcsik R., Paulos J.** Toward a general-purpose analog vlsi neural network with on-chip

learning // IEEE Transactions on Neural Networks. – Vol. 8. – N 2. – March 1997. – P. 413–423.
7. **Varshavsky V.** Beta-driven threshold elements, Proceedings of the 8-th Great Lakes Symposium on VLSI // IEEE Computer Society. – Feb. 19–21, 1998. – P. 52–58.
8. **Varshavsky V.** Threshold element and a design method for elements, filed to Japan's Patent Office. – Jan.30, 1998, the application number is JPA H10-54079.
9. **Varshavsky V.** Simple CMOS learnable threshold element, International ICSC/IFAC Symposium on Neural Computation. – Vienna, Austria, Sept. 23–25, 1998.
10. **Varshavsky V.** CMOS artificial neuron on the base of beta-driven threshold element // IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA, Oct. 11–14, 1998. – P. 1857–1861.
11. **Varshavsky V.** Synapse, threshold circuit and neuron circuit, filed to Japan's Patent Office on Aug. 7,1998, the application number is JPA-H10-224994.
12. **Varshavsky V.** Threshold element, filed to Japan's Patent Office on Aug. 12, 1998, the application number is JPA-H10-228398.
13. **Varshavsky V., Marakhovsky V.** Beta-CMOS implementation of artificial neuron, SPIE's 13th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls. Applications and Science of Computational Intelligence II, Orlando, Florida, April 5–8, 1999. – P. 210–221.
14. **Varshavsky V., Marakhovsky V.** Beta-CMOS artificial neuron and implementability limits, Lecture Notes in Computer Science 1607, Engineering Applications of Bio-Inspired Artificial Neural Networks, Jose Mira, Juan V. Sanchez-Andves (Eds.)// Proceedings of International Work-Conference on Artificial and Natural Neural Networks (IWANN'99)}, Spain, June 2–4, Springer, Vol. 11, 1999. – P. 117–128.
15. **Varshavsky V., Marakhovsky V.** The simple neuron CMOS implementation learnable to logical threshold functions // Proceedings of International Workshop on Soft Computing in Industry'99 (IWSCI'99), June 16–18, Muroran, Hokkaido, Japan, IEEE, 1999. – P. 463–468.
16. **Varshavsky V., Marakhovsky V.** Implementability restrictions on the beta-CMOS Artificial Neuron, Proceedings of the 6th International Conference on Electronics, Circuits and Systems (ICECS'99), September 5–8, 1999, Pafos, Cyprus, IEEE, 1999. – P. 401–405.
17. **Varshavsky V., Marakhovsky V.** Learning experiments with CMOS artificial neuron, Lecture Notes in Computer Science 1625, Computational Intelligence Theory and Applications, ed. by Bernard Reusch. Proceedings of the 6th Fuzzy Days International Conference on Computational Intelligence, Dortmund, Germany, May 25–27, Springer, 1999. – P. 706–707.
18. **Varshavsky V., Marakhovsky V.** Non-isotonous beta-driven artificial neuron, Proceedings of SPIE. Applications and Science of Computational Intelligence III, 24–27 April, Orlando, Florida, Vol. 4055. 2000. – P. 250–257.
19. **Weste N. H. E., Eshraghian K.** Principles of CMOS VLSI Design. A System Perspective, 2nd ed. – Addison-Wesley Reading, MA, 1993.
20. **Dertouzos M. L.** Threshold logic: A synthesis approach – The MIT Press, Cambridge, MA, 1965.
21. **Verhoeff T.** Delay-insensitive codes - an overview // Distributed Comput., 3, 1988. 1–8.

УДК 681.3.06

РЕАЛИЗАЦИЯ АНИМАЦИИ ПРИ ПОСТРОЕНИИ ВИЗУАЛИЗАТОРОВ АЛГОРИТМОВ НА ОСНОВЕ АВТОМАТНОГО ПОДХОДА

М. А. Казаков,
аспирант
А. А. Шалыто,
доктор техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

В статье предлагается подход к реализации анимации алгоритмов, который расширяет технологию построения визуализаторов алгоритмов на основе автоматного подхода.

Article describes proposed approach for algorithm animation extension above existing technology for algorithm visualizations implementation based on the automaton programming approach.

Введение

Визуализаторы алгоритмов широко используются в процессе преподавания дискретной математики и теории классических вычислительных алгоритмов [1 – 3].

Визуализатор – программа, иллюстрирующая выполнение алгоритма при определенных исходных данных [4].

В работе [5] рассматривался метод построения визуализаторов на основе автомата Мили, а также предлагалась технология формального преобразования вычислительного алгоритма в визуализатор. Аналогичным образом логика визуализаторов может строиться на основе автомата Мура.

Ранее визуализатор рассматривался как дискретная последовательность статических иллюстраций, что в большинстве случаев достаточно для обучения. Однако в некоторых алгоритмах статических иллюстраций недостаточно. Примером такого алгоритма может служить обход дерева [6], так как именно процесс движения указателя по вершинам дерева является наиболее интересным и важным для понимания этого алгоритма.

В настоящей работе предлагается расширение технологии построения визуализаторов с целью включения в нее возможности анимации требуемых шагов визуализации. При этом рассматривается такая разновидность анимации, при которой изображается переход от предыдущих значений визуализируемых переменных к последующим. Поскольку каждая статическая иллюстрация является функцией от визуализируемых переменных, то указанная анимация и будет визуализировать шаг алгоритма в динамике. Предлагаемый подход иллюстрируется на примере традиционной реализации алгоритма обхода двоичного дерева [7].

Автоматная технология построения визуализаторов без анимации

В работе [5] описана автоматная технология построения визуализаторов алгоритмов, состоящая из следующих этапов.

1. *Постановка задачи, которую решает визуализируемый алгоритм.*
2. *Решение задачи в словесно-математической форме.*
3. *Выбор визуализируемых переменных.*
4. *Анализ алгоритма для визуализации.* Анализируется решение с целью определения того, что и как отображать на экране.
5. *Алгоритм решения задачи.*
6. *Реализация алгоритма на выбранном языке программирования.* На этом шаге производится реализация алгоритма, его отладка и проверка работоспособности.
7. *Построение схемы алгоритма по программе.*

8. *Преобразование схемы алгоритма в граф переходов автомата, реализующего алгоритм*, который может быть как автоматом Мили, так и автоматом Мура [8, 9].

9. *Преобразование автомата реализующего алгоритм в автомат визуализации.*

10. *Выбор интерфейса визуализатора.*

11. *Сопоставление иллюстраций и комментариев с состояниями автомата* (иллюстрации формируются компонентом программы, называемым «рисовальщик»).

12. *Архитектура программы визуализатора.*

13. *Программная реализация визуализатора.*

С целью обеспечения возможности анимации в последовательность действий вводятся дополнительные шаги. Отметим следующее: поскольку изменения визуализируемых переменных в используемых в настоящей работе автоматах Мура производятся в состояниях, что весьма удобно, то будем рассматривать вопрос о введении анимации в визуализаторы, построенные на основе автоматов Мура.

Анимация в визуализаторах на основе автоматов Мура

Для построения визуализатора с анимацией предлагается применять четыре типа автоматов:

1) *автомат, реализующий алгоритм* (формируется на восьмом этапе);

2) *автомат визуализации* (формируется на девятом этапе);

3) *преобразованный автомат визуализации*;

4) *автомат анимации.*

Автомат визуализации отображает последовательно *статические иллюстрации* в каждом состоянии, что приводит к *статической (пошаговой) визуализации.* Здесь под статической иллюстрацией понимается фиксированный кадр, не изменяющийся с течением времени.

Преобразованный автомат визуализации кроме статических иллюстраций отображает также и *динамические иллюстрации* в дополнительно вводимых анимационных состояниях. Это позволяет говорить о *динамической визуализации* (*анимации*).

Динамическая иллюстрация состоит из набора промежуточных статических иллюстраций, отображаемых на экране автоматом анимации через определенные промежутки времени. Это приводит к динамическому изменению иллюстрации в анимационном состоянии.

Для автоматов Мура анимацию (также как и формирование статических изображений и комментариев) естественно проводить в состояниях. Для введения анимации в визуализатор расширим технологию за счет введения следующих шагов:

a) *выбор состояний автомата визуализации, в которых выполняется анимация* (такие состояния будем называть анимационными);

b) *построение преобразованного автомата визуализации путем замены каждого из анимационных состояний тремя состояниями*, в первом из которых производятся преобразования данных, во втором – анимация, соответствующая этому состоянию, при переходе в третье состояние анимация заканчивается, а непосредственно в третьем состоянии отображается статическая иллюстрация, соответствующая состоянию исходного автомата;

c) *разработка анимационного автомата* (выполняется один раз для всех визуализаторов, проектируемых по этой технологии);

d) *обеспечение взаимодействия между преобразованным автоматом визуализации и анимационным автоматом;*

e) *разработка и реализация функции вывода каждой динамической иллюстрации, зависящей от состояния автомата визуализации и шага анимации.*

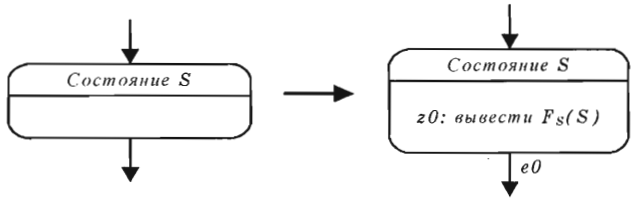
До перехода к более подробному рассмотрению новых этапов изложим, в чем состоит девятый этап исходной технологии (рис. 1).

На этом этапе вводится выходное воздействие $z0$, осуществляющее формирование статического изображения и комментария в рассматриваемом состоянии. Различия в изображениях и комментариях формализуются с помощью функции $F_S(S)$, параметром которой является номер состояния S. В каждом состоянии, формирующем выходное воздействие $z0$, переход к следующему состоянию осуществляется по событию $e0$ (нажатие клавиши «шаг» в интерфейсе визуализатора).

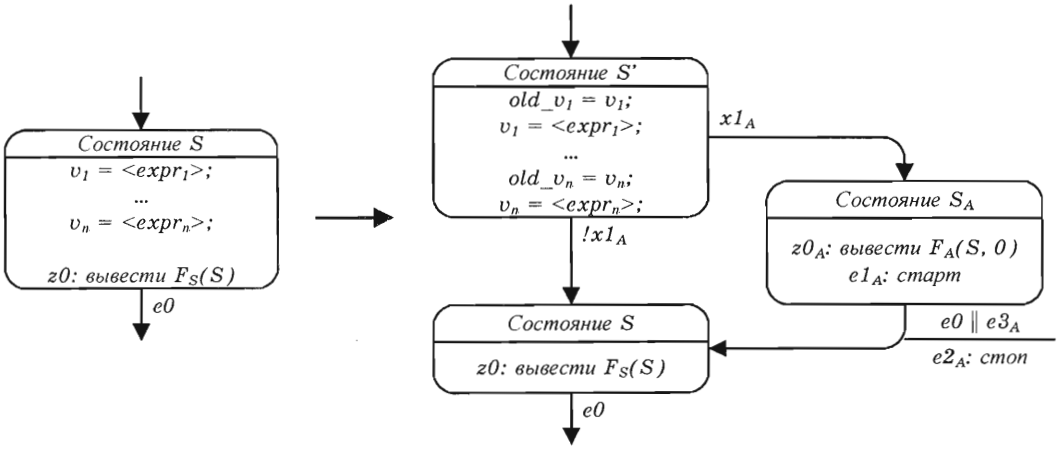
Поясним каждый из вновь вводимых этапов $a - e$.

На этапе a выполняется выбор анимационных состояний в соответствии с особенностями алгоритма и принятыми решениями по анимации на четвертом этапе исходной технологии (анализ алгоритма для визуализации). Так как в алгоритме обхода двоичного дерева наибольший интерес представляет процесс перехода между вершинами, то состояния, в которых происходит переход, и будут выбраны в качестве анимационных.

На этапе b в автомате визуализации для получения преобразованного автомата этого типа



■ ***Рис. 1.*** *Формирование состояний автомата визуализатора*



■ *Рис. 2. Преобразование анимационного состояния*

каждое анимационное состояние формально заменяется тремя состояниями (рис. 2).

Опишем это преобразование. Предположим, что состояние S выбрано для визуализации. Предположим также, что в этом состоянии изменяются значения визуализируемых переменных $v_1, ..., v_n$. При этом переменной v_i присваивается значение выражения $< expr_i >$. Преобразование состоит в замене выбранного состояния на три состояния – S', S и S_A.

В состоянии S' значения переменных $v_1, ..., v_n$ предварительно сохраняются в переменных $old\_v_1, ..., old\_v_n$, а также проводятся все действия, выполняемые в состоянии S автомата визуализации.

В состоянии S' преобразованного автомата визуализации в отличие от состояния S исходного автомата этого типа не происходит ожидания события, а выполняется переход в одно из состояний – S_A или S.

Переход в состояние S_A производится, если переменная $x1_A$ принимает значение `true` (анимация включена). Анимация осуществляется, пока преобразованный автомат визуализации находится в состоянии S_A. При входе в это состояние осуществляется запуск автомата анимации при помощи события $e1_A$: *старт* и отображается иллюстрация $F_A(S, 0)$, соответствующая началу анимации.

Выход из состояния анимации и переход в состояние S происходит, как по событию $e0$ (нажатие клавиши), так и по событию $e3_A$ (окончание анимации), и сопровождается сигналом $e2_A$: *стоп*. Таким образом, обеспечивается как автоматическое, так и ручное завершение анимации.

Состояние S является завершающим. В этом состоянии осуществляется отображение статической иллюстрации $F_S(S)$. Отметим, что поскольку в преобразованном анимационном автомате присутствуют действия на дугах, то этот автомат является смешанным автоматом.

Также следует отметить, что при выключении анимации ($x1_A$ отсутствует) преобразованный автомат визуализации становится изоморфным исходному автомату визуализации. Поэтому включение анимации может рассматриваться именно как *расширение* исходного визуализатора, а не его преобразование.

На этапе c строится автомат анимации (рис. 3). Он является смешанным автоматом, поскольку содержит действия как на дугах, так и в вершинах графа перехода.

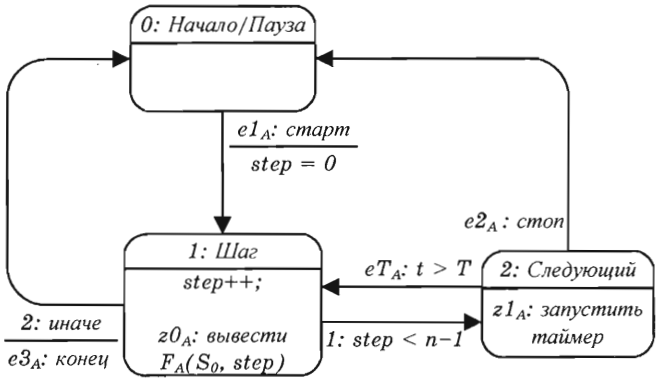
Из рассмотрения рис. 3 следует, что автомат предназначен для изменения значения переменной *step*, которая хранит номер шага анимации.

Автомат анимации формирует следующие выходные воздействия и события:

- $z1_A$: *запустить таймер* – запускает таймер, который через период времени T генерирует событие eT_A;
- $z0_A$: *вывести* $F_A(S_0, step)$ – информирует «рисовальщик» о необходимости перерисовать иллюстрацию;
- $e3_A$: *конец анимации* – сигнализирует о том, что анимация закончилась.

Анимационный автомат реагирует на следующие события:

- $e1_A$: *старт* – автомат визуализации перешел в состояние, в котором должна начаться анимация;



■ *Рис. 3. Автомат анимации*

• $e2_A$: *стоп* – автомат визуализации перешел в состояние, в котором анимация должна закончиться;

• eT_A – событие от таймера, по которому осуществляется переход к следующему шагу анимации.

Описанный автомат является универсальным и может быть использован в любом визуализаторе.

На этапе *d* автомат визуализации и автомат анимации связываются посредством событий и выходных воздействий, как показано на рис. 4.

На этапе *e* стандартная функциональность «рисовальщика» сопоставления номера состояния с текстовым комментарием и статической иллюстрацией расширяется. В режиме анимации «рисовальщик» принимает на вход значение номера шага. Другими словами, если функция F_S реализует статические иллюстрации, а функция F_A – динамические (анимационные) иллюстрации, то при переходе от статики к анимации осуществляется преобразование вида

$$F(state) \to \begin{cases} F_S(state); \\ F_A(state, step). \end{cases}$$

При этом справедливы следующие соотношения:

$$F_S(state) = F(state); \quad (1)$$

$$F_A(state, 0) = F_S(state - 1); \quad (2)$$

$$F_A(state, N) = F_S(state). \quad (3)$$

Соотношение (1) отражает тот факт, что преобразованный автомат визуализации формирует статические иллюстрации, совпадающие с иллюстрациями исходного автомата. Соотношения (2), (3) показывают, что формирование динамических (анимационных) иллюстраций в состоянии *state* происходит таким образом, что начальная анимационная иллюстрация совпадает со статической иллюстрацией в предыдущем состоянии автомата, а конечная – со статической иллюстрацией в состоянии *state*.

Из изложенного следует, что рисовальщик в визуализаторе в общем случае будет реализовываться посредством не одного, а двух операторов *switch*. При этом первый оператор *switch* будет соответствовать статическим иллюстрациям, а второй – динамическим.

Таким образом, первые четыре перечисленных шага вводятся в указанную выше технологию между девятым и десятым этапами, а пятый шаг – между одиннадцатым и двенадцатым этапами. При этом этапы технологии с первого по девятый не изменяются, а дальнейшие шаги преобразуются следующим образом.

10. *Выбор состояний, в которых будет выполняться анимация.*

11. *Замена каждого из выбранных состояний тремя состояниями.*

12. *Использование автомата анимации.*

13. *Обеспечение взаимодействия между преобразованным и анимационным автоматами.*

14. *Выбор интерфейса визуализатора.*

15. *Сопоставление иллюстраций и комментариев с состояниями автомата.*

16. *Обеспечение выбора в каждом анимационном состоянии статического либо динамического изображения*, зависящего не только от состояния основного автомата, но и от номера шага анимации.

17. *Архитектура программы визуализатора.*

18. *Программная реализация визуализатора.*

Построение визуализатора для алгоритма обхода двоичного дерева

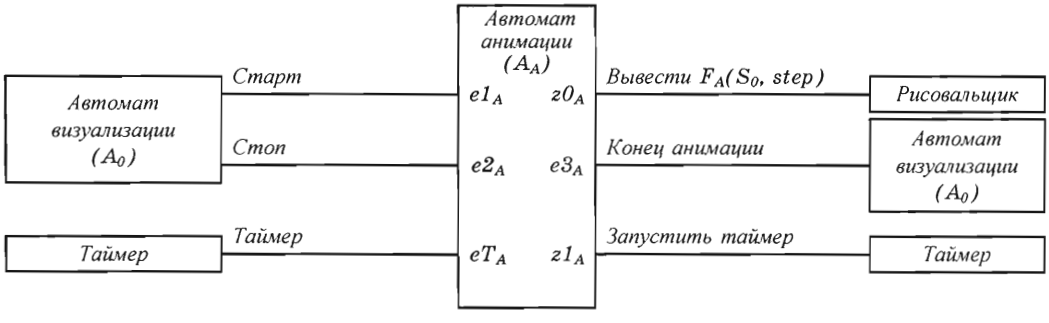
Продемонстрируем технологию на примере построения визуализатора обхода двоичного дерева.

Постановка задачи обхода двоичного дерева. Задано двоичное дерево, состоящее из *n* вершин, каждая из которых имеет ссылки на правого и левого потомка. Требуется реализовать алгоритм, который выводит по одному разу все вершины этого дерева [7].

Решение задачи. Приведем словесную формулировку решения этой задачи.

Для описания вершины дерева введем следующую структуру данных:

```
public class Node {
   private int left;  // левое поддерево
   private int right; // правое поддерево
```



■ *Рис. 4. Схема связей автомата анимации*

```
    private int index; // индекс в массиве
    ...
}
```

Введем массив `visited[0..n-1]`, в котором будем отмечать пройденные вершины.

Для решения этой задачи предлагается создать вспомогательный стек вершин (*stack*). Далее, двигаясь от корня к листьям, предлагается совершить следующее действие в цикле:

- если в текущей вершине (*node*) левое поддерево не пройдено (`node.left > 0 && !visited[node.left]`), то записать ее в стек и двигаться к левому потомку;
- иначе, если правое поддерево не пройдено (`node.right > 0 && !visited[node.right]`), то записать текущую вершину в стек и двигаться к правому потомку;
- иначе, если в стеке находится хотя бы одно значение, то «снять» с вершины стека предыдущую вершину и двигаться к ней;
- иначе завершить алгоритм.

Выбор визуализируемых переменных. Предлагается визуализировать следующие переменные:

- `stack` – стек вершин от корня до текущей вершины;
- `visited` – массив флагов для идентификации пройденных вершин;
- `result` – результирующий массив;
- `node` – текущая вершина.

Выбор алгоритма визуализации. Предлагается визуализировать следующие особенности алгоритма для обеспечения возможности наилучшего разъяснения его действия:

- процесс занесения вершины в стек;
- текущая вершина;
- процесс перехода из предыдущей в следующую вершину;
- результат выполнения алгоритма;
- пройденные вершины (отмечать цветом).

Алгоритм решения задачи. Приведем алгоритм решения задачи на псевдокоде [10]:

```
1  node Я nodes[0]
2  добавить node в результат
3  while true
4    do
5       отметить node.left как пройденную
6       if node.left ≥ 0 and node.left не пройдена
7            then
8                 добавить node в стек
9                 node ← node.left
10                добавить node в результат
11      else if node.right ≥ 0 and node.right не пройдена
12           then
13                добавить node в стек
14                node ← node.left
15                добавить node в результат
16      else if стек не пуст
17           then node ← с вершины стека
18      else break
19 return результат
```

Реализация алгоритма на языке *Java*. Перепишем программу, записанную на псевдокоде, с помощью языка *Java*:

```
/**
 * @param nodes массив вершин дерева
 * @return список всех вершин, выведенных в порядке обхода
 */
private static List traverse(Node[] nodes) {
    // Содержит результат
    List result = new ArrayList();
    // Стек для обеспечения возврата
    LinkedList stack = new LinkedList();
    // Хранят флаги того, что вершина пройдена
    boolean[] visited = new boolean[nodes.length];
    // Заполняем исходными значениями
    Arrays.fill(visited, false);
    // Выбираем корневую вершину
    Node node = nodes[0];
    result.add(node);
    while (true) {
        // Отмечаем текущую вершину, как пройденную
        visited[node.index()] = true;
        if (node.left() >= 0 && !visited[node.left()]) {
            // Переход к левому поддереву
            stack.addLast(node);
            node = nodes[node.left()];
            result.add(node);
        } else if (node.right() >= 0 && !visited[node.right()]) {
            // Переход к правому поддереву
            stack.addLast(node);
            node = nodes[node.right()];
            result.add(node);
        } else {
            // Возврат
            if (stack.isEmpty()) {
                break;
            }
            node = (Node)stack.removeLast();
        }
    }
    return result;
}
```

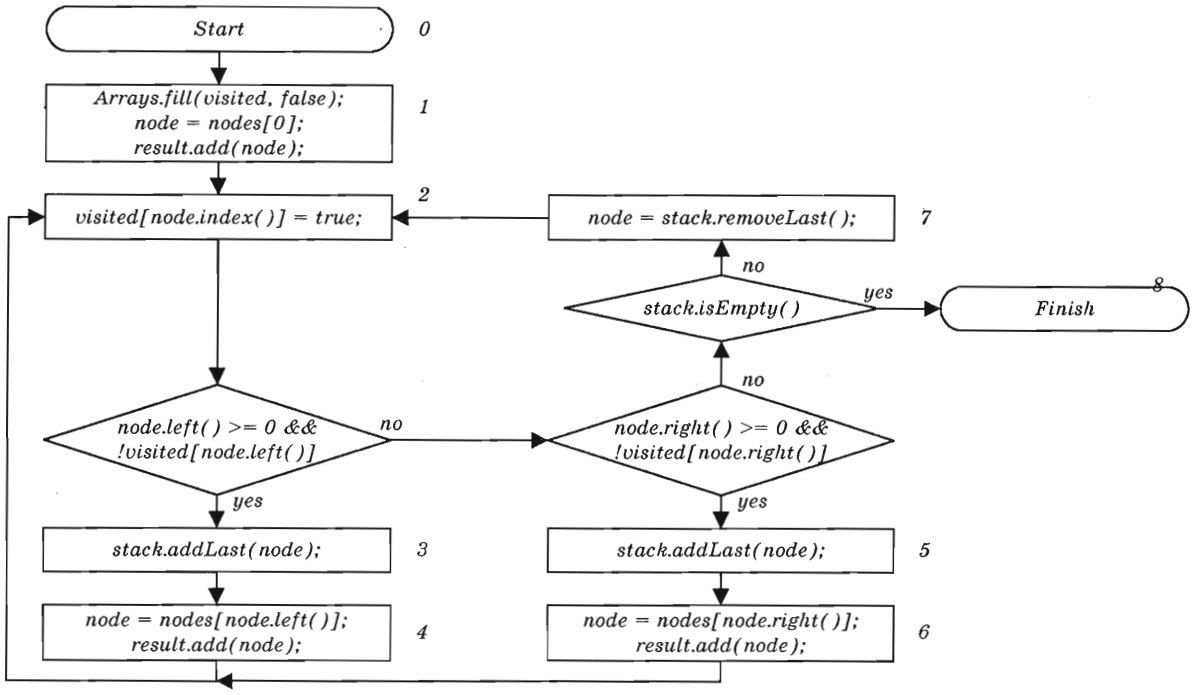
В этой программе операции ввода/вывода не приведены, так как их будет выполнять визуализатор.

Построение схемы алгоритма по программе. Построим по тексту программы схему алгоритма (рис. 5).

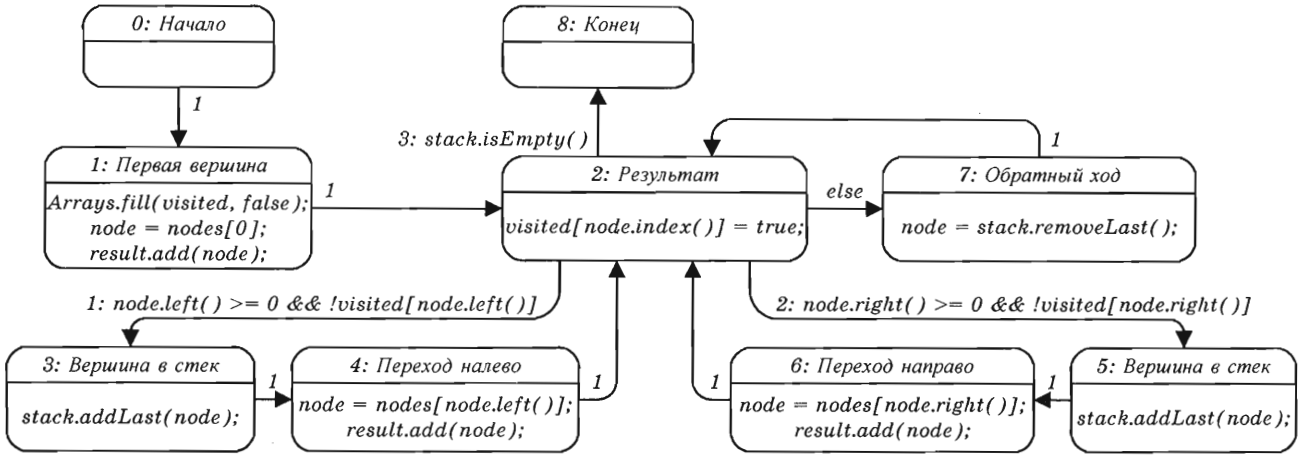
Преобразование схемы алгоритма в автомат Мура. Следуя методу, приведенному в работе [8], построим автомат Мура, соответствующий приведенной выше схеме алгоритма (рис. 6).

Исходя из схемы алгоритма на рис. 4, выделим восемь состояний. В состоянии 1 осуществляется инициализация алгоритма. В состоянии 2 текущая вершина отмечается как пройденная. В состояниях 3 и 5 текущая вершина помещается в стек, а в состояниях 4 и 6 осуществляется переход. Состояние 7 отвечает за обратный ход. Состояние 8 является конечным.

Преобразование автомата, реализующего алгоритм в автомат визуализации. В соответствии с методом, схема которого представлена на рис. 1,



■ *Рис. 5. Схема алгоритма двоичного обхода дерева*



■ *Рис. 6. Автомат Мура, реализующий обход двоичного дерева*

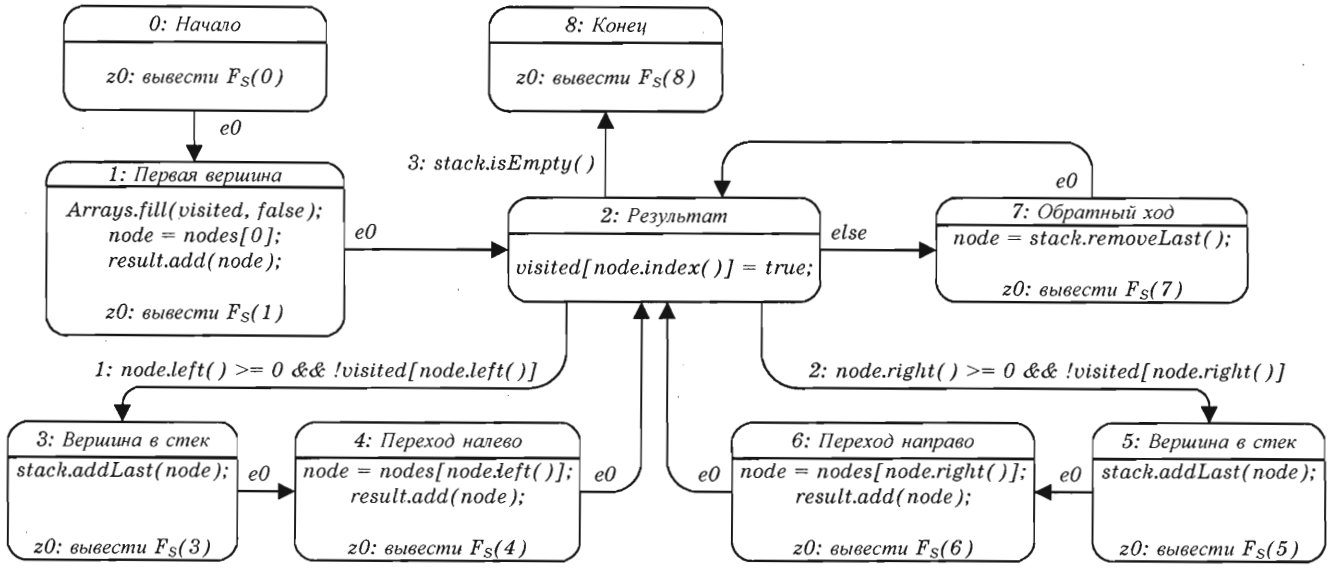
автомат, изображенный на рис. 6, преобразуется в автомат визуализации (рис. 7).

Выбор состояний, в которых будет выполняться анимация. В алгоритме обхода дерева наиболее важным для отображения является процесс передвижения по вершинам, поэтому анимационными являются состояния, в которых изменяется номер текущей вершины. Такими состояниями являются состояния с номерами 4, 6, 7.

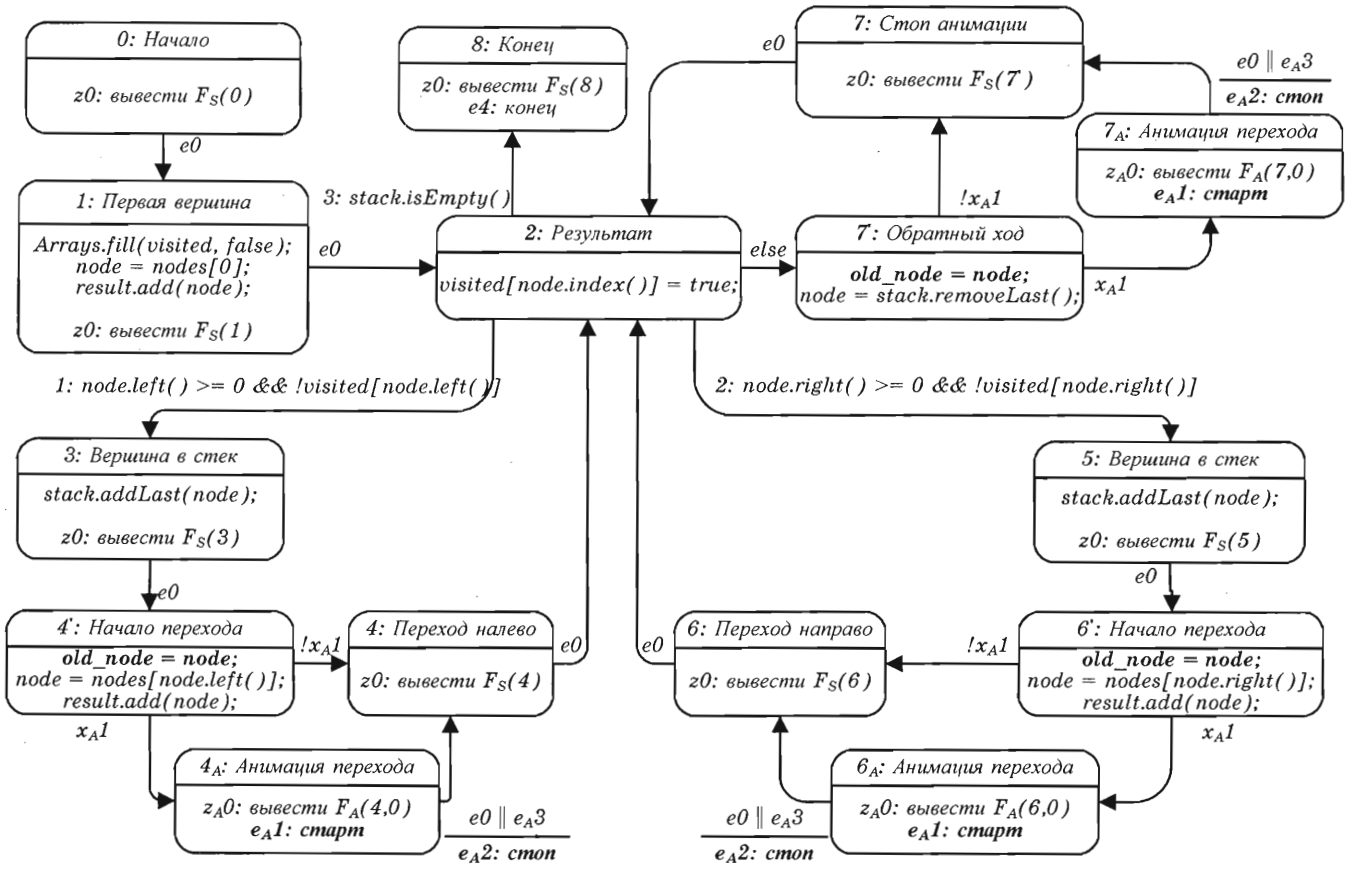
Замена каждого из анимационных состояний тремя состояниями. В соответствии с методом, изложенным выше (см. рис. 2), заменим состояния 4, 6, 7 тройками состояний, в первом из которых запоминается номер предыдущей вершины, во втором формируется событие начала анимации, а в третьем завершается шаг визуализатора и создается конечная иллюстрация в состоянии (рис. 8). При этом событие окончания анимации формируется на переходе между анимационным и конечным состояниями шага визуализации.

Использование автомата анимации. Как отмечалось выше, для анимации используется стандартный автомат, приведенный на рис. 3.

Обеспечение взаимодействия между преобразованным и анимационным автоматами. Для обеспечения взаимодействия между автоматом визуализации и автоматом анимации вводятся связи, представленные на рис. 4.



■ ***Рис. 7.*** *Автомат визуализации*

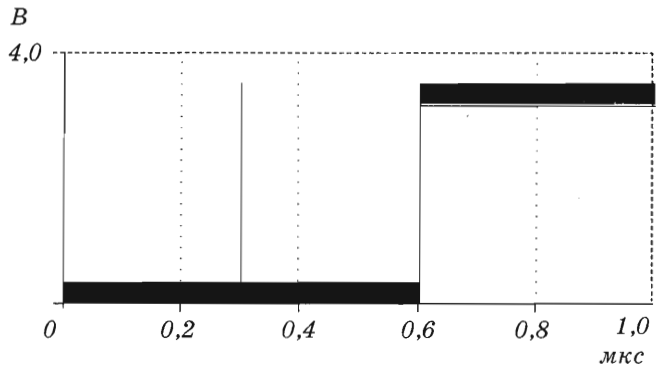


■ ***Рис. 8.*** *Преобразованный автомат визуализации*

Выбор интерфейса визуализатора. В верхней части визуализатора (рис. 9) приводится следующая информация:

• дерево, изображенное графически. Серым цветом отмечены пройденные вершины, белым цветом – непройденные. Текущая вершина отмечена темно-серым цветом;

• в правой части иллюстрации изображен стек, а в нижней части – порядок обхода дерева (массив пройденных вершин).



■ ***Рис. 15.*** *Результаты Монте-Карло SPICE-моделирования для устройства на рис. 14 (200 прогонов)*

В схеме на рис. 14 транзистор М3 соответствует переменной $a_0 = z_a$ с единичным весом. Пара транзисторов М10, М11 (М14, М15) соответствует переменным $z_b(z_c)$, причем открытый транзистор М11 (М15) моделирует набор $I(z_b) = 1(I(z_c) = 1)$, а открытая пара транзисторов М10, М11 (М14, М15) моделирует $I(z_b) = 15$ $(I(z_c) = 15)$. В эксперименте была реализована следующая последовательность состояний $[x, I(z_a), I(z_b), I(z_c)]$:

$$[0,0,15,15] \underset{\substack{t=0\div300\text{ нс}\\ Y=0}}{\Rightarrow} [0,0,1,1\} \underset{\substack{t=300\div600\text{ нс}\\ Y=0}}{\Rightarrow} [0,1,1,1] \underset{\substack{t=600\div1000\text{ нс}\\ Y=1}}{\quad} . \quad (30)$$

На рис. 15 приведены результаты 200 прогонов Монте-Карло SPICE-моделирования схемы на рис. 14 для 10 %-го разброса параметров транзисторов:

– TOX (тонкий окисел) Dev 2,5 %, Lot 2,5 %,[1]

– VTO (порог) Dev 2,5 %, Lot 2,5 %.

Из рис. 15 следует, что реализуемость рассмотренного устройства вполне удовлетворительна[2]. Как показано в работе [10], для 10 %-го разброса параметров устойчивая работа β-управляемого порогового элемента гарантирована при значениях порога $\eta \le \min(3, \sum_j \omega_j - 3)$. Это условие, как видно из результатов моделирования, сохраняется и для пороговых элементов с функциональными входами.

Результаты эксперимента также позволяют надеяться, что аналогично обычному β-управляемому пороговому элементу использование процедуры обучения для формирования V_{ref1} [13–16] позволит поднять критическое значение порога с 3 до более чем 100. Это предположение основывается на том, что при использовании процедуры обучения все технологические вариации параметров компенсируются в процессе обучения и на критическую величину порога начинают влиять только вариации эксплуатационных характеристик (температура, напряжение источника питания и т. д.) и точность стабилизации тока в функциональном входе.

[1] Dev – разброс внутри одного lot, Lot – разброс между lots.

[2] Всплеск в момент времени $t = 300$ нс – помеха, связанная с одновременным переключением транзисторов М10 и М14, что эквивалентно одновременному изменению значений 30 переменных.

Заключение

Приведенное выше ограничение на значение порога может вызвать разочарование в перспективности использования β-управляемого порогового элемента как такового и β-управляемого порогового элемента с функциональными входами в частности. Природу невозможно обмануть, и асимптотические оценки сложности логических схем являются их очевидным свойством. С другой стороны, на практике мы имеем дело не со схемами вообще, а с практически необходимыми схемами, которые, именно в силу их практической ориентации, обладают, как правило, некоторой внутренней организацией. Если эта организация толерантна пороговой логике, то можно надеяться на эффективность применения пороговых элементов. Для подтверждения сказанного обратимся к примеру.

Во многих применениях весьма эффективным оказывается использование равновесных кодов «k из n» или «k и только k из n» – это и асинхронная передача данных (самосинхронные коды) [21–22], и задачи тестирования (равновесное кодирование) [22], и многое другое. В этих задачах необходимо синтезировать индикаторы (чекеры), распознающие такие коды.

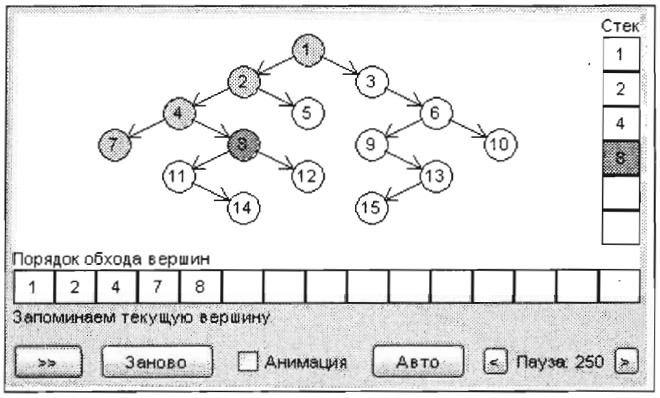
Пусть нам необходимо реализовать чекер кода «6 и только 6 из 12». Функция, реализуемая чекером, при этом определяется, как

$$Ch_{12}^{6}(x_0, x_1, \ldots, x_{11}) = \begin{cases} 1, \text{ если } \sum_{j=0}^{11} x_j = 6 \\ 0, \text{ если } \sum_{j=0}^{11} x_j \ne 6 \end{cases} . \quad (31)$$

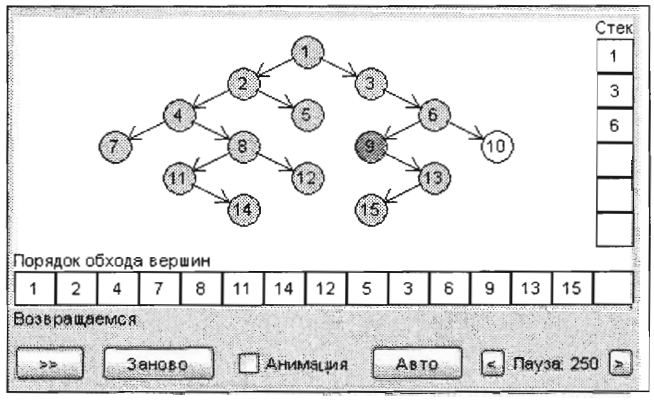
Минимальная дизъюнктивная форма этой функции содержит 924 терма ранга 12. Использование декомпозиции снижает сложность реализации этой функции в булевом базисе, однако это снижение не принципиально и сохраняет порядок сложности.

Рассмотрим реализацию чекера (31) в пороговом базисе с функциональными входами при ограничении $\min(\eta, \sum_j \omega_j - \eta) \le 3$.

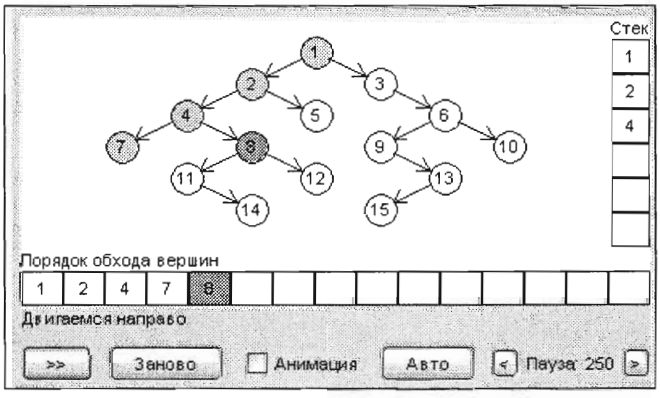
Функция чекера (31) немонотонна, однако она достаточно просто выражается через монотонные (пороговые):



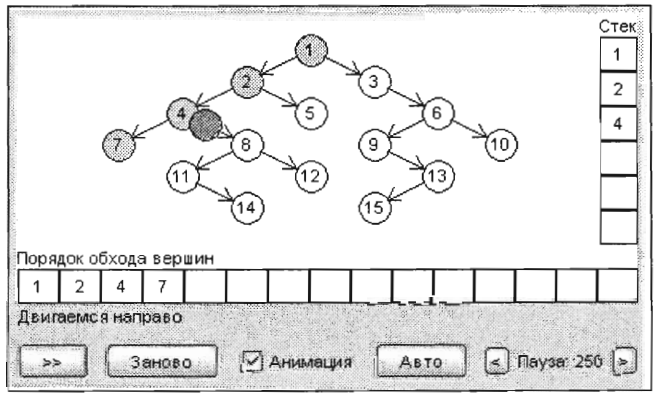
■ *Рис. 9. Иллюстрация в состояниях 3 и 5*



■ *Рис. 11. Иллюстрация в состоянии 7*



■ *Рис. 10. Интерфейс визуализатора*



■ *Рис. 12. Динамическая иллюстрация*

Под массивом пройденных вершин динамически отображаются текстовые комментарии, соответствующие текущим состояниям автомата визуализации. Например, на рис. 10 изображен комментарий «Двигаемся направо», соответствующий состоянию 6. Под комментарием расположены элементы управления. Следует отметить, что по сравнению с обычным визуализатором появляется флаг управления анимацией:

«>>» – сделать шаг алгоритма;

«Заново» – начать алгоритм сначала;

«Анимация» – отображать или не отображать анимацию;

«Авто» – перейти в автоматический режим;

«<», «>» – изменить паузу между автоматическими шагами алгоритма.

Сопоставление иллюстраций и комментариев с состояниями автомата. Для наглядности визуализации алгоритма предлагается акцентировать внимание на следующих элементах.

В состояниях 4 и 6 (см. рис. 10) статическая иллюстрация отображает: текущую вершину; пройденные вершины; последнюю вершину в результирующем массиве.

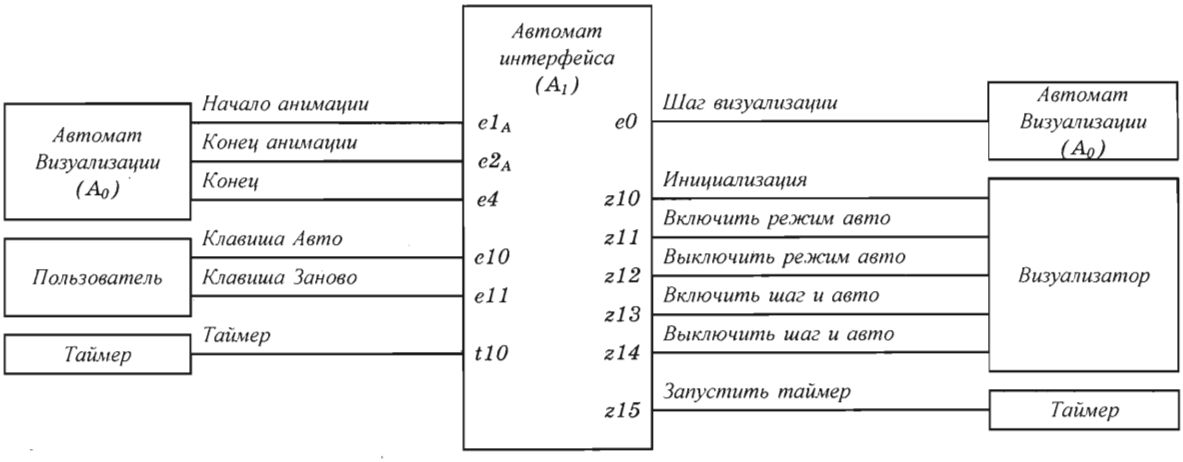
В состояниях 3 и 5 (см. рис. 9) отмечается вершина, добавленная в стек.

В состоянии 7 (рис. 11) отображается обратный ход. При этом отмечается только текущая вершина, поскольку данные в стек и массив пройденных вершин не добавляются.

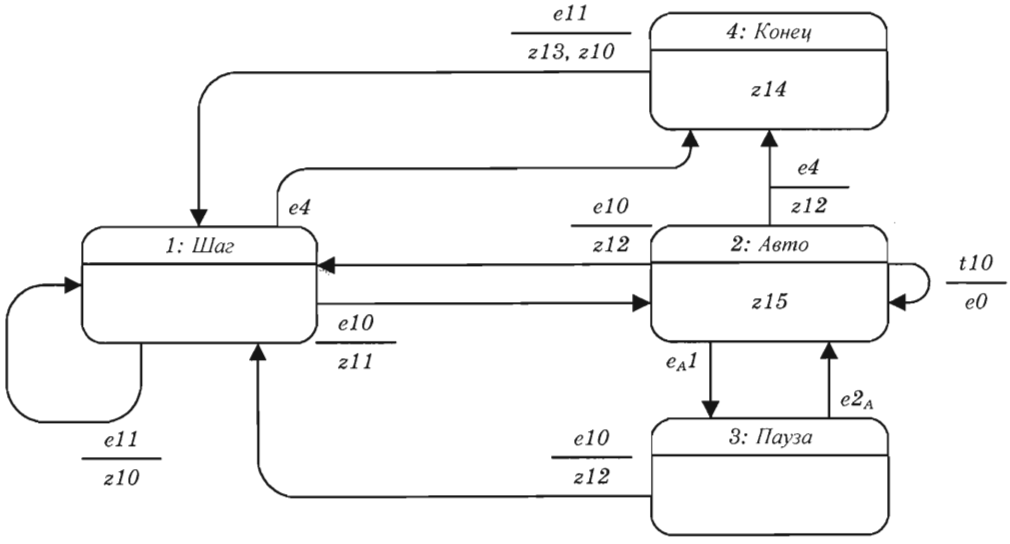
Обеспечение выбора в каждом анимационном состоянии статического либо динамического изображения. Для отображения динамических иллюстраций используются статические иллюстрации с наложенным изображением движущегося указателя (рис. 12).

Архитектура программы визуализатора. Для реализации пользовательского интерфейса сформирован еще один автомат, реализующий поведение интерфейса визуализатора. Схема взаимодействия этого автомата приведена на рис. 13.

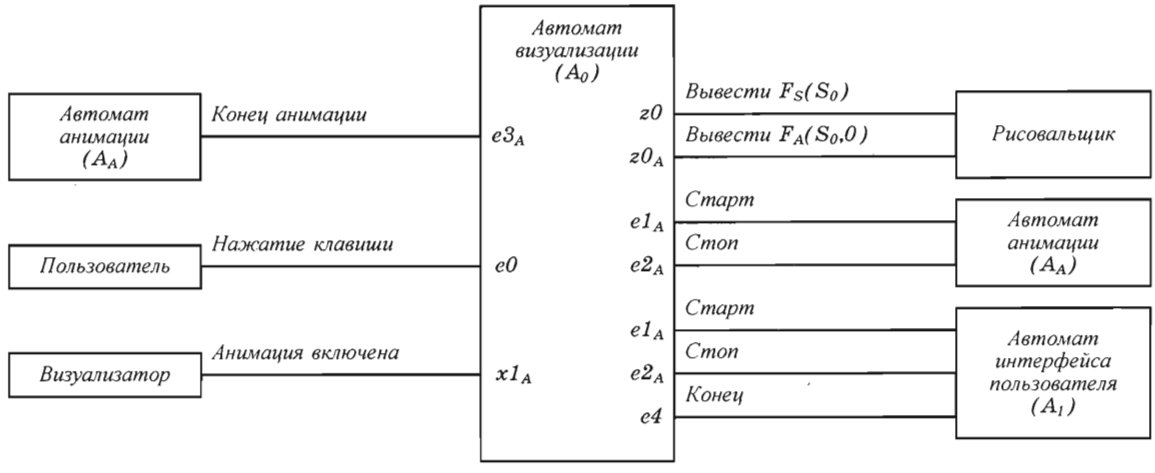
Как следует из этого рисунка, автомат взаимодействует с автоматом визуализации и управляет интерфейсом визуализатора. Таймер используется для реализации функции *Авто*. Диаграмма переходов интерфейсного автомата приведена на рис. 14.



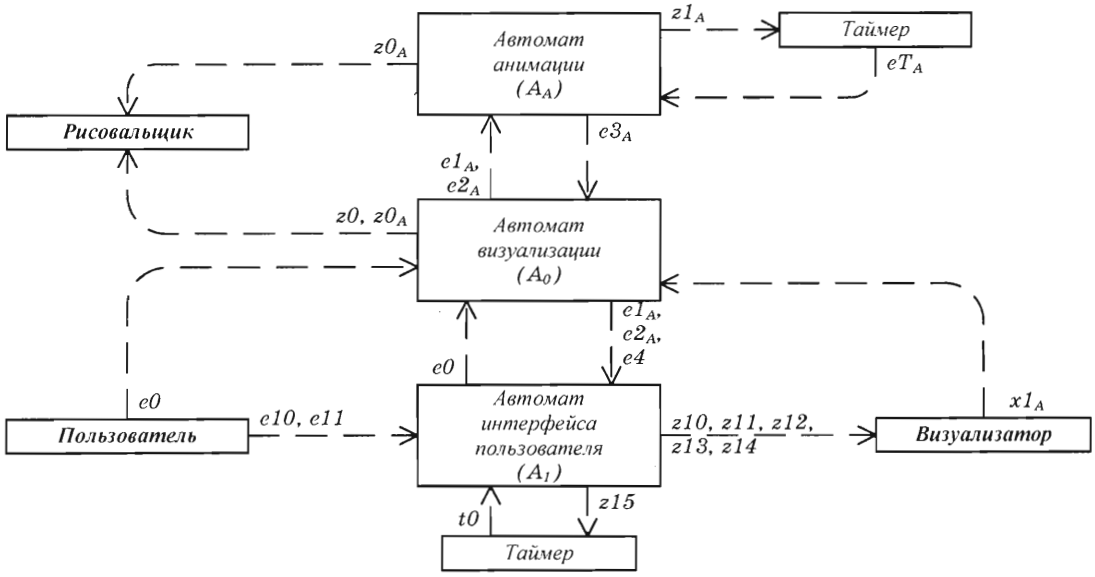
■ ***Рис. 13.*** *Схема взаимодействия автомата интерфейса визуализатора*



■ ***Рис. 14.*** *Диаграмма переходов автомата интерфейса визуализатора*



■ ***Рис. 15.*** *Схема взаимодействия преобразованного автомата визуализации*



■ ***Рис. 16.*** *Схема взаимодействия компонент визуализатора*

Схема взаимодействий преобразованного автомата визуализации изображена на рис. 15.

Схема взаимодействия всех компонент визуализатора представлена на рис. 16.

Заключение

В статье предложен подход к реализации визуализаторов алгоритмов с анимацией, улучшающий восприятие алгоритмов обучающимися, что особенно важно при дистанционном обучении. Подход расширяет технологию построения визуализаторов алгоритмов на основе автоматного подхода, предложенную в работе [5].

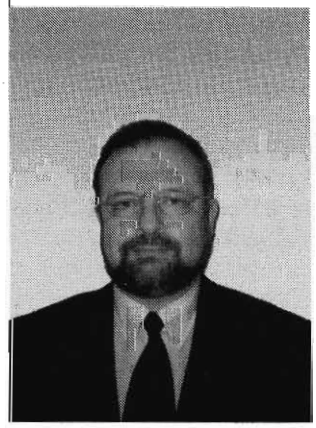
Литература

1. **Интернет-школа** программирования. http://ips.ifmo.ru
2. **Казаков М. А., Мельничук О. П., Парфенов В. Г.** Интернет школа программирования в СПбГИТМО (ТУ). Реализация и внедрение // Материалы Всероссийск. научно-методич. конф. «Телематика'2002». – СПб., 2002. – С. 308–309. http://tm.ifmo.ru/tm2002/db/doc/get_thes.php?id=170
3. **Столяр С. Е., Осипова Т. Г., Крылов И. П., Столяр С. С.** Об инструментальных средствах для курса информатики // Труды II Всероссийской конференции «Компьютеры в образовании». – СПб., 1994. – С.18–19.
4. **Казаков М. А., Столяр С. Е.** Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования // Материалы международн. научно-методич. конф. «Телематика'2000». – СПб., 2000. – С.189–191.
5. **Казаков М. А., Шалыто А. А.** Использование автоматного программирования для реализации визуализаторов // Компьютерные инструменты в образовании. 2004. – № 2. – С.19–33. http://is.ifmo.ru, раздел «Статьи».
6. **Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А.** Обход деревьев на основе автоматного подхода // Компьютерные инструменты в образовании. – 2004. – № 3. – С. 32–37. http://is.ifmo.ru, раздел «Статьи».
7. **Шень А.** Программирование: теоремы и задачи. – М.: МЦНМО, 2004. – 296 с.
8. **Шалыто А. А., Туккель Н. И.** Преобразование итеративных алгоритмов в автоматные // Программирование. – 2002. – № 5. – С. 12–26. http://is.ifmo.ru, раздел «Статьи».
9. **Шалыто А. А.** SWITCH-технология. Алгоритмизация и программирование задач логического управления. – СПб.: Наука, 1998. – 628 с.
10. **Кормен Т., Лайзерсон Ч., Ривест Р.** Алгоритмы. Построение и анализ. – М.: МЦНМО, 2000. – 960 с.

ВАРШАВСКИЙ Виктор Ильич

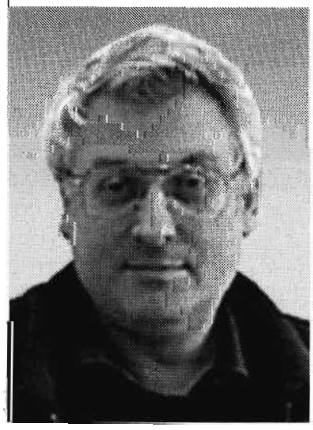
Профессор, в последние годы жизни – заведующий отделом логического управления в фирме «Технологии нейронных сетей» (Тель-Авив, Израиль).
В 1956 году окончил Ленинградский институт точной механики и оптики.
В 1970 году защитил диссертацию на соискание ученой степени доктора технических наук.
Автор более 200 научных публикаций, соавтор и редактор восьми книг, автор более 150 изобретений.
Научные интересы – асинхронные электронные устройства и системы, пороговая логика и искусственные нейроны, коллективное поведение автоматов.

ЛЕВИН Илья Семенович



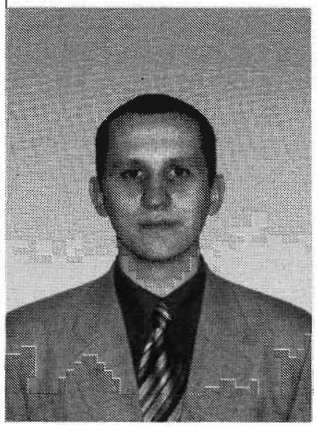
Профессор кафедры «Computer Engineering» Бар-Иланского университета (Израиль).
В 1976 году окончил электротехнический факультет Ленинградского института инженеров железнодорожного транспорта по специальности «Электронные вычислительные машины».
В 1987 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 100 научных публикаций.
Область научных интересов – теория автоматов, системы автоматизации проектирования, логическое управление, проектирование надежных вычислительных систем.

ШАЛЫТО Анатолий Абрамович



Заведующий кафедрой технологий программирования Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Ученый секретарь НПО «Аврора».
В 1971 году окончил Ленинградский электротехнический институт по специальности «Автоматика и телемеханика».
В 1999 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 250 научных публикаций, трех монографий и 70 изобретений.
Область научных интересов – системы логического управления, автоматное программирование.

КАЗАКОВ Матвей Алексеевич



Аспирант, ассистент кафедры компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики.
В 2002 году окончил Санкт-Петербургский государственный университет информационных технологий, механики и оптики с присуждением степени магистра математики по направлению «Прикладная математика и информатика».
Является автором более десяти научных публикаций.
Область научных интересов – автоматное программирование, дистанционное обучение программированию, визуализаторы алгоритмов.

МАРАХОВСКИЙ Вячеслав Борисович

Профессор, заведующий лабораторией логического проектирования вычислительных устройств университета Айдзу (Япония).
В 1963 году окончил Ленинградский политехнический институт.
В 1992 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 200 научных публикаций, соавтором четырех монографий, автором около 80 изобретений.
Область научных интересов – логическое проектирование устройств, прикладная теория автоматов, проектирование асинхронных устройств, логическая синхронизация и децентрализованное управление в вычислительных системах.

УДК 681.3

Системное время и синхронизация систем

Варшавский В. И. Информационно-управляющие системы, 2005. – № 4. – С. 6–21.

В статье проводится анализ концепции причинной обусловленности как основы правильного асинхронного поведения. Предлагается декомпозировать систему на процессорный стратум и синхро-стратум, функционирующий подобно распределенным асинхронным часам. Универсальный синхро-стратум обеспечивает возможность прямого перехода от синхронного прототипа к асинхронной системе. Приводятся несколько типов спецификаций и аппаратной реализации синхро-стратума вместе с их сравнительными характеристиками.

Список лит.: 19 назв.

UDK 681.3

System Time and System Timing

Varshavsky V. I. IUS, 2005. – N 4. – P. 6–21.

In the paper, the concept of causal conditionality is analyzed as the basis for correct asynchronous behavior. It is suggested to decompose the system to Processors Stratum and Synchro-Stratum which acts like a distributed asynchronous clock. A universal Synchro-Stratum provides the possibility of direct transition from the synchronous prototype to an asynchronous system. Several types of specification and hardware implementation of Synchro-Stratum are given with their characteristics compared.

Pefs: 19 titles.

УДК 681.3

Логическое проектирование и квантовый вызов

Варшавский В. И. Информационно-управляющие системы, 2005. – № 4. – С. 22–32.

Готова ли методология проектирования логических и вычислительных структур к эффективному использованию новых функциональных возможностей квантовых устройств? Обеспечивает ли ряд существующих и предлагаемых квантовых устройств эффективное проектирование логических и вычислительных структур? На основе примеров обсуждается возможность получения положительных ответов на эти вопросы за счет объединенных усилий физиков, технологов и экспертов в вычислительной технике. Статья указывает возможные направления, где такое взаимодействие может оказаться продуктивным.

Список лит.: 26 назв.

UDK 681.3

Logic Design and Quantum Challenge

Varshavsky V. I. IUS, 2005. – N 4. – P. 22–32.

Is the design methodology of logical and computing structures ready to efficiently use new functional possibilities of quantum devices? Does the range of existing and suggested quantum devices provide effective design of logical and computing structures? With some examples, we discuss the possibility of obtaining positive answers to these questions uniting the efforts of physicists, technologists and experts in computer engineering. The paper indicates some directions where such collaboration may prove to be productive.

Pefs: 26 titles.

УДК 681.3

Самосинхронизируемый конечный автомат: от примера к синтезу

Варшавский В. И., Мараховский В. Б. Информационно-управляющие системы, 2005. – № 4. – С. 33–37.

В статье рассматриваются проблемы синтеза самосинхронных устройств и способы их решения. Показано, что широко используемый язык конечных автоматов может успешно применяться для проектирования таких устройств. Возникающие проблемы синтеза и методы их решения иллюстрируются на примере проектирования самосинхронного буферного запоминающего устройства типа СТЕК.

Список лит.: 14 назв.

UDK 681.3

Self-Timed Finite State Machine: from Example to Synthesis

Varshavsky V. I., Marakhovsky V. B. IUS, 2005. – N 4. – P. 33–37.

In the article the problems of self-timed devices synthesis and the methods of their solution are discussed. It is shown that the widely used language of finite automata can be successfully applied for designing such devices. The appearing problems of synthesis and the methods of their solution are illustrated on the base of the example of designing self-synchronous buffer memory of the STACK type.

Pefs: 14 titles.

УДК 681.3.06

КМОП пороговые элементы с функциональными входами

Варшавский В.И., Мараховский В.Б., Левин И.С. Информационно-управляющие системы, 2005. – № 4. – С. 38–50.

Предлагается метод увеличения функциональных возможностей порогового элемента за счет введения так называемых функциональных входов. Каждый такой вход соответствует булевой сумме (или произведению) некоторого подмножества входных переменных. Показано, что бета-управляемй пороговый элемент с функциональными входами способен реализовать произвольные монотонные булевы функций. Предложены КМОП реализации таких элементов и методы токовой стабилизации функциональных входов. Приводятся примеры реализации сложных логических функций на основе бета-управляемых пороговых элементов с функциональными входами. Представлены также результаты SPICE-моделирования поведения предложенного элемента.

Список лит.: 21 назв.

UDK 681.3.06

CMOS – Driven Threshold Elements with functional inputs

Varshavsky V.I., Marakhovsky V.B., Levin I.S. IUS, 2005. –N 4. – P. 38–50.

A method for increasing the functional capability of threshold elements by introducing so-called functional inputs is proposed. Each functional input corresponds to a Boolean sum (or product) of a particular subset of input variables. It is shown that introducing functional inputs enables expansion of the functional capability of beta-driven elements up to the capability to implement an arbitrary monotonic function. The CMOS based implementation of the beta-driven threshold element with newly proposed functional inputs is presented. Methods of the current stabilization of functional inputs are proposed. The paper presents examples of the SPICE simulation of the proposed threshold element behavior.

Pefs: 21titles.

УДК 681.3.06

Реализация анимации при построении визуализаторов алгоритмов на основе автоматного подхода

Казаков М.А., Шалыто А.А. Информационно-управляющие системы, 2005. – № 4. – С. 51–60.

В статье предлагается подход к реализации анимации алгоритмов, который расширяет технологию построения визуализаторов алгоритмов на основе автоматного подхода.

Список лит.: 10 назв.

UDK 681.3.06

Implementation of animation in viewers of algorithms designing based on automatic approach

Kazakov M.A., Shalyto A.A. IUS, 2005. –N 4. – P. 51–60.

Article describes proposed approach for algorithms animation realization, which extend technology of algorithms' viewers creation based on the automatic approach.

Pefs: 10 titles.

ПАМЯТКА ДЛЯ АВТОРОВ

Поступающие в редакцию статьи проходят обязательное рецензирование.

Редакция журнала напоминает, что ответственность за подбор, достоверность и точность фактов, экономико-статистических и технических показателей, собственных имен и прочих сведений, а также за то, что в материалах не содержится сведений, не подлежащих открытой публикации, несут авторы публикуемых в журнале материалов и рекламодатели.

Ноябрьская демонстрация 1952 г., слева
В. И. Варшавский

Виктор Ильич Варшавский (1933 - 2005)

Университет Айдзу, Япония.
В. И. Варшавский с В. Б. Мараховским

В. И. Варшавский с сотрудниками фирмы
"Технологии нейронных сетей", Израиль

С внуками Ильей и Андреем

Одна из последних
фотографий, Израиль

С женой Натальей