Аннотация курса «Теория логического проектирования»

Курс состоит из двух частей: теоретические основы проектирования самосинхронных схем и применение многозначной логики при проектировании аналоговых устройств управления, заданных средствами «размытой, или нечеткой» логики. В основном излагается оригинальный материал, ранее не представленный в программах Российских университетов.

Первая часть курса содержит систематическое описание теории самосинхронных схем. Ее целью является ознакомление студентов с основными принципами и методами проектирования асинхронных логических и управляющих устройств, а также приобретение студентами навыков как «ручного» проектирования так и с использованием формальных методов, реализованных в системе автоматизации проектирования Petrify.

Во второй части курса излагаются теоретические вопросы и методы построения аналоговых управляющих устройств, заданных в виде фаззи-контроллеров, схемами из суммирующих усилителей, реализующих многозначные логические функции. После изучения второй части курса студенты должны овладеть методами декомпозиции многозначных логических функций и уметь синтезировать логическую схему любого фаззи-контроллера.

Ключевые слова: асинхронный процесс; динамическая модель; сеть Петри; сигнальный граф; самосинхронные коды; самосинхронная схема; свойство полумодулярности; схема, не зависящая от скорости; фаззи-контроллер; фаззификатор; дефаззификатор; фаззи-вывод; функция принадлежности; многозначная логика; суммирующий усилитель; декомпозиция.

Теория Логического Проектирования

Проф. Мараховский Вячеслав Борисович

Часть 1:
Логическое Проектирование Асинхронных Схем

Лекция 1

Источники

1. *Апериодические автоматы,* под ред. В.И.Варшавского, М., Наука, 1976.

2. *Автоматное управление асинхронными процессами в ЭВМ и дискретных системах,* под ред. В.И.Варшавского, М., Наука, 1986.

3.M. Kishinevsky, A. Kondratyev, A. Taubin and V. Varshavsky, *Concurrent Hardware. The theory and Practice of Self-timed Design*, J. Wiley & Sons, 1993.

4.J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, *Logic Synthesis of Asynchronous Controllers and Interfaces*, Springer-Verlag, 2002.

5.Jens Sparsø, Asynchronous Circuit Design. A Tutorial, Technical University of Denmark, 2006. Free copy: http://www.ee.technion.ac.il/courses/048878/book.pdf

6.Varshavsky, V.; Marakhovsky, V.; Levin, I., Saito, H., "Multi-Valued Logic Approach to Fussy Controllers Implementation", *WSEAS Transaction on Electronics*, Issue 6, Vol.4, June 2007, pp. 109 – 133.

7.Статьи в журналах и конференциях.

1. Введение

1.1 Мотивация

Существует важный подкласс современных систем управления – *управляющие устройства, осуществляющие координацию процессов с конечным множеством состояний*. Примеры из самых различных областей: управление вычислительными процессами, обменом информации в сетях ЭВМ, пуском и остановкой энергоустановок, локомоторной деятельностью роботов, химическими процессами и др.

Существенные черты таких систем:

- наличие у процессов (подпроцессов) *явно выраженных фаз*, в течение которых происходит изменение их состояний; часто в каждой фазе выдается информация о ее завершении;
- *согласованность* – переход к следующей фазе процесса инициируется только после получения сигнала об окончании предыдущего перехода;
- *параллельность* – возможность одновременных переходов в нескольких процессах;
- *асинхронность* – отсутствие ограничений на длительность переходов, зависящих от многих неконтролируемых факторов.

1. Введение

Параллельность и *асинхронность* – естественные черты практически всех технических систем.

Вариации длительностей переходных процессов в элементах, блоках и каналах связи порождаются разбросом технологических параметров и изменениями физических условий функционирования (температуры, питающего напряжения, давления и т.д.). *Асинхронность* выдвигает требования инвариантности поведения системы к изменениям длительностей переходных процессов.

Согласованность отражает причинно-следственные связи взаимодействия объекта и системы управления, а также связи с внешней средой.

Поскольку речь идет о системе с конечным множеством состояний, то в качестве ее модели можно использовать конечный автомат. Однако такая модель игнорирует понятие физического времени, а любое устройство, ее реализующее, является динамической системой, функционирующей в реальном времени. Обычно это противоречие преодолевается за счет использования системы внешней синхронизации.

1. Введение

Недостатки внешней синхронизации:

- сложность проектирования системы доставки синхросигналов в точки синхронизации из-за разброса задержек в проводах и усилителях;
- дополнительный расход мощности (до 30-40%);
- недоиспользование скоростных возможностей элементов (расчет на худший случай);
- возможность появления параметрических отказов из-за изменения условий функционирования;
- источник возникновения помех из-за одновременного срабатывания большого числа элементов;
- Трудность организации согласованной работы многопроцессорной системы с индивидуальными подсистемами синхронизации из-за явления арбитража, так как не существует абсолютно надежных синхронизаторов, которые всегда обладают некоторой вероятностью сбоя. Любые такие системы подвержены сбоям.

Основное достоинство – простота проектирования устройств и минимальные затраты оборудования.

1. Введение

Попытки избежать этих недостатков с помощью проектирования устройств на основе *классической теории асинхронных автоматов* не привели к желаемым результатам. Переходы автомата в этой теории инициируются не сигналами от внешних часов, а сменой набора входных сигналов. При этом часы, внешние по отношению к системе «автомат – внешняя среда», переносятся во внешнюю среду и управляют частотой смены входных наборов.

Достоинством таких устройств является *отсутствие системы доставки синхросигналов* в точки синхронизации, однако сохраняются остальные недостатки синхронных устройств и дополнительно возникают новые (более сложные) проблемы:

- очень жесткие ограничения на организацию внешней среды, определяемые устранением функциональных состязаний;
- необходимость борьбы с логическими состязаниями;

необходимость применения мало эффективного противогоночного кодирования внутренних состояний.

1. Введение

Неприятности, связанные с синхронизацией от часов, объясняются тем, что события в часах никак не связаны с событиями в системе «автомат – внешняя среда».

С другой стороны, смена событий в системе в реальном физическом времени порождает систему отношений между событиями в автомате и событиями во внешней среде и эти отношения являются системным временем для системы «автомат – внешняя среда». Квантование времени событиями, происходящими в такой системе, приводит к понятию *самосинхронизации*.

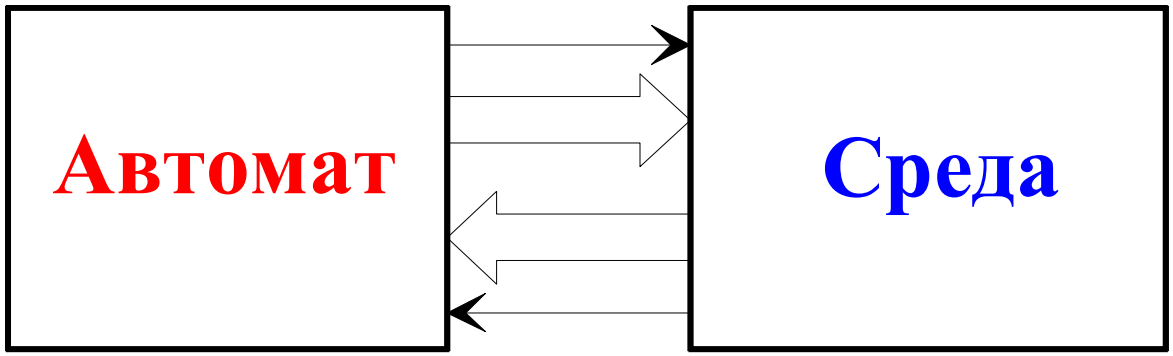
Попытки ввести систему отношений между событиями, определяющую время, предпринимались еще Аристотелем, но первая строгая система была предложена, по-видимому, Дж. Н. Финдлером в 1941 году.

1. Введение

1.2 Самосинхронизация

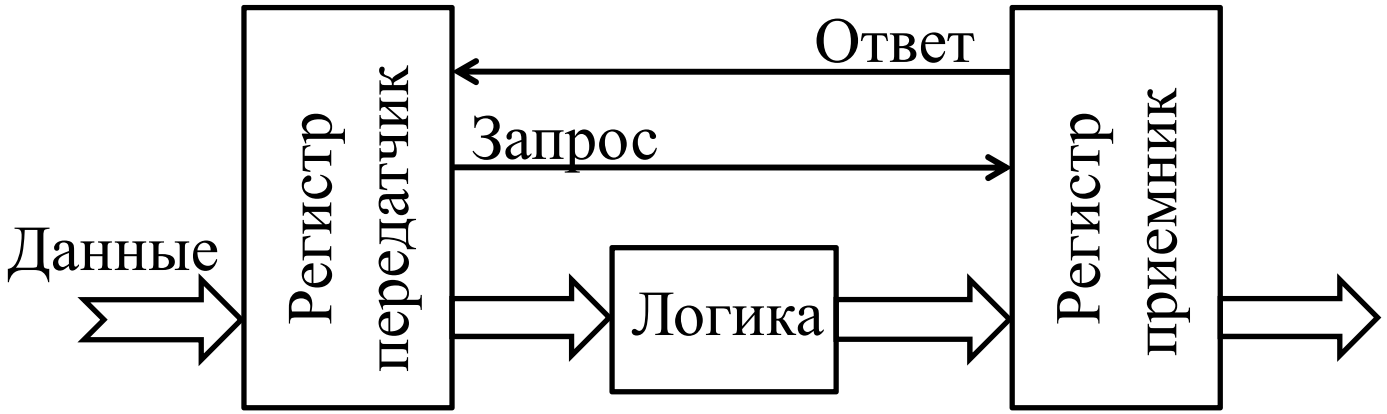
Структура системы «автомат – внешняя среда», в которой внешняя среда порождает время для автомата и автомат порождает время для окружающей среды, называется ***согласованной моделью***.

Автомат в этой модели называется ***самосинхронным***.



1. Введение

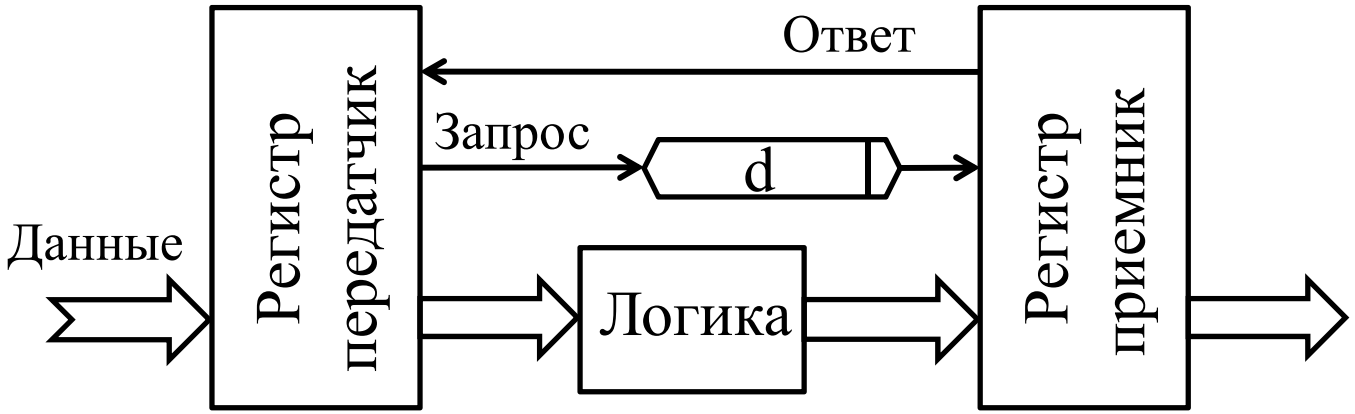
Асинхронное взаимодействие модулей устройства



- Асинхронное взаимодействие по принципу ***«запрос – ответ»*** (*handshake*) вместо синхросигналов от часов.
- Запрос-ответное взаимодействие может быть реализовано с помощью ***встроенной задержки*** и/или ***детектора окончания переходных процессов***.

1. Введение

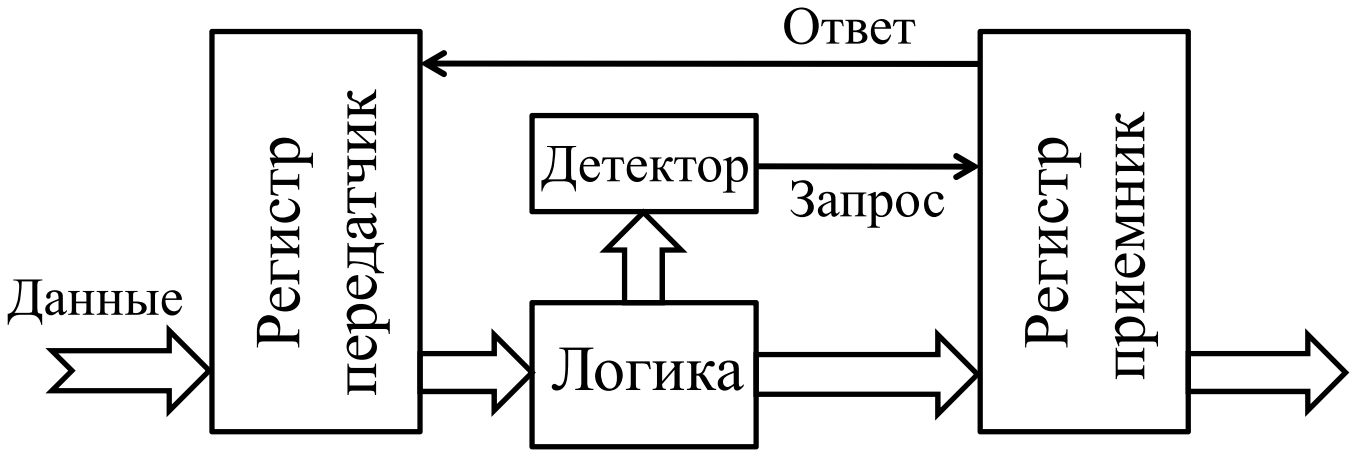
Использование встроенной задержки



- Очень похоже на синхронную реализацию. Встроенная задержка может рассматриваться как измеритель времени.
- Величина встроенной задержки должна превышать максимальную задержку логической схемы и задержку записи данных в приемный регистр.
- Временная избыточность. Ограничение на максимальную длительность переходных процессов.

1. Введение

Использование детектора окончания переходных процессов



- Избыточное кодирование данных самосинхронными кодами $\Rightarrow$ дополнительный расход площади кристалла.
- При завершении переходного процесса детектор вырабатывает сигнал «запрос» $\Rightarrow$ дополнительный расход площади.
- Детектор завершения переходного процесса может находиться в приемнике.
- Инвариантность к разбросу задержек в проводах входных данных логики.

1. Введение

Потенциальные преимущества самосинхронных схем:

- Отсутствуют часы и система доставки синхросигналов.
- Производительность определяется не максимальными задержками элементов, а реальными усредненными задержками.
- Пониженное потребление мощности.
- Инвариантность к разбросам задержек элементов и, как следствие, устойчивость к параметрическим отказам (robustness).
- Возможность модульного построения сложных устройств.
- Самопроверяемость некоторого класса самосинхронных устройств относительно константных неисправностей и, как следствие, возможность организации саморемонта.
- Возможность эволюционного развития.
- Метастабильность не опасна. Она либо не возникает, либо завершается за произвольное время.
- Поскольку отсутствует система глобальной синхронизации удобно применять при совместном аппаратно/программном проектировании (hardware/software codesign).

1. Введение

Проблемы:

- Большой расход оборудования или площади кристалла (иногда трехкратный) по сравнению с синхронным системам.
- Сложность проектирования (необходимость в специалистах высокого уровня).
- Не работают накопленный опыт проектирования и существующая схемотехника, т.е. необходимость перепроектирования всех устройств.
- Сложность тестирования.
- Сложность разработки CAD систем (САПР – систем автоматизации проектирования).

Тем не менее, крайне необходимо развивать методологию проектирования самосинхронных систем.

2. Асинхронные процессы и их интерпретация

2.1 Метамодель асинхронного процесса

Определение 2.1. Назовем *асинхронным процессом* (АП) четверку <*S*, *F*, *I*, *R*>, в которой S – непустое конечное множество *ситуаций,* F – *отношение непосредственного следования ситуаций*, определенное на множестве $S \times S$ пар ситуаций ($F \subset S \times S$), I – множество *инициаторов* $(I \subset S)$, т.е. таких ситуаций из *S,* для которых, если $iFs_k, i \in I$ и $s_k \notin I$, то из $s_k Fs_l$ следует , что $s_l \notin I$, R – множество *результантов* $(R \subset S)$, т.е. таких ситуаций из *S*, для которых, если $rFs, r \in R$, то $s \in R$.

Инициаторы – подмножество ситуаций, активизирующих процесс.
Результанты – подмножество, состоящее из финальных ситуаций.
Их назначение зависит от семантики АП.

АП можно представить в виде орграфа, вершинами которого являются ситуации. В таком графе можно выделить множества последовательностей ситуаций.

2. Асинхронные процессы и их интерпретация

Определение 2.2. Назовем последовательность *допустимой*, если она не является частью любой другой последовательности (определяет возможный ход процесса).

На множестве ситуаций АП можно ввести *отношение эквивалентности E* (симметричное, рефлексивное, транзитивное), разбивающее множество ситуаций на классы эквивалентности. Заменяя каждый класс эквивалентности одной ситуацией, можно минимизировать множество *S* и соответственно *I* и *R*. Заметим, что классы эквивалентности множеств *I* и *R* могут содержать соответственно только инициаторы и результанты, остальные классы эквивалентности не содержат ситуаций из этих множеств.

Минимизация числа ситуаций приводит к ликвидации циклов в орграфе АП и все последовательности ситуаций в АП становятся конечными.

Определение 2.3. АП назовем *эффективным*, если все цепочки ситуаций конечны и ведут из инициаторов в результанты.

2. Асинхронные процессы и их интерпретация

Определение 2. 4. Если каждая траектория в АП содержит в точности по одному инициатору и результанту, то такой АП будем называть *простым*.

Определение 2.5. *Протоколом* простого АП будем называть отношение $Q \subseteq I \times R$.

Протокол простого АП – это такой АП, в котором за каждым инициатором непосредственно следует результант и $S \subseteq I \times R$ Протокол является удобной формой «вход-выходного» описания поведения АП, когда нас не интересует внутренне содержание АП.

Для того, чтобы сделать АП возобновляемым, введем понятие *репозиции* АП.

Определение 2.6. *Репозицией* АП $P = < S, F, I, R >$ назовем эффективный АП $P' = < S', F', I', R' >$ такой, что $S' \subseteq I \cup R \cup S^{\partial}$, $I' \subseteq R$, $R' \subseteq I$.

Ситуации репозиции S' могут содержать инициаторы и результанты основного процесса и, возможно, некоторые дополнительные ситуации S^{∂}, не принадлежащие S. Репозиция АП может быть *полной*, если $I' = R$, $R' = I$, или *частичной*.

2. Асинхронные процессы и их интерпретация

Часто репозицию АП удобно задавать протоколом.

АП вместе с его репозицией образует *автономный асинхронный процесс*.

Рассмотренную модель асинхронного процесса можно считать самой общей *дискриптивной неинтерпретированной моделью,* описывающей порядок действий над объектами, которые в частности, могут быть представлены параллельно функционирующими асинхронными схемами. Ее также можно понимать как *метамодель*, порождающую различные широко используемые динамические модели.

Частные поведенческие модели порождаются использованием механизма интерпретации АП, который может быть двухступенчатым. Первый этап является *модельной интерпретацией*, ставящей в соответствие основным понятиям АП понятия частных моделей. Эти модели, в свою очередь, также являются неинтерпретированными.

Второй этап – *предметная интерпретация*, которая зависит от приложений. Рассмотрим некоторые из наиболее известных частных интерпретаций АП.

2. Асинхронные процессы и их интерпретация

2.2. Сети Петри

Динамическая модель, предложенная К.А.Петри для описания потоков *событий*. Событие можно рассматривать как некий компонент ситуации АП.

Определение 2.7. Сетью Петри называется пятерка $N =< P, T, M_0, H, F >$, где

$P = \{p_1, p_2, ..., p_n\}$ – конечное непустое множество *условий*,

$T = \{t_1, t_2, ..., t_m\}$ – конечное непустое множество *событий*,

$\left.\begin{array}{l} F : P \times T \to \{0,1\} \\ H : T \times P \to \{0,1\} \end{array}\right\}$ – *функции инцидентности*,

$M_0 : P \to \{0,1,2,...\}$ – *начальная разметка.*

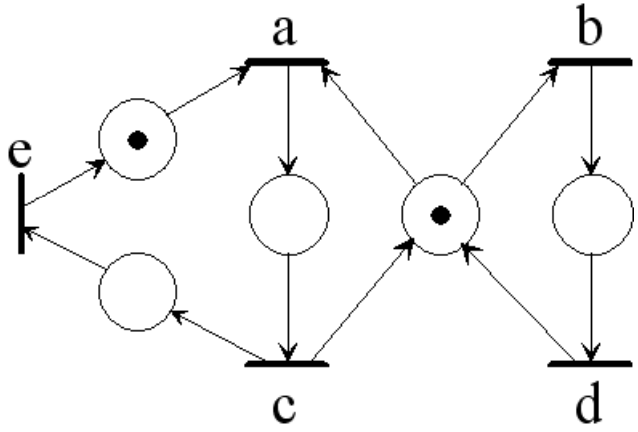
Разметкой M сети N называется функция $M : P \to \{0,1,2,...\}$.

Сеть петри обычно представляют в виде двудольного орграфа из кружков , соответствующих условиям, и полочек, соответствующих событиям. Начальная разметка M_0 осуществляется точками в некоторых кружках. Текущая разметка M указывает число точек $M(p_i)$ в каждом кружке p_i.

2. Асинхронные процессы и их интерпретация

Событие становится *возможным* при выполнении всех его входных условий (во всех его входных кружках появились точки), в противном случае событие является *невозможным*. При реализации возможного события изымается по одной точке из всех его входных условий и добавляется по одной точке во все его выходные условия (неделимая операция).

Сеть Петри функционирует, переходя от разметки к разметке путем реализации возможных событий. Последовательность разметок (маркировок) в сети отражает динамику процесса. Поток событий распространяется в направлении дуг.



2. Асинхронные процессы и их интерпретация

Сети Петри позволяют учитывать реальную *асинхронность* событий и *непредсказуемость* времени их наступления.

Разметку *M*, при которой ни одно из событий сети *N* невозможно, называют *тупиковой*.

Разметка *M* достижима* в сети *N* из разметки M, если существует последовательность разметок (маркировок), ведущая от *M* к *M**.

Событие называется *живым,* если из начальной разметки достижима разметка, при которой оно становится возможным. Сеть называют *живой*, если каждое ее событие живо.

Сеть называют *устойчивой*, если она не содержит *конфликтных* разметок. Конфликтные разметки могут возникать, например, при наличии *вершин свободного выбора.*

Сеть Петри называют *безопасной*, если в каждом из ее условий может возникнуть не более одной точки, т.е. $M(p_i) \le 1$ для всех $p_i \in P$.

2. Асинхронные процессы и их интерпретация

Сеть Петри – это модельная интерпретация АП.

Множество ситуаций – это множество разметок.

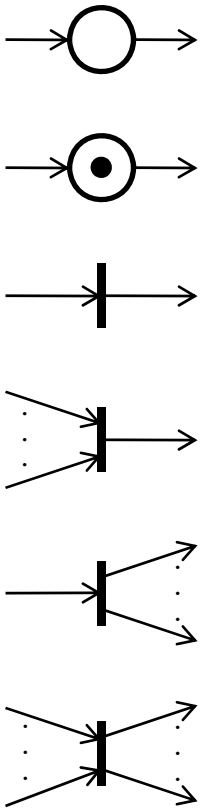
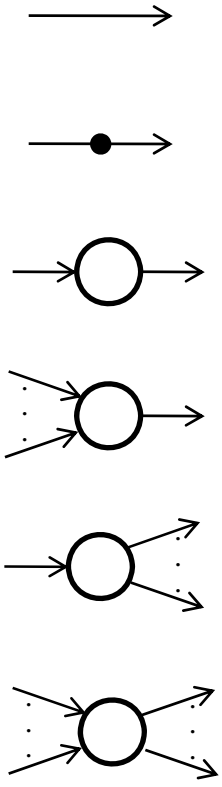
Работа реальных объектов часто описывается некоторым множеством сетей Петри, отличающимися лишь начальными разметками (с одинаковой структурой графа). Для таких сетей Петри (со входными и выходными условиями), представляющих неавтономные объекты,
инициаторы – это множество начальных разметок,
результанты – множество финальных разметок.

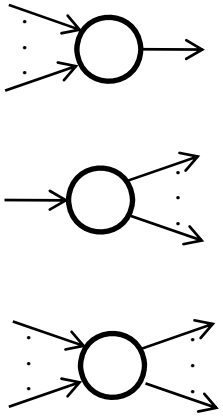
2.3. Сигнальные графы

Определение 2.8. Сеть Петри, в которой любое условие может иметь только по одной входной и одной выходной дуге, называется *маркированным графом*.

В силу традиции маркированные графы изображаются в виде обычного ориентированного графа, дуги которого маркируются точками.

2. Асинхронные процессы и их интерпретация

Фрагмент сети Петри	*Фрагмент маркированного графа*
	

Фрагмент сети Петри	*Фрагмент маркированного графа*
	—
	—
	—

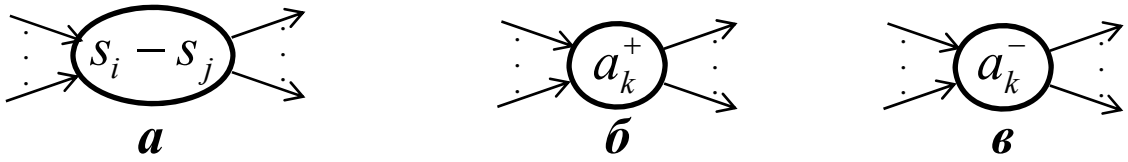
Вершины маркированного графа, имеющие более одной входной дуги, называют ***синхронизаторами***, а имеющие более одной выходной дуги, – ***бифуркаторами***.

2. Асинхронные процессы и их интерпретация

Маркированные графы являются подклассом сетей Петри; доказано, что их функциональные возможности ниже, чем у сетей Петри. Маркированные графы, как и сети Петри, могут быть *живыми*, *безопасными*, *автономными*. Они могут моделировать *параллельные процессы*, но не могут моделировать *конфликты*, так как отсутствуют вершины с альтернативными выходами (такие графы всегда *устойчивы*).

Предметная интерпретация маркированных графов может быть введена следующим образом:

Три типа вершин в сигнальном графе



а ***б*** ***в***

Вершина ***а*** обозначает переход событий $s_i - s_j$, любое промежуточное событие перехода не вызывает изменения маркировки. События переходов можно структурировать, поставив им в соответствие некоторые двоичные наборы. Вершины ***б*** и ***в*** обозначают изменение значения некоторой компоненты события.

2. Асинхронные процессы и их интерпретация

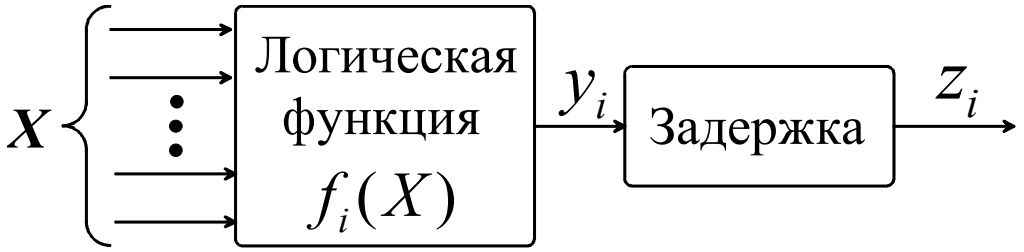
Определение 2.9. ***Сигнальным графом*** будем называть предметную интерпретацию маркированного графа такую, что его вершины помечены обозначениями переходов $s_i - s_j$, a_k^+, a_k^-, а смена маркировок подчиняется правилу смены маркировок в сети Петри.

2.4. Модель Маллера

Рассмотрим автономный АП , обладающий следующими свойствами:

1. Ситуации из S структурированы: представлены в виде наборов значений двоичных переменных $z_i = (a_1, a_2, ..., a_n)$, $a_j = \{0,1\}$. Назовем их *состояниями*.

2. Отношение F изображается орграфом, вершины которого соответствуют состояниям, а направленные дуги – парам из F. Значение каждого элемента состояния вычисляется с помощью следующей схемы



2. Асинхронные процессы и их интерпретация

$z_i = \{0,1,0^*,1^*\}$,

$z_i = \{0,1\}$ – устойчивые значения, $z_i = y_i = f_i(\boldsymbol{X})$,

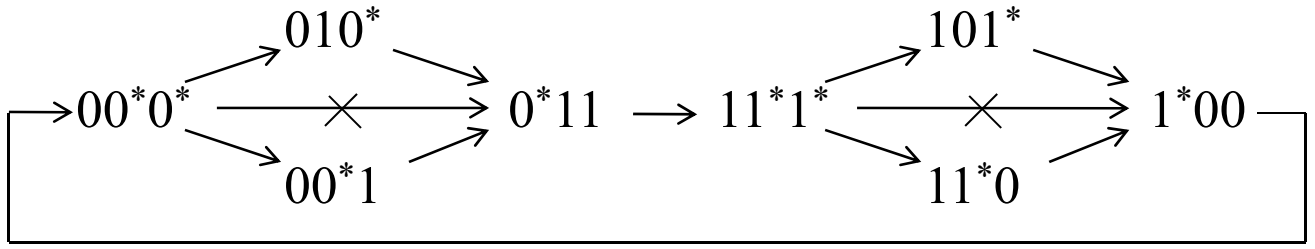
$z_i = \{0^*,1^*\}$ – неустойчивые значения $z_i \neq y_i = f_i(\boldsymbol{X})$.

Элементы состояния, помеченные звездочками, называются *возбужденными*, а остальные – *устойчивыми*.

3. Инициаторы и результанты назначаются в соответствии с их определениями в АП.

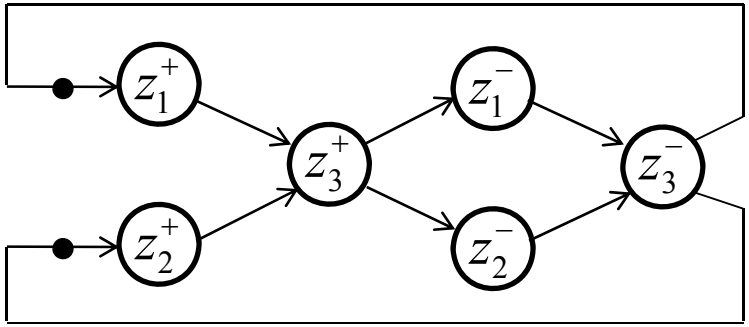
Определение 2.10. Модельную и предметную интерпретацию асинхронного процесса, удовлетворяющую эти свойствам, будем называть ***диаграммой переходов***.

Диаграмма переходов, соответствующая некоторому автономному АП



2. Асинхронные процессы и их интерпретация

Представление диаграммы переходов в виде сигнального графа



Состояние s_i диаграммы переходов называют конфликтным, если существует состояние s_j такое, что $s_i F s_j$ и в обоих состояниях имеется хотя бы один элемент x_k с одинаковым значением, но в s_i это значение неустойчиво (со значком *), а в s_j – устойчиво. Состояние, в котором впервые возникает возбуждение хотя бы двух переменных с разными значениями также является конфликтным.

$0 \longrightarrow 0$
$0^* \longrightarrow 1$
$1 \longrightarrow 1$
$1^* \longrightarrow 0$

Допустимые переходы

$\mathbf{0^* \longrightarrow 0}$
$\mathbf{1^* \longrightarrow 1}$

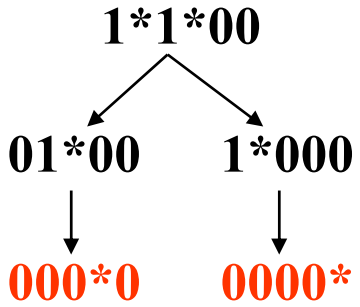
Переходы, приводящие к конфликту

2. Асинхронные процессы и их интерпретация

2. Состояния s_k и s_l диаграммы переходов называются *противоречивыми* (*несовместимыми*), если

$$s_k = z_1,\ldots,z_i^*,\ldots,z_j,\ldots,z_n$$
$$s_l = z_1,\ldots,z_i,\ldots,z_j^*,\ldots,z_n$$

Пример



Определение 2.11. Диаграмму переходов будем называть ***полумодулярной***, если она не содержит конфликтных и противоречивых состояний. (Это определение в некотором роде аналогично определению устойчивости сети Петри).

Поясним причину использования понятия *полумодулярности*. Для этого введем термин ***кумулятивное состояние***, под которым понимается вектор, *i*-я компонента которого представляет число изменений значения компоненты z_i на траектории, ведущей из некоторого исходного состояния.

Построим диаграмму переходов на кумулятивных состояниях, в которой каждая переменна z_i изменяется четное число раз за цикл работы схемы. Легко видеть, что такая диаграмма является алгебраической структурой.

2. Асинхронные процессы и их интерпретация

Определение 2.12. Будем называть диаграммы переходов ***полумодулярными*** (***U***), ***дистрибутивными*** (***D***) или ***последовательными*** (***K***), если соответствующие им диаграммы на кумулятивных состояниях образуют соответственно ***полумодулярную***, ***дистрибутивную*** или ***полностью упорядоченную*** алгебраическую структуру.

Если обозначить множество диаграмм переходов как ***W***, то имеет место

$$K \subset D \subset U \subset W.$$

Определение 2.13. ***Моделью Маллера*** асинхронного процесса будем называть систему булевых уравнений

$$z_i = f_i(z_1,...,z_i,...,z_n),\ i = 1,2,...,n.$$

Модель Маллера можно рассматривать как аналитическое описание логической схемы. Если эта модель соответствует неавтономному процессу, то часть переменных z_i, являются входами схемы, а другая часть переменных служит выходами схемы.

2. Асинхронные процессы и их интерпретация

Для нахождения системы уравнений по диаграмме переходов следует воспользоваться таблицей истинности, составленной следующим образом.

Табица истинности для примера на слайде 25.

z_1	z_2	z_3	Z_1	Z_2	Z_3
0	0*	0*	0	1	1
1*	0	0	0	0	0
0	1	0*	0	1	1
1	1*	0	1	0	0

z_1	z_2	z_3	Z_1	Z_2	Z_3
0	0*	1	0	1	1
1	0	1*	1	0	0
0*	1	1	1	1	1
1	1*	1*	1	0	0

Не встречающиеся в диаграмме наборы могут быть доопределены произвольно.

С помощью любой известной процедуры минимизации булевых функций по таблице можно найти систему уравнений Маллера. В данном случае имеем:

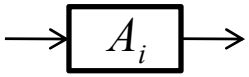
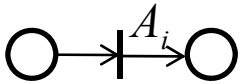
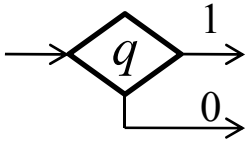
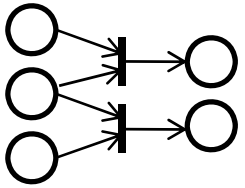
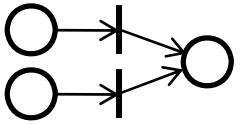
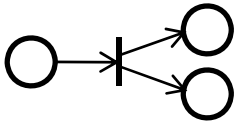
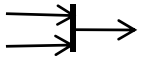
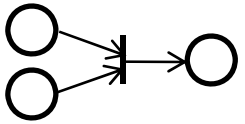
$$z_1 = z_2 z_3 \vee (z_2 \vee z_3) z_1, \quad z_2 = \bar{z}_1, \quad z_3 = \bar{z}_1.$$

Модель Маллера может описывать АП с точностью до собственных функций элементов соответствующей ей схемы.

2. Асинхронные процессы и их интерпретация

2.5. Параллельные асинхронные блок-схемы

Параллельные асинхронные блок-схемы (ПАБС) – это такие граф-схемы алгоритмов, которые могут быть представлены в виде орграфа с пятью типами вершин: оператор, условный переход, сборка, бифуркатор и синхронизатор.

№	Тип вершины ПАБС	Наименование	Фрагмент Сети Петри
1		Оператор	
2		Условный переход	
3		Сборка	
4		Бифуркатор	
5		Синхронизатор	

2. Асинхронные процессы и их интерпретация

Операторы и условные переходы определяют предметную интерпретацию АП. Модельная интерпретация определяется механизмом маркировки входных и выходных дуг как и в маркированных графах.

Обычно в ПАБС выделяют две дополнительные вершины: начальный оператор и оператор конца. Тогда инициатором процесса является точка на выходной дуге начального оператора, а результантом процесса – маркировка, при которой все точки поглощены конечным оператором.

2.6. Асинхронные конечные автоматы

Были рассмотрены в рамках курса «Логическое проектирование».

Модельная интерпретация АП в терминах конечных асинхронных автоматов состоит в следующем. Ситуация – полное состояние автомата $s_i = (x_j^a, y_k^a, z_l^a)$, где x_j^a, y_k^a и z_l^a – входной символ, выходной символ и символ внутреннего состояния соответственно. На уровне абстрактного синтеза автомата нормальная таблица переходов и выходов по существу является протоколом простого АП. Промежуточные ситуации могут возникать на этапах кодирования.

3. Самосинхронизирующиеся коды

Рассмотрим некоторые вопросы кодирования сигналов при построении самосинхронных схем и систем передачи информации, поведение которых инвариантно к временным параметрам приёмопередающих устройств и каналов связи.

Поставим в соответствие ситуациям АП наборы значений двоичных переменных $z_1,\ldots,z_i,\ldots,z_n$. Пусть набор A соответствует ситуации s_i, а набор B ситуации s_k.Тогда переход $s_i - s_k$ может быть представлен переходом $A - B$.

Наборы A и B могут быть *соседними*, *сравнимыми* и *несравнимыми*. Далее потребуются понятия *характеристической функции* набора (*коституента*), *подкуба перехода* и *терма перехода*. Необходимы также понятия *изотонной*, *антитонной* и *монотонной* логической функции. (См. лекции курса «Логическое проектирование»).

Пусть имеется кодовая система $\boldsymbol{Z}$ для кодирования некоторого множества символов алфавита $\boldsymbol{\Sigma}$. Каждому символу ставится в соответствие по крайней мере один уникальный кодовый набор (не совпадающий с кодовым набором другого символа).

3. Самосинхронизирующиеся коды

Определение 3.1. Переход $A - B$ в кодовой системе Z назовем *допустимым*, если $A, B \in Z, \quad A \neq B$ и для любого набора $C \in \{(A, B) \setminus A, B\} \quad C \notin Z.$

Определение 3.2. Кодовая система, в которой каждый возможный переход является допустимым, называется *самосинхронизирующимся кодом* (ССК).

Можно показать, что момент завершения перехода $A - B$ можно зафиксировать независимо от времени реализации перехода по самому факту установки набора B.

3.1. Коды с прямыми переходами

Рассмотрим вариант кодирования, когда

1) для каждого символа σ в явном виде задается множество $\Sigma(\sigma)$ символов, которые могут непосредственно следовать за ним;

2) один и тот же символ не может появиться в два последовательных момента времени, т.е. $\sigma \notin \Sigma(\sigma)$;

3) каждому символу ставится в соответствие единственный кодовый набор.

3. Самосинхронизирующиеся коды

Определение 3.3. Кодовая система, удовлетворяющая этим условиям, называется *кодом с прямыми переходами*.

Рассмотрим одну из возможных процедур построения таких кодовых систем.

1. Пусть $Z(A)$ – множество наборов, которые могут непосредственно следовать за А. Для каждого перехода $A - B$ и любого набора $C \in \{Z(A) \setminus B\}$ должна существовать кодирующая переменная, которая в подкубе (A,B) принимает одно значение (например,1), а на наборе C – противоположное (0). Эта переменная порождает π-разбиение $\pi = \{A, B, \overline{C}\}$.

2. Если после набора A может следовать единственный набор B, то это означает, что переход $A - B$ не накладывает ограничений на кодирование;

3. Все символы алфавита Σ должны быть закодированы различными наборами. Это означает, что для любой пары символов из Σ должна найтись кодирующая переменная, которая в наборах, соответствующих этой паре принимает различные значения (порождает π-разбиение этих наборов).

3. Самосинхронизирующиеся коды

Пример 3.1. Пусть $\Sigma = \{\alpha,\beta,\sigma,\delta,\varepsilon,\varphi\}$, $\Sigma(\alpha) = \{\beta,\sigma,\delta\}$, $\Sigma(\beta) = \{\delta,\varepsilon\}$, $\Sigma(\sigma) = \{\delta,\varphi\}$, $\Sigma(\delta) = \{\varphi\}$, $\Sigma(\varepsilon) = \{\alpha\}$, $\Sigma(\varphi) = \{\alpha,\varepsilon\}$. Результатом будут наборы кодовой системы Z = {A,B,C,D,E,F}, соответствующие символам. Тогда Z(A) ={B,C,D}, Z(B) = {D,E}, Z(C) = {D,F}, Z(D) = {F}, Z(C) = {A}, Z{F} = {A,E}.

Таблица π-разбиений

π_i	A	B	C	D	E	F
$\pi_1 = \{\overline{A,B},\overline{C}\}$	1	1	0	–	–	–
$\pi_2 = \{\overline{A,B},\overline{D}\}$	1	1	–	0	–	–
$\pi_3 = \{\overline{A,C},\overline{B}\}$	1	0	1	–	–	–
$\pi_4 = \{\overline{A,C},\overline{D}\}$	1	–	1	0	–	–
$\pi_5 = \{\overline{A,D},\overline{B}\}$	1	0	–	1	–	–
$\pi_6 = \{\overline{A,D},\overline{C}\}$	1	–	0	1	–	–
$\pi_7 = \{\overline{B,E},\overline{D}\}$	–	1	–	0	1	–

π_i	A	B	C	D	E	F
$\pi_8 = \{\overline{B,D},\overline{E}\}$	–	1	–	1	0	–
$\pi_9 = \{\overline{C,D},\overline{F}\}$	–	–	1	1	–	0
$\pi_{10} = \{\overline{C,F},\overline{D}\}$	–	–	1	0	–	1
$\pi_{11} = \{\overline{A,F},\overline{E}\}$	1	–	–	–	0	1
$\pi_{12} = \{\overline{E,F},\overline{A}\}$	0	–	–	–	1	1
$\pi_{13} = \{\overline{B},\overline{F}\}$	–	1	–	–	–	0
$\pi_{14} = \{\overline{C},\overline{E}\}$	–	–	1	–	0	–

В клетку таблицы записывается 1, если символ входит в первый блок π-разбиения, 0, если во второй, и прочерк, если он не входит ни в один блок.

3. Самосинхронизирующиеся коды

Варианты кодирования наборами минимальной длины могут быть найден с помощью следующей процедуры (метод Треси).

1. Строки из правой части таблицы доопределяем нулями и единицами всеми возможными способами.
2. Находим максимальные группы, состоящие из одинаковых и инверсных строк. В каждой группе оставляем по одной строке, а остальные удаляем.
3. Каждую из оставшихся строк сравниваем со строками таблицы разбиений и определяем все разбиения, порождаемые данной строкой.
4. Далее решается задача покрытия всех разбиений минимальным количеством строк.

Выполним эту процедуру для примера 3.1. После выполнения пунктов 1, 2 и 3 Получим следующую таблицу. Легко видеть, что отмеченные три строки покрывают все π-разбиения предыдущей таблицы. Столбцы этих трех строк определяют вариант кодирования символов следующими кодовыми наборами: A=111, B=110, C=011, D=101, E=000, F=010.

3. Самосинхронизирующиеся коды

α	β	σ	δ	ε	φ	π_i
1	1	0	0	0	0	1,2,12,13
1	1	0	1	0	0	1,6,8,10,12,13
1	1	0	0	1	0	1,2,7,13,14
1	1	0	1	1	0	1,6,10,13,14
1	1	0	0	0	1	1,2,9,11
1	1	0	1	0	1	1,6,8,11,14
1	1	0	0	1	1	1,2,7,9,14
1	1	0	1	1	1	1,6,14
1	1	1	0	0	0	2,4,12,13,14
1	1	1	0	1	0	2,4,7,13
1	1	1	0	0	1	2,4,10,11,14
1	1	1	0	1	1	2,4,7,10
1	0	1	0	0	0	3,4,12
1	0	1	1	0	0	3,5,7,9,12,14
1	0	1	0	1	0	3,4,8,14

α	β	σ	δ	ε	φ	π_i
1	0	1	1	1	0	3,5,9
1	0	1	0	0	1	3,4,10,11,13
1	0	1	1	0	1	3,5,7,11,13,14
1	0	1	0	1	1	3,4,8,10,13,14
1	0	1	1	1	1	3,5,13
1	0	0	1	0	0	3,6,7,10,12
1	0	0	1	1	0	5,6,10,13
1	0	0	1	0	1	5,6,7,11,13
1	0	0	1	1	1	5,6,13,14
1	0	0	0	1	1	8,9,13,14
1	1	1	1	0	0	8,9,12,13,14
1	0	0	0	1	0	8,14
1	1	1	1	0	1	8,11,14
1	0	0	0	0	1	9,11,13
1	1	1	1	1	0	9,13

Это один из методов противогоночного кодирования.

3. Самосинхронизирующиеся коды

Для кода с прямыми переходами все переходы осуществляются за одну фазу и поэтому этот код является *однофазным*.

Все рассматриваемые ниже системы кодирования ССК – *двухфазные*. В таких системах все множество кодирующих наборов разбивается на два непересекающихся подмножества $Z = R \cup S$. Переходы осуществляются из наборов одного подмножества в наборы другого подмножества, переходы между наборами одного подмножества запрещены.

Обычно подмножество S состоит из единственного набора, кодирующего пустой символ. Этот набор называется *разделителем*, или *спейсером*. Тогда подмножество R состоит из *несравнимых наборов*, кодирующих остальные символы и называемых *рабочими наборами*. Все рабочие наборы сравнимы со спейсером, поэтому все переходы осуществляются между сравнимыми наборами, что ведет к отсутствию функциональных состязаний.

3. Самосинхронизирующиеся коды

3.2. Парафазные коды

Пронумеруем все символы кодируемого алфавита Σ= $\{\sigma_i\}$ двоичными номерами, содержащими $n =]\log|\Sigma|[$ разрядов, где]x[означает ближайшее верхнее целое значение. Представим каждый разряд z_i двоичного кода двумя переменными кодовой системы Z: z_{2i} и z_{2i+1} , так что, если $z_i = a$, то $z_{2i} = a$, $z_{2i+1} = \overline{a}$. Легко видеть, что наборы такой кодовой системы попарно несравнимы. Добавим в кодовую систему дополнительный вектор (спейсер), состоящий из всех нулей или всех единиц. Все переходы между наборами разделяются спейсером, который сравним со всеми остальными наборами. Такая кодовая система называется *парафазным кодом* (ПК).

Нетрудно показать, что ПК относится к классу ССК. Из 2^{2n} возможных наборов он использует только $2^n + 1$. Каждую пару переменных, представляющую разряд z_i двоичного кода, будем обозначать как $(x_i, \hat{x}_i)$ при использовании нулевого спейсера или как $(x_i, \tilde{x}_i)$ при использовании спейсера из всех единиц.

Логическое Проектирование Асинхронных Схем

Лекция 2

3. Самосинхронизирующиеся коды

3.3. Код с идентификатором

Рассмотрим кодовую систему, наборы которой содержат *n основных* и *k дополнительных* разрядов, $k =]\log_2(n+1)[$. Основные разряды представляют двоичный номера кодируемых символов, а дополнительные – *коды идентификаторов*.

Кодовые наборы основных разрядов разбьем на (n+1) класс так, что к одному классу (*w-классу*) относятся наборы с одинаковым числом единиц ($0 \le w \le n$).

Построим кодовую систему, удовлетворяющую следующим требованиям:

1. Одному идентификатору ставится в соответствие единственный *w*-класс.
2. Если для одного *w*-класса требуется несколько идентификаторов, то они должны быть несравнимыми.
3. Если два набора *c* и *d* кодовой системы принадлежат разным *w*-классам и $w(c) > w(d)$, то либо двоичный код идентификатора набора *c* должен быть меньше двоичного кода идентификатора *d*, либо должно выполняться $v(c) < v(d)$, где $v(a)$, число единиц в идентификаторе набора *a*.

3. Самосинхронизирующиеся коды

Легко видеть, что в такой кодовой системе любые два набора несравнимы и значит она принадлежит классу ССК. Назовем ее ***кодом с идентификатором*** (КИ).

Все варианты назначения идентификаторов в наборах КИ определяются условиями 1 – 3. Варианты КИ отличаются сложностью кодирования и декодирования. Наиболее простое кодирование и декодирование обеспечивает ***код Бергера***, в котором идентификатор является двоичным эквивалентом числа нулей в основных разрядах.

Пример 3.2. Код Бергера для передачи байта информации.

w	x_9	x_{10}	x_{11}	x_{12}
0	0	0	0	1
1	1	1	1	0
2	0	1	1	0
3	1	0	1	0
4	0	0	1	0

w	x_9	x_{10}	x_{11}	x_{12}
5	1	1	0	0
6	0	1	0	0
7	1	0	0	0
8	0	0	0	0

Для $n = 8$ $k =]\log_2 9[= 4$

$n + k = 12$

Заметим, что из 16 комбинаций идентификатора задействовано только 9.

3. Самосинхронизирующиеся коды

Вообще говоря, с помощью наборов КИ длиной ($n + k$) можно передать $2^{n+1} - 1$ наборов двоичного кода, однако при этом возрастает сложность кодирования и декодирования.

Для организации двухфазной передачи КИ один из дополнительных наборов назначается спейсером, в качестве которого удобно выбрать набор из всех нулей в основных разрядах и всех единиц в идентификаторе.

3.4. Оптимальный равновесный код (код Спернера)

Оптимальный равновесный код (ОРК) обладает наименьшей избыточностью среди всех ССК.

Кодовая система называется ***равновесной***, если для кодирования рабочих символов используются коды, содержащие фиксированное число единиц.

Из множества n-разрядных двоичных наборов можно выделить максимум C_n^k наборов, каждый из которых содержит k единиц, $0 \le k \le n$.

3. Самосинхронизирующиеся коды

Пусть требуется закодировать N различных символов, тогда длину n кодового слова можно найти $C_n^k \geq N$, $0 \leq k \leq n$. Максимальное значение C_n^k достигается при $k =]n/2[= [n/2]$. Обозначим $\max\limits_k C_n^k = C_n^{n/2}$ и выберем такое n, что

$$C_{n-1}^{(n-1)/2} \leq N \leq C_n^{n/2}.$$

Равновесный код с весом $k = n/2$ назовем ***оптимальным равновесным кодом*** (ОРК).

Теорема 3.1. Наборы ОРК имеют минимальную длину по всем кодовым системам, из класса ССК. (Доказательство можно найти в [2].)

При передаче наборов ОРК в качестве спейсера обычно выбирается набор из всех нулей или набор из всех единиц.

Все другие известные кодовые системы, относящиеся к классу ССК, являются комбинациями рассмотренных кодовых систем, не отличаются простотой кодирования и декодирования, используют частные свойства множеств кодовых наборов и, следовательно, не являются универсальными.

3. Самосинхронизирующиеся коды

3.5. Об избыточности кодовых систем

Сравним избыточности ПК, КИ и ОРК для случая кодирования информации, представленной двоичным позиционным кодом длины n.

Избыточность кодовых систем оценивается по формуле $R = 1 - \dfrac{\log_2 N_0}{\log_2 N}$,

Где N_0 – число используемых кодовых наборов, а N – общее число возможных наборов.

Для ПК длина кодовых наборов $l = 2n$ и

$$R_{\text{ПК}} = 1 - \frac{\log_2 2^n}{\log_2 2^{2n}} = \frac{2n - n}{2n} = 0.5.$$

Для КИ длина кодовых наборов $l = n +]\log_2(n+1)[$,

$$R_{\text{КИ}} = 1 - \frac{\log_2 2^n}{\log_2 2^{n+]\log_2(n+1)[}} = \frac{]\log_2(n+1)[}{n+]\log_2(n+1)[} = \frac{l-n}{l}.$$

3. Самосинхронизирующиеся коды

Для ОРК длина наборов удовлетворяет неравенствам $C_{l-1}^{(l-1)/2} < 2^n < C_l^{l/2}$

или $$\log_2 C_{l-1}^{(l-1)/2} < n < \log_2 C_l^{l/2}.$$

Очевидно, что $$R_{\text{ОРК}} = 1 - \frac{\log_2 2^n}{\log_2 2^l} = \frac{l-n}{l}.$$

Для больших n воспользуемся приближенной формулой Стирлинга для $l!$

$$l! \approx l^l e^{-l} \sqrt{2\pi},$$

Откуда $$C_l^{l/2} = \frac{l!}{(\frac{l}{2})!(\frac{l}{2})!} \approx \frac{2^{l+1}}{\sqrt{2\pi l}},$$

$$C_{l-1}^{(l-1)/2} = \frac{(l-1)!}{(\frac{l-1}{2})!(\frac{l-1}{2})!} \approx \frac{2^l}{\sqrt{2\pi(l-1)}}.$$

Таким образом,

$$l - \log_2 \sqrt{2\pi(l-1)} < n < l + 1 - \log \sqrt{2\pi l}$$

и $$\frac{\log\sqrt{2\pi l}-1}{l} < R_{\text{ОРК}} < \frac{\log_2\sqrt{2\pi(l-1)}}{l}.$$

Рассчеты показывают, что для небольших значений n ($2 \le n \le 16$)

$$l_{\text{ОРК}} \le n +]\log_2 n[\quad \text{и} \quad R_{\text{ОРК}} \le R_{\text{КИ}} \le R_{\text{ПК}}.$$

3.6. Передача наборов ССК кодами в изменениях

Недостатком двухфазных кодовых систем ПК, КИ и ОРК является то, что рабочие наборы перемежаются спейсером. (В иностранной литературе такие ССК называют четырехфазными, так как передача одного кодового набора в системах с хендшейковым взаимодействием осуществляется последовательным выполнением четырех действий: <установка рабочего набора – получение ответа – установка спейсера – получение ответа>.)

При длинных интерфейсных связях желательно исключить передачу спейсера, что вдвое увеличивает скорость передачи. Такое исключение возможно, если использовать передачу наборов ССК *в изменениях*.

3. Самосинхронизирующиеся коды

В этом случае в качестве носителя информации используются не сами значения сигналов, а их изменения (бит информации равен 1 при наличии изменения и 0 при его отсутствии).

Рассмотрим передачу в изменениях n-разрядных кодовых наборов ССК по шине, содержащей n информационных линий связи и одну дополнительную линию сигнала ответа.

Пусть $A=a_1,a_2,...,a_n$ – набор значений потенциалов на информационных линиях передатчика и требуется передать рабочий набор ССК $X=x_1,x_2,...,x_n$. Тогда в следующий момент времени на шине устанавливаются значения потенциалов $B=b_1,b_2,...,b_n$ такие, что $b_i = a_i \oplus x_i,\ i = 1,2,...,n.$

Приемник хранит набор сигналов A, получает набор B, и восстанавливает на своих выходах набор ССК X, $x_i = a_i \oplus b_i,\ i = 1,2,...,n.$ После того, как набор X будет использован, приемник осуществляет смену набора A на набор B, после чего на выходах приемника устанавливается спейсер. После использования спейсера приемник отвечает передатчику изменением значения сигнала ответа. Передатчик, получив сигнал ответа, инициирует следующий цикл передачи.

4. Самосинхронные схемы

Рассмотрим подходы к построению схем конечных автоматов, функционирование которых инвариантно к задержкам элементов. Такие схемы называют ***самосинхронными*** (или *speed independent*).

4.1. Двухфазная реализация конечного автомата

Установим следующие ограничения на реализацию конечного автомата.

1. Работа автомата осуществляется в две фазы, условно называемые *рабочей* и *подготовительной* (*нерабочей*).

2. Взаимодействие с внешней средой организуется по принципу хендшейка (запрос – ответ).

3. Символы входного и выходного алфавитов кодируются рабочими кодовыми наборами двухфазных ССК. Рабочие наборы чередуются со спейсером.

4. Множество состояний автомата делится на два подмножества (*рабочие состояния* и *нерабочие состояния*).

4. Самосинхронные схемы

5. *Используемая гипотеза о характере задержек:*

а) величины задержек элементов являются произвольными, но конечными;

б) задержки элементов могут быть как инерциальными, так и чистыми;

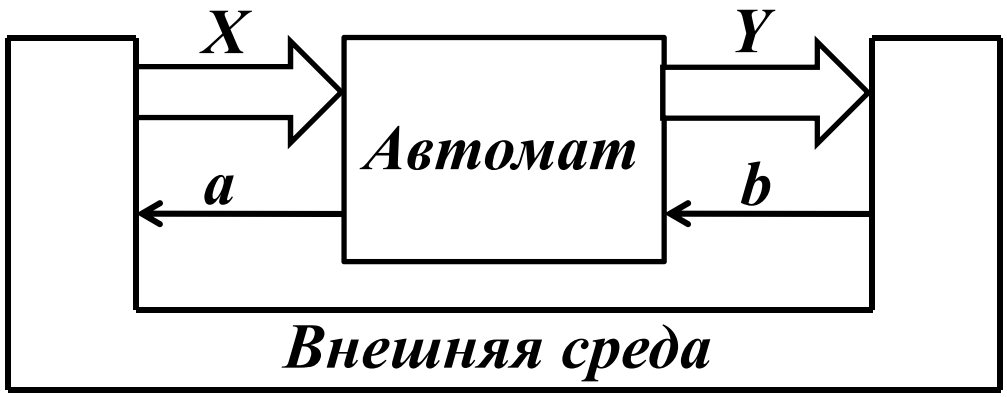
в) задержка в проводе до разветвления приводится к задержке элемента, работающего на провод; задержки проводов после разветвления приводятся ко входам нагрузочных элементов и их разность не должна превосходить минимальной задержки элемента.

6. Дополнительное требование на реализацию: *применение встроенных задержек недопустимо.*

Реализация автомата, удовлетворяющую перечисленным требованиям будем называть *двухфазной* (в англ. терминологии - *fundamental mode*).

Согласованная реализация. Двухфазная реализация системы «автомат – внешняя среда», в которой взаимодействие автомата и среды организовано по принципу «запрос- ответ» (хендшейк), называется *согласованной реализацией.*

4. Самосинхронные схемы



Сигналы ***a*** *и* ***b*** – *фазовые сигналы*, вызывающие смену фаз во внешней среде и в автомате;

a – сигнал запроса от автомата на смену фазы среды, или сигнал ответа в среду, формируемый индикатором окончания переходных процессов а автомате;

b – сигнал запроса от среды на смену фазы автомата, или сигнал ответа от среды, формируемый индикатором окончания переходных процессов в среде;

X и ***Y*** – наборы ССК, которые тоже можно рассматривать как сигналы запроса и ответа.

4. Самосинхронные схемы

4.2. Индикаторы и тестеры

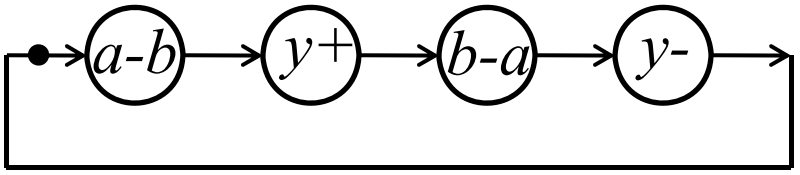
Индикатор – обязательная принадлежность согласованных реализаций. Изменение сигнала на его выходе свидетельствует об окончании соответствующей фазы переходного процесса.

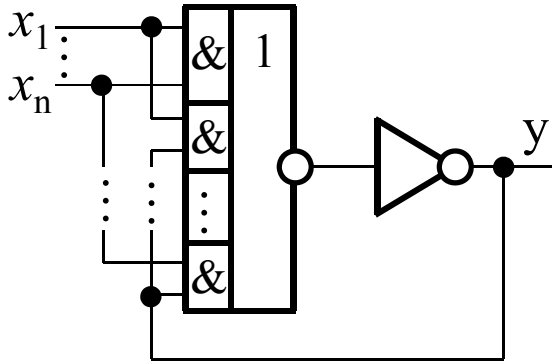
К индикаторам относятся и так называемые *тестеры*, предназначенные для определения принадлежности кодового набора ССК или спейсеру.

Простейший тестер, отличающий только два набора, состоящих из всех нулей (набор *a*) и всех единиц (набор *b*), реализует автоматное уравнение

$$y = \&_{i=1}^{n} x_i \vee y(\mathrm{V}_{i=1}^{n} x_i)$$

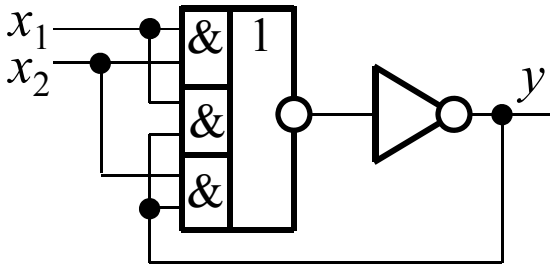
и называется *C*-элементом.

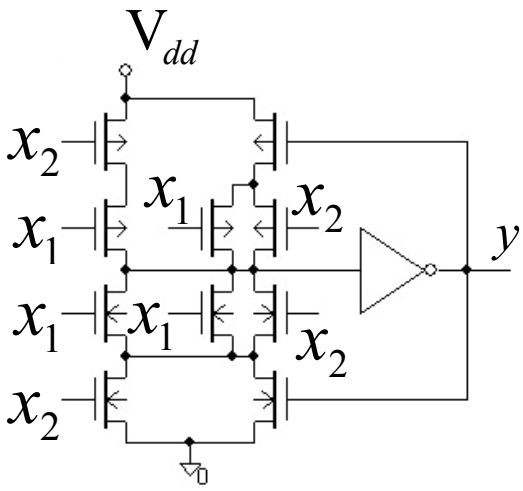


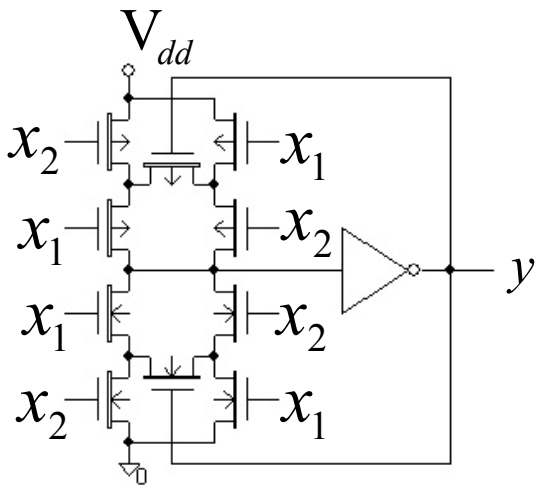


4. Самосинхронные схемы

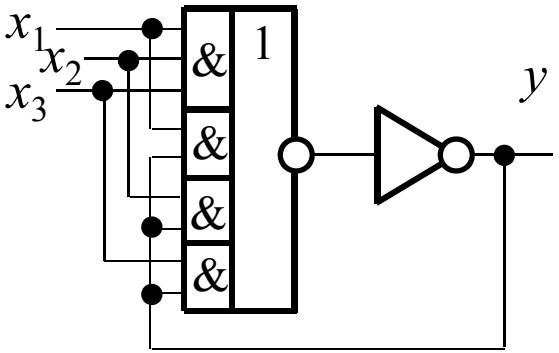
Двухвходовый С-элемент Маллера $y = x_1x_2 \vee y(x_1 \vee x_2)$

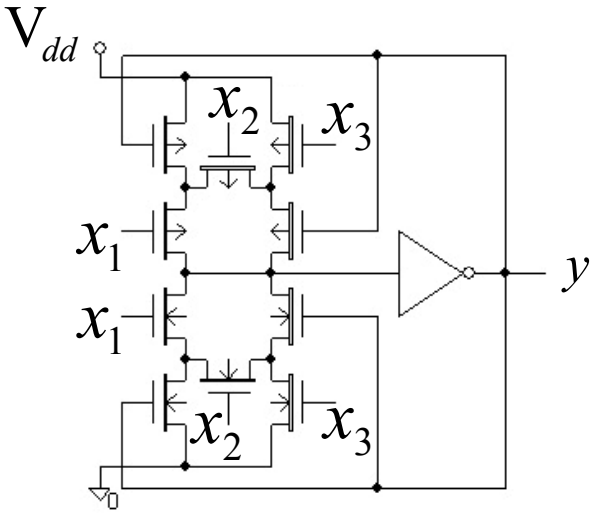






Трехвходовый С-элемент Маллера $y = x_1x_2x_3 \vee y(x_1 \vee x_2 \vee x_3)$





4. Самосинхронные схемы

Реализация С-элемента с использованием слабых транзисторов

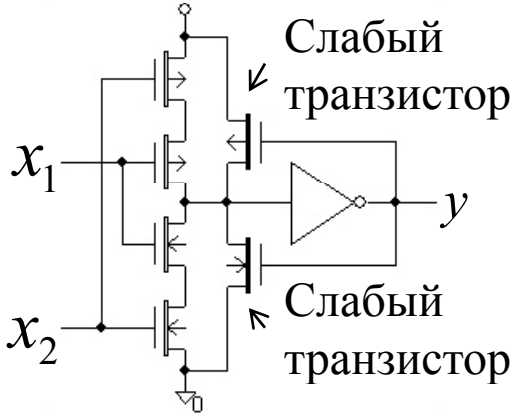
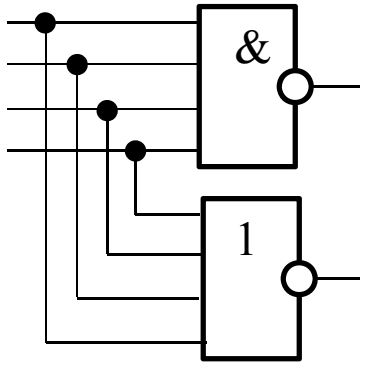


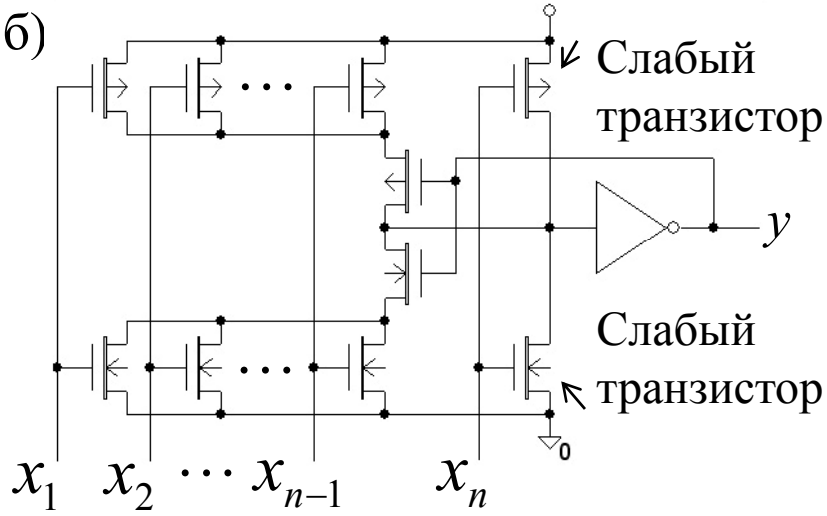
Схема параллельного сжатия числа индицируемых выходов «4 в 2»



Реализации многовходовых С-элементов

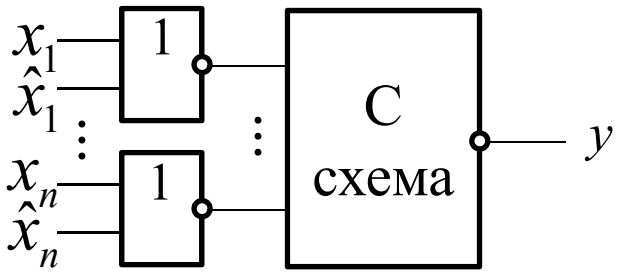
а)

б)

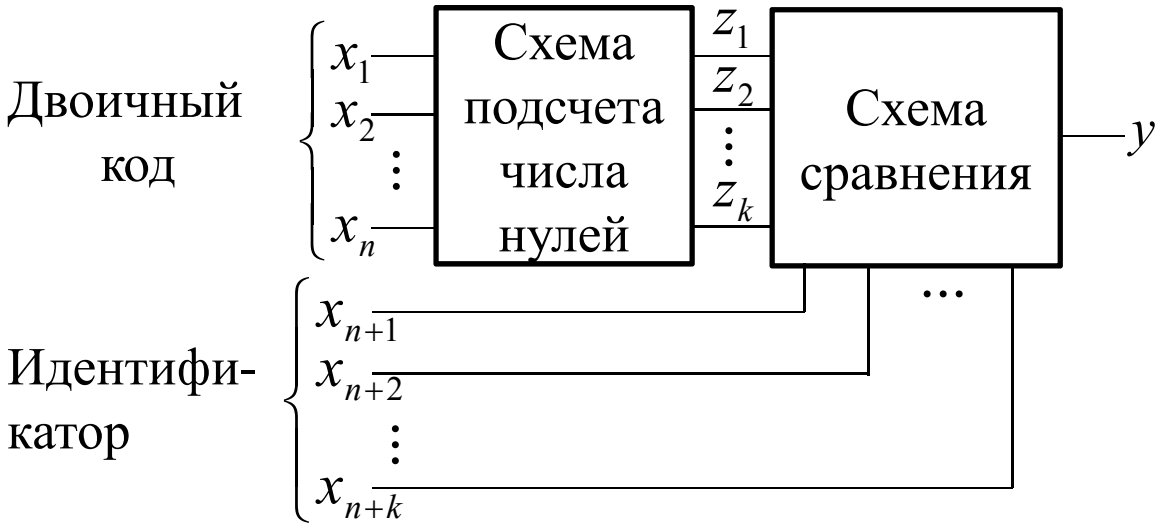


4. Самосинхронные схемы

Тестер ПК



Тестер кода Бергера



Тестеры других КИ могут оказаться намного более сложными.

4. Самосинхронные схемы

Тестеры ОРК

Тестер оптимального равновесного кода описывается автоматным уравнением

$$y = \mathrm{maj}(x_1, x_2, ..., x_n) \vee y(\bigvee_{i=1}^{n} x_i) = S \vee y\overline{R},$$

где $\mathrm{maj}(x_1, x_2, ..., x_n)$ – мажоритарная функция, содержащая $C_n^{]n/2[}$ термов ранга]n/2[.

Тестер многоразрядного ОРК может быть построен с помощью самосинхронной реализации логических функций S и R.

Приведенные конструкции индикаторов и тестеров являются базовыми при построении любых индикаторов.

4. Самосинхронные схемы

4.3. Синтез комбинационных схем

Рассмотрим способы построения двухфазных самосинхронных схем, реализующих булевы функции. Их поведение должно быть инвариантно к величинам задержек логических элементов.

Очевидно, что такие схемы должны быть избыточными, хотя бы в силу использования избыточного кодирования входных и выходных переменных наборами ССК.

Пусть необходимо построить самосинхронную схему, реализующую систему из m булевых функции $Y = \{y_1, y_2, ..., y_m\}$ n переменных $X = \{x_1, x_2, ..., x_n\}$, используя набор базисных элементов с собственными функциями

$$z_j = f_j\{x_1, x_2, ..., y_k\},\ k < n.$$

Во избежание функциональных и логических состязаний приведем все функции к супермонотонному виду. Для этого все входные переменные представим в парофазном коде $(X, \hat{X})$, $X = \{x_1, x_2, ..., x_n\}$, $\hat{X} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_n\}$ и используем двухфазную реализацию с нулевым спейсером.

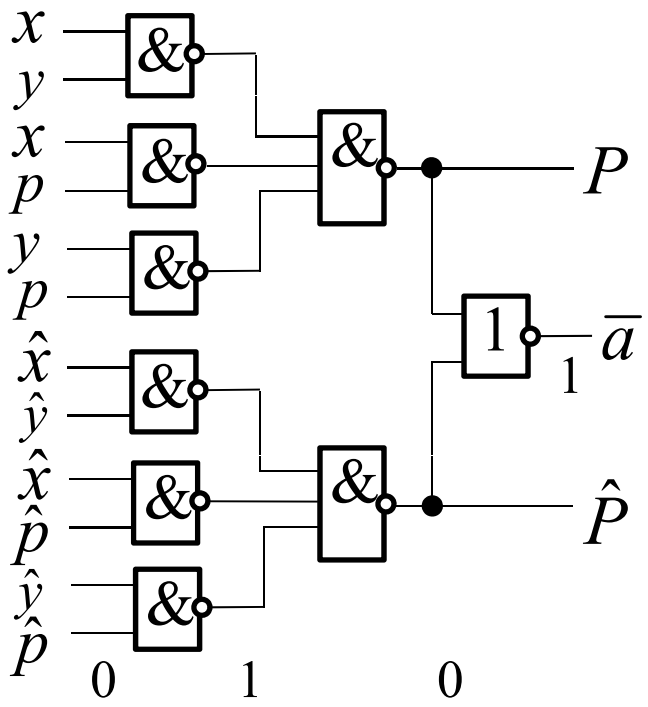
4. Самосинхронные схемы

Выходы схемы тоже будем представлять парофазным кодом $(Y, \hat{Y})$

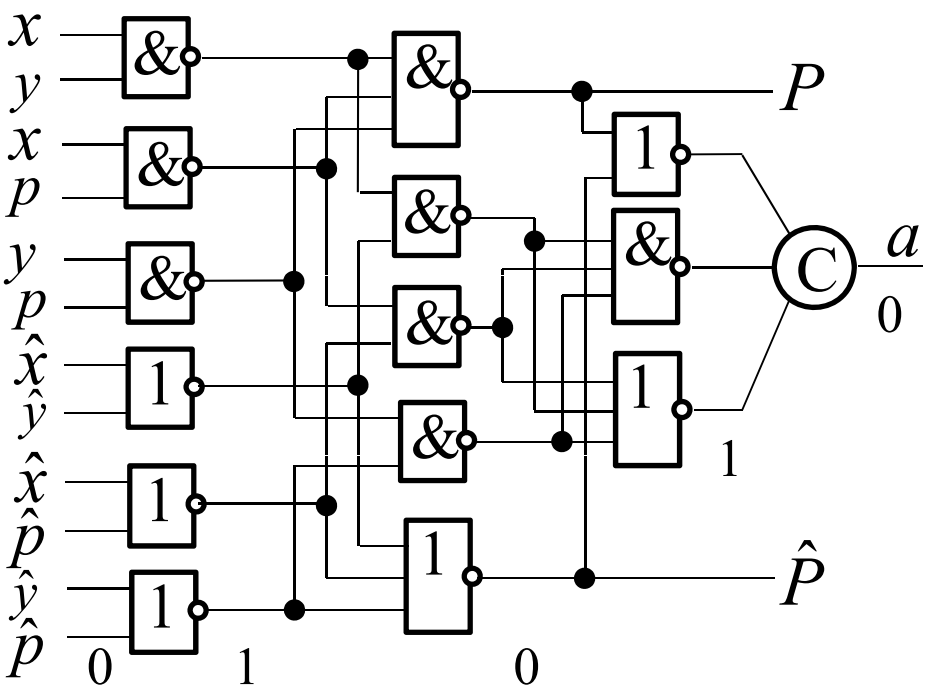
$$Y = \{y_1, y_2, ..., y_m\},\ \hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_m\}.$$

Пример 4.1. Пусть требуется построить самосинхронную схему, реализующую функцию переноса одноразрядного сумматора $P = xy \vee xp \vee yp$.

а) Некорректная реализация



б) Корректная реализация



4. Самосинхронные схемы

4.3.1. Индицируемость

Пусть на входах схемы $Y = F(X)$ осуществляется допустимый переход $a - b$, инициирующий на выходах Y допустимый переход $k - l$. Если для любого набора $t \in [a, b]$ выполняется $F(t) \neq l$, то будем говорить, что на переходе $a - b$ входные переменные схемы, изменяющие свои значения, *индицируются* на выходах Y. В случае, когда это выполняется на всех допустимых переходах, будем говорить, что *входы X схемы индицируются на ее выходах Y*.

Схема является *самосинхронной*, если выходные сигналы всех составляющих ее элементов *индицируемы* на ее выходах. При этом входные сигналы схемы являются внешними по отношению к ней и схема может не индицировать установку на ее входах наборов парафазного кода. Для фиксации их установки необходим тестер парафазного кода.

Таким образом, для того, чтобы схема, использующая двухфазную дисциплину смены входных наборов парофазного кода, была самосинхронной, необходимо и достаточно выполнение трех условий:

4. Самосинхронные схемы

1. Реализуемая функция представляется двумя супермонотонными функциями (прямой и инверсной).
2. Рабочие наборы выходных сигналов каждого яруса реализации этих функций должны быть наборами ССК с нулевым или единичным спейсером.
3. Выходные сигналы всех элементов схемы, изменяющие свои значения в процессе функционирования, должны быть индицируемы.

4.3.2. Стандартные реализации

Поскольку процедура анализа схем на индицируемость всех составляющих ее элементов достаточно трудоемка, имеет смысл рассмотреть *канонические приемы* синтеза самосинхронных комбинационных схем, приводящие к самосинхронным реализациям. Такие реализации будем называть *стандартными*.

4. Самосинхронные схемы

Реализация по минимальным формам БФ. Функция задается в минимальной форме ДНФ или КНФ. Предполагается двухканальная реализация. В первом канале реализуется сама функция, а в другом ее инверсия таким образом, чтобы элементы ее составляющие, были двойственны элементам первого канала. Тогда выходные сигналы элементов каждого яруса реализации функций будут представлены парофазным кодом.

Будем называть такую реализацию *перекрестной*, поскольку она допускает использование элементов первого канала во втором и наоборот. Для выходных сигналов каждого яруса, неиндицируемых элементами следующего яруса, следующий ярус дополняется элементами индикации парофазного кода.

Пояснения к **примеру 4.1,б**. Реализации функций двух каналов:

$$P = xy \vee xp \vee yp = \overline{\overline{xy} \cdot \overline{xp} \cdot \overline{yp}}, \quad \hat{P} = \overline{xy \vee xp \vee yp} = \overline{\overline{\hat{x}} \vee \overline{\hat{y}} \vee \overline{\hat{x}} \vee \overline{\hat{p}} \vee \overline{\hat{y}} \vee \overline{\hat{p}}}.$$

Во второй ярус добавлены 3 индикатора парофазного кода. Третий ярус состоит из индикатора выходов $P, \overline{P}$ и схемы параллельного сжатия «3 в 2». Фазовый сигнал a вырабатывается С-элементом.

4. Самосинхронные схемы

Реализация по ортогональным формам. В ортогональном представлении функций пары каналов только один из термов может принимать значение 1. Если каждый терм может быть реализован одним элементом базиса, то это существенно упрощает реализацию, так как наборы этого кода индицируются на выходах схемы.

Пример 4.1 (продолжение). Ортогональное представление пары функций переноса одноразрядного сумматора имеет вид:

$$P = xy \vee x\hat{y}p \vee \hat{x}yp, \quad \hat{P} = \hat{x}\hat{y} \vee \hat{x}y\hat{p} \vee x\hat{y}\hat{p}.$$

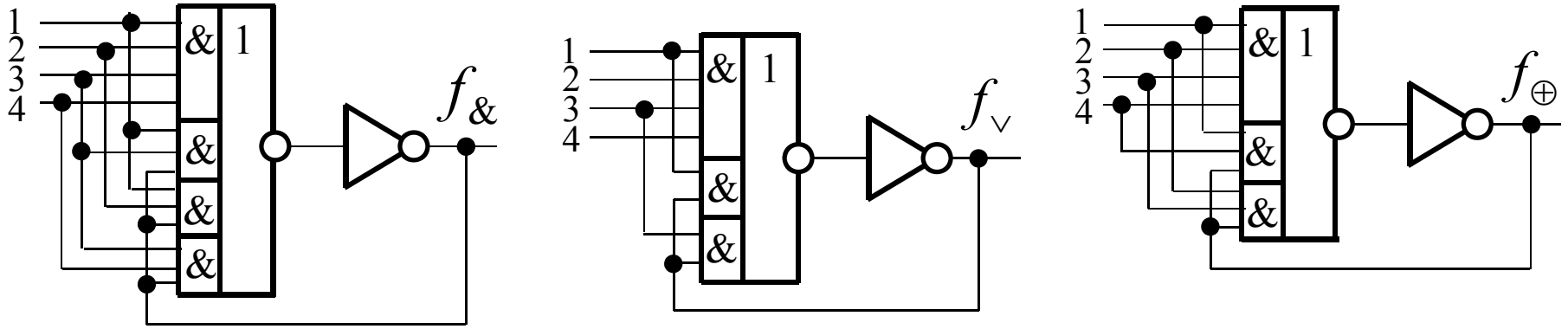
Эти функции реализуются самосинхронной схемой из 9 элементов:

$$P = \overline{\overline{xy} \cdot \overline{x\hat{y}p} \cdot \overline{\hat{x}yp}}, \quad \hat{P} = \overline{\overline{\hat{x}\hat{y}} \cdot \overline{\hat{x}y\hat{p}} \cdot \overline{x\hat{y}\hat{p}}}, \quad a = \overline{P \vee \hat{P}}$$

Реализация на обобщенных С-элементах. Такие элементы совмещают реализацию базисных функций с индикаторами входных сигналов и могут быть построены в базисе, состоящим из элементов И-ИЛИ-НЕ и инвертора. Двухфазные схемы, построенные из таких элементов, являются самосинхронными и индицируют входные наборы (не требуют тестера).

4. Самосинхронные схемы

а*) *Элемент И ***б*) Элемент *ИЛИ*** ***в*) *Элемент* ⊕**



$$а)\quad f_{\&} = x_1\tilde{x}_1x_2\tilde{x}_2 \vee f_{\&}(x_1x_2 \vee x_1\tilde{x}_1 \vee x_2\tilde{x}_2)$$

$$б)\quad f_{\vee} = x_1\tilde{x}_1x_2\tilde{x}_2 \vee f_{\vee}(x_1 \vee x_2)$$

$$в)\quad f_{\oplus} = x_1\tilde{x}_1x_2\tilde{x}_2 \vee f_{\oplus}(x_1\tilde{x}_2 \vee \tilde{x}_1x_2)$$

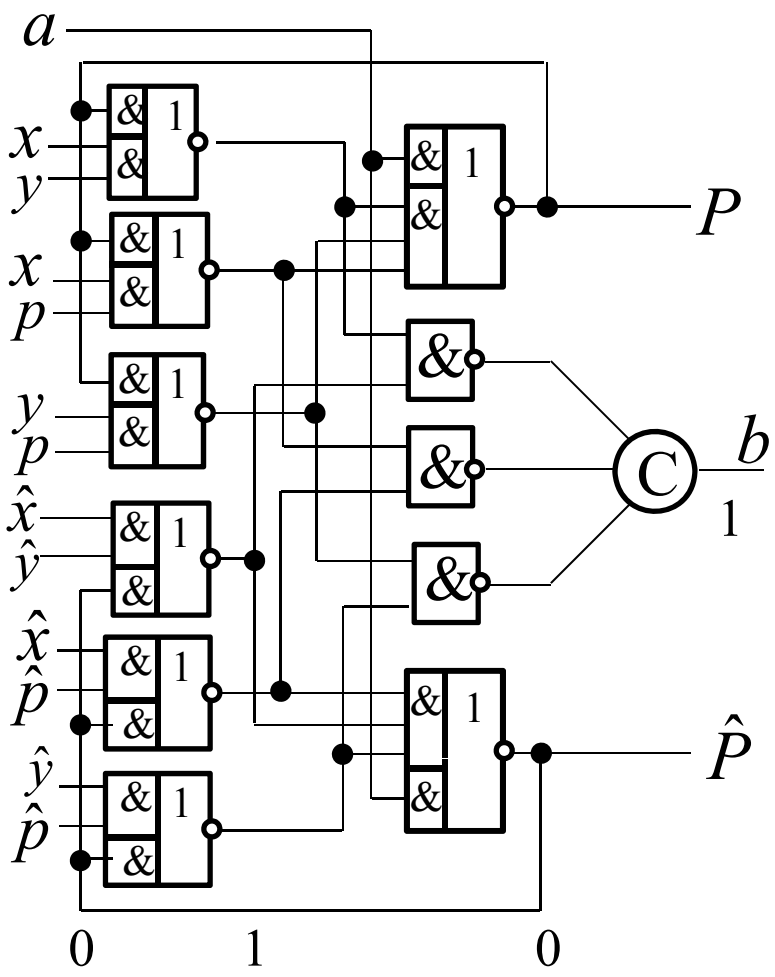
Эти схемы используют спейсер из всех единиц. Для элемента ***И*** выход равен 1, при появлении спейсера и сохраняет значение 1 при $x_1 = x_2 = 1,\ \tilde{x}_1 = \tilde{x}_2 = 0.$ Выход элемента ИЛИ равен 1 на спейсере и при $x_1 \vee x_2 = 1.$ Выход элемента ⊕ равен 1 на спейсере и сохраняет это значение при $x_1 = \tilde{x}_2 = 1$ или $\tilde{x}_1 = x_2 = 1.$

4. Самосинхронные схемы

Реализация «с коллективной ответственностью». Вернемся к примеру на слайде 19 (*a*).

Некорректную реализацию переноса, можно привести к корректной, если ввести обратные связи так, как это показано на рисунке. Некорректность возникает из-за того, что среди элементов первого яруса срабатывает либо один элемент, либо три.

В схеме на рисунке всегда срабатывают либо верхние 3 элемента первого яруса, либо 3 нижних элемента. В результате выходные сигналы элементов первого яруса представлены парофазным кодом.

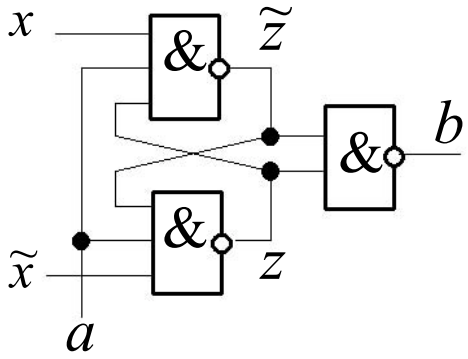


4. Самосинхронные схемы

4.4. Самосинхронные элементы памяти

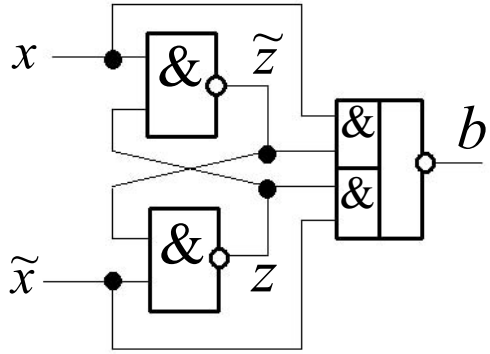
4.4.1. Простые триггеры типа «защелка»

а) С фазовым сигналом



$$(x \neq \tilde{x}) \rightarrow a^{+} \rightarrow (z \neq \tilde{z}) \rightarrow b^{+}$$
$$b^{-} \leftarrow (z = \tilde{z} = 1) \leftarrow a^{-} \leftarrow (x = \tilde{x} = 1)$$

б) Сравнение со значениями установочных входов

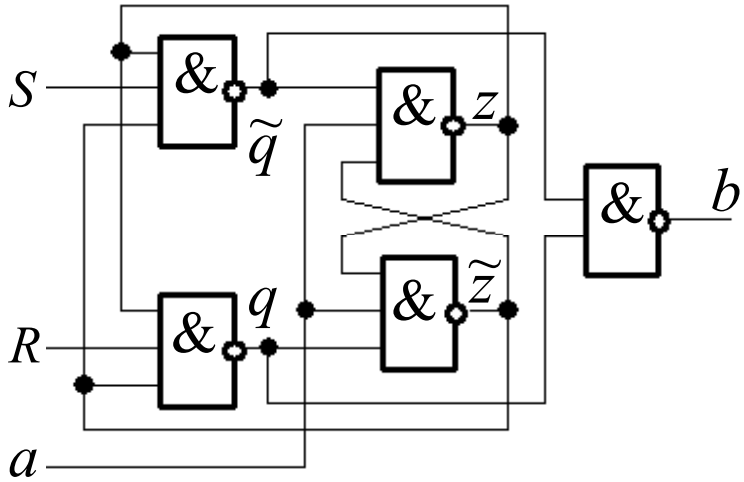


$$(x \neq \tilde{x}) \rightarrow (\tilde{z} = \tilde{x}, z = x) \rightarrow b^{+}$$
$$b^{-} \leftarrow (x = \tilde{x} = 1)$$

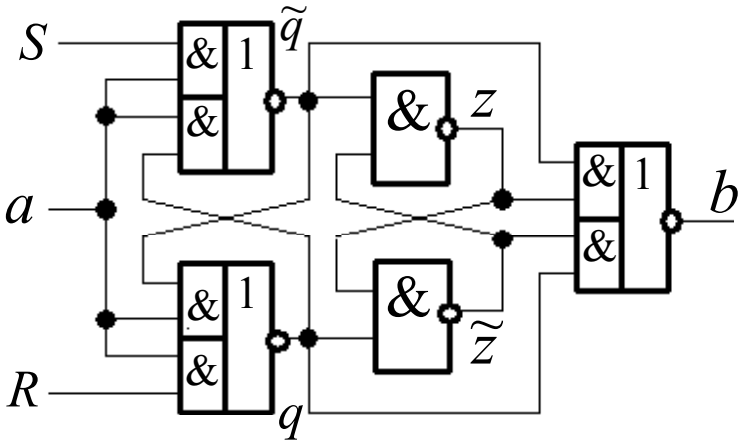
4. Самосинхронные схемы

4.4.2. Простые *RS*-триггеры

а) Схема с разрушением информации и отсечкой от входов



б) Схема без разрушения информации с индикатором Маллера



Цикл работы схемы (*а*) равен 6τ (по 3τ в обеих фазах), а для схемы (*б*) длительность цикла зависит от информации и составляет либо $2\tau + 2\tau = 4\tau$, либо $4\tau + 2\tau = 6\tau$.

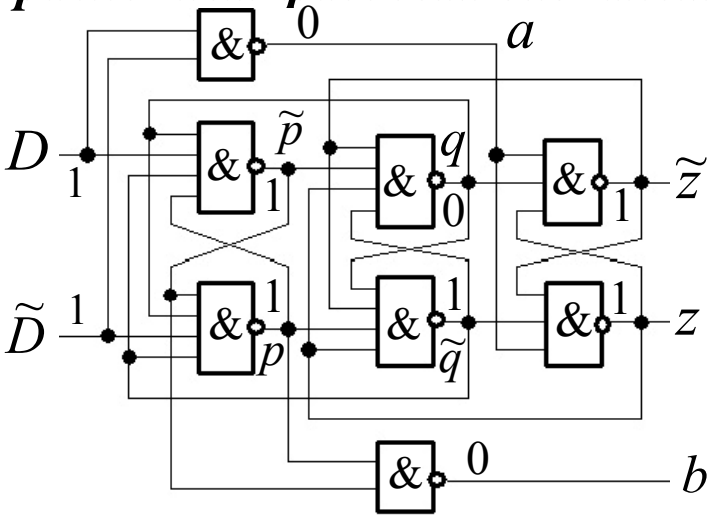
Нетрудно построить двойственные схемы простых триггеров и схемы из комплексных элементов.

4. Самосинхронные схемы

4.4.2. *D*-триггеры

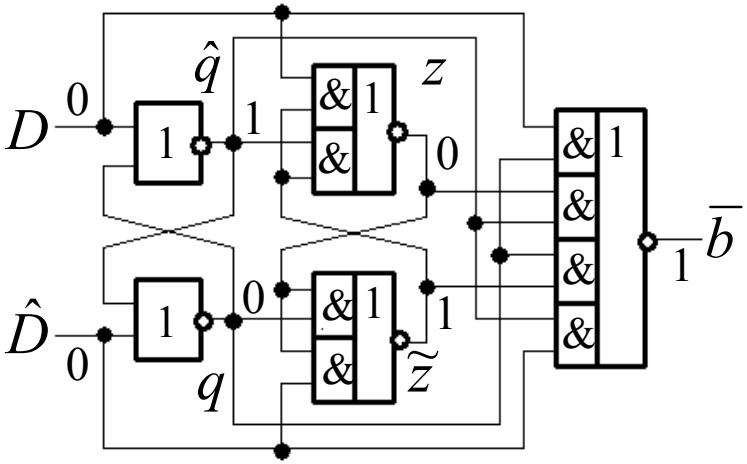
D-триггер задерживает сигнал на один такт. В нашем случае это соответствует изменению состояния триггера во второй фазе.

а) Схема из простых элементов с управлением фазовым сигналом



Триггер непригоден для построения автоматов, так как при переходе из спейсера в рабочую фазу информация перемещается из триггера *q* в триггер z.

б) Схема из комплексных элементов без перемещения информации

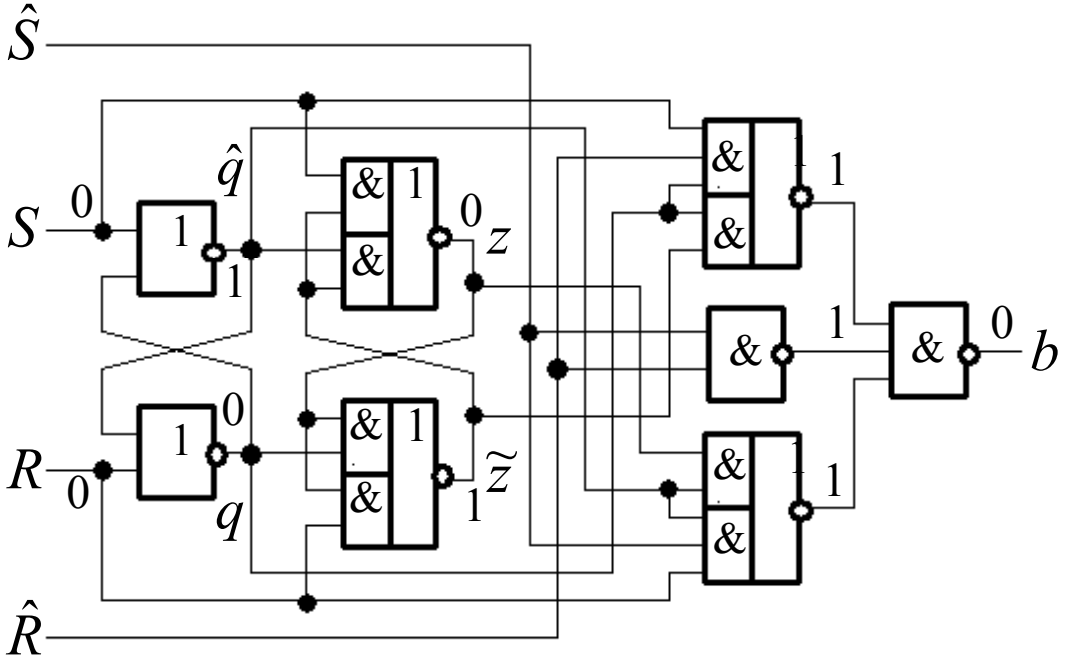


Может быть использована для построения автоматов.

$$\bar{b} = \overline{Dq \vee \hat{q}z \vee q\tilde{z} \vee \hat{D}\hat{q}}$$

4. Самосинхронные схемы

4.4.3. Двухрегистровый *RS*-триггер



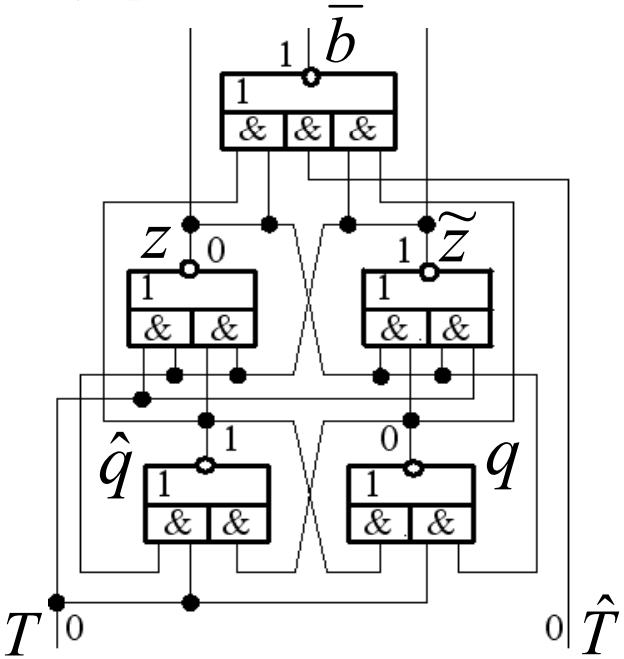
$$b = \overline{\overline{S\hat{R}q \vee q\tilde{z}} \cdot \overline{\hat{S}\hat{R}} \cdot \overline{\hat{S}R\hat{q} \vee \hat{q}z}}$$

Состояние триггера снимается с выходов z, $\tilde{z}$. Сигналы с этих выходов не имеют спейсера и переключаются через промежуточное состояние 11. Вместо спейсера используется фазовый сигнал который блокирует все входы комбинационной схемы. Необходим тестор установки спейсера на входах.

4. Самосинхронные схемы

4.4.4. Счетные триггеры

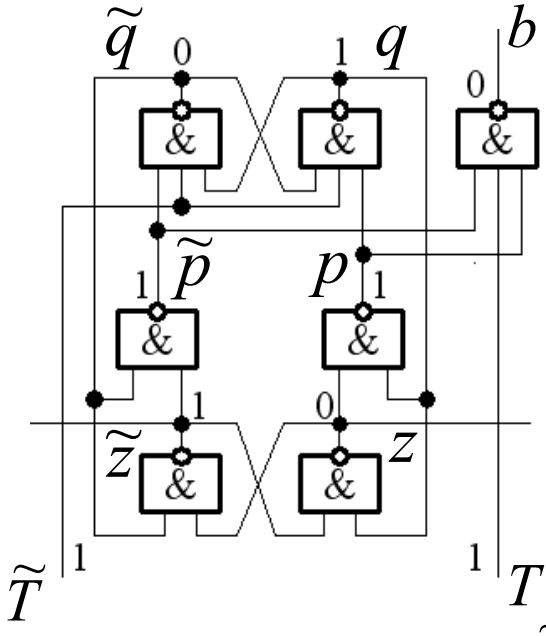
а) Триггер с разнополярным управлением



$$\overline{b} = \overline{\hat{q}z \vee q\tilde{z} \vee \hat{T}}$$

$$D_T = 3\tau + 3\tau$$

б) Триггер из простых элементов



$$D_{\tilde{T}=1} = 4\tau + \tau$$

$$D_{\tilde{T}=0} = 2\tau + \tau$$

$$\rightarrow -\tilde{T} \rightarrow +\tilde{q} \rightarrow -\tilde{p} \rightarrow +b \rightarrow +\tilde{T} \rightarrow -q \rightarrow +z$$

$$-p \leftarrow +q \leftarrow -\tilde{T} \leftarrow -b \leftarrow +\tilde{p} \leftarrow -\tilde{z} \leftarrow$$

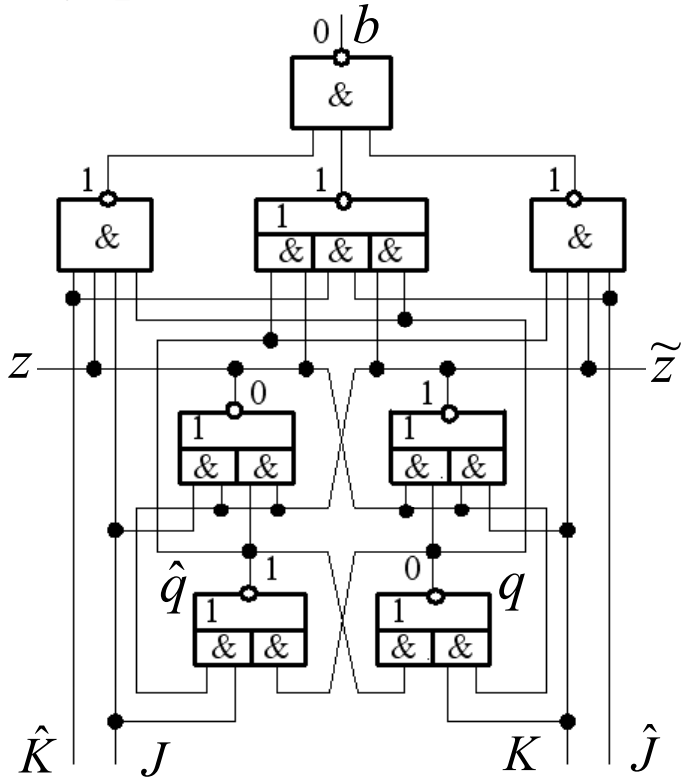
$$+b \rightarrow +\tilde{T} \rightarrow -\tilde{q} \rightarrow +\tilde{z} \rightarrow -\tilde{z} \rightarrow +p \rightarrow -b$$

$$\rightarrow -T \rightarrow +b \rightarrow +T \rightarrow -b$$

4. Самосинхронные схемы

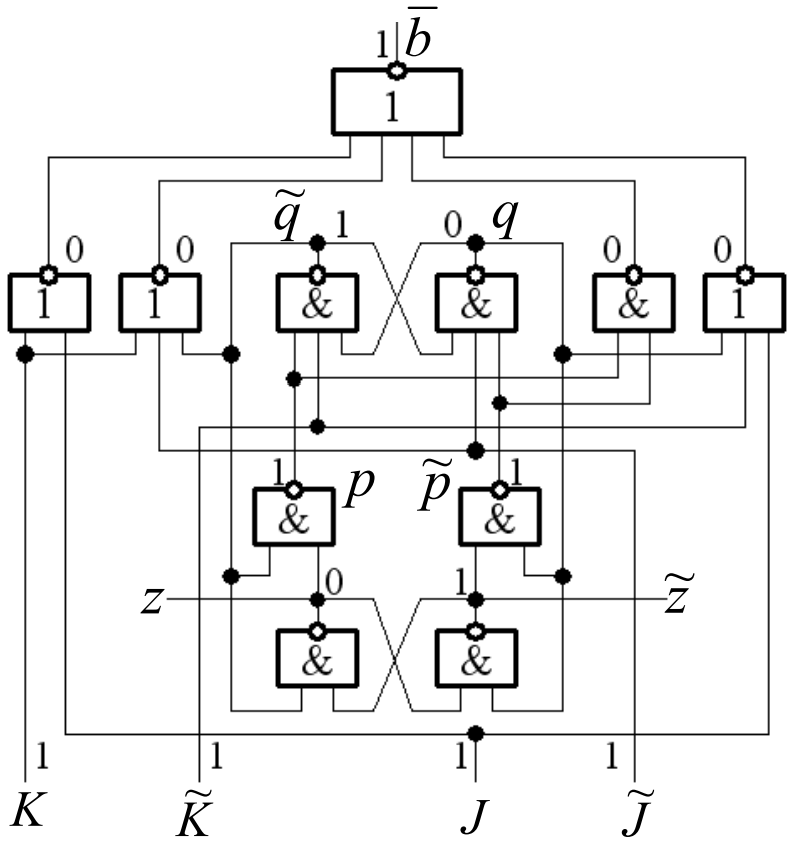
4.4.5. *JK*-триггеры

а) Триггер с разнополярным управлением



$$b = \overline{J\hat{K}qz \cdot \hat{q}z} \vee \overline{qz} \vee \overline{\hat{J}\hat{K} \cdot \hat{J}K\hat{q}\tilde{z}}$$

б) Триггер из простых элементов



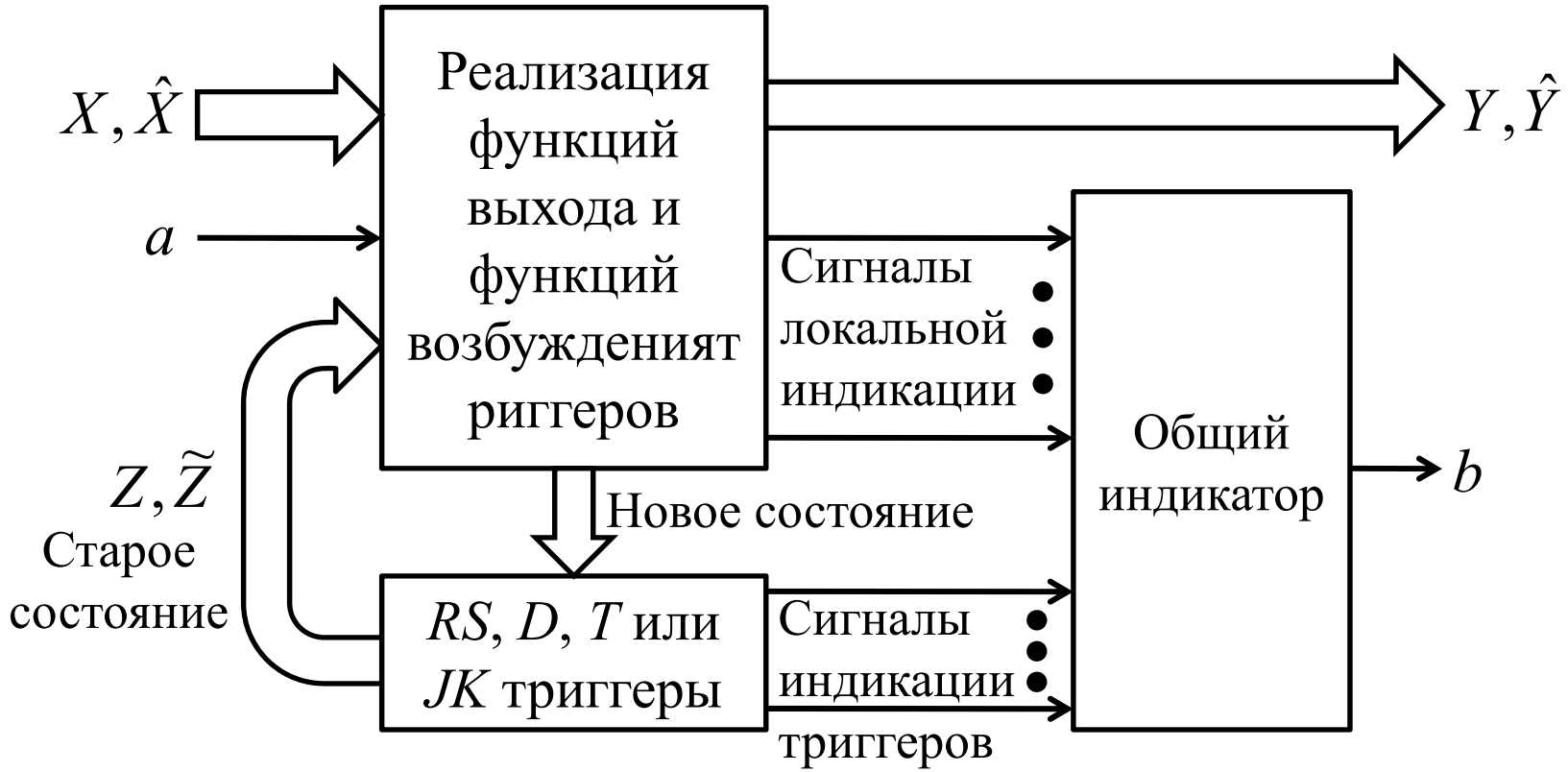
$$\bar{b} = \overline{p\tilde{p}} \vee \overline{q \vee J \vee \tilde{K}} \vee \overline{\tilde{q} \vee \tilde{J} \vee K} \vee \overline{J \vee K}$$

Необходимы тесторы установки спейсера на входах.

4. Самосинхронные схемы

4.5. Стандартная реализация самосинхронных автоматов

Структурная схема самосинхронного автомата



Фазовый сигнал $a = 0$ блокирует входы $Z, \widetilde{Z}$ (равносилен нулевому спейсеру).

4. Самосинхронные схемы

Стандартная двухфазная реализация самосинхронного автомата значительно упрощает процедуру синтеза ценой увеличения аппаратурных затрат. Ее достоинством является отсутствие состязаний и возможность неизбыточного кодирования внутренних состояний (нет проблемы гонок).

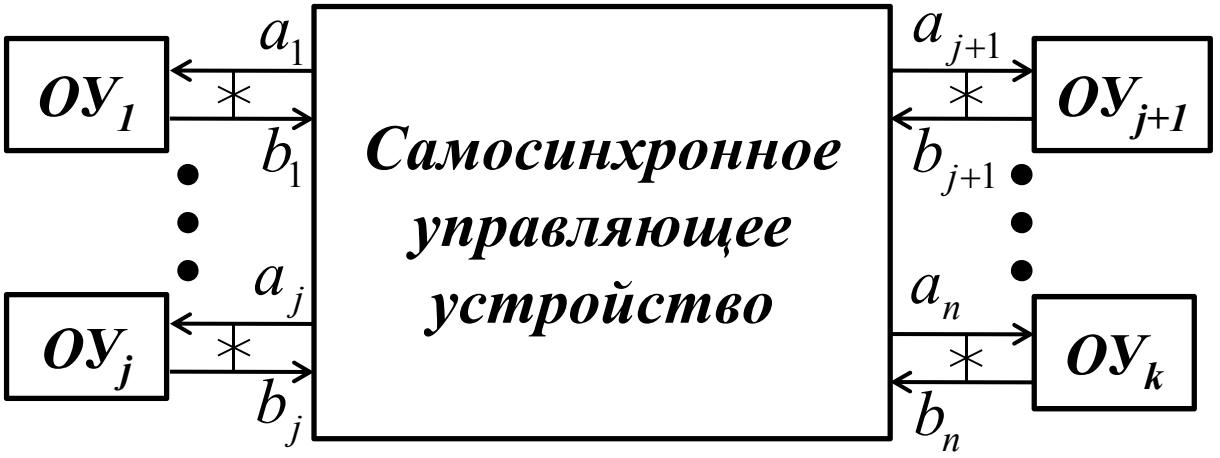
Таким образом, конечный автомат является ***самосинхронным*** (правильнее ***самосинхронизирующимся***), если он принадлежит классу индицируемых автоматов, поведение которых инвариантно к величинам задержек элементов.

Заметим, что самосинхронные автоматы могут работать с синхронной внешней средой. При этом на вход a подаются синхросигналы, а выходной сигнал b служит для контроля правильности функционирования.

Логическое Проектирование Асинхронных Схем

Лекция 3

5. Моделирование управления



Спецификация асинхронных процессов с помощью динамических моделей (языков) описания их функционирования является основой для создания средств анализа и синтеза самосинхронных схем. С другой стороны, работа динамических моделей может быть, в свою очередь, промоделирована с помощью схем. Отсюда возникла идея непосредственного перехода от описаний к схемам (***прямая трансляция описаний в самосинхронные схемы***).

Этот подход очень эффективен при ограниченных сроках проектирования самосинхронных схем управления. Он представляет собой простейший метод проектирования, но может привести к большому перерасходу оборудования.

5. Моделирование управления

5.1. Моделирование сетей Петри

Будем рассматривать только устойчивые и безопасные сети Петри. Такое ограничение приводит к получению полумодулярных моделирующих схем.

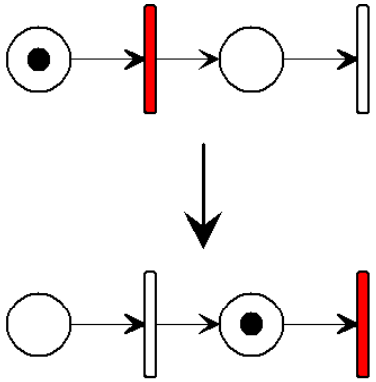
Основные идеи, лежащие в основе прямой трансляции таких сетей Петри:

1. Поведение асинхронных систем определяется хендшейк взаимодействием между блоками системы и/или между ее блоками и схемой управления.
2. В терминах управления моделью блока системы является инкорпорированная задержка между сигналами «запрос» и «ответ».
3. На уровне спецификации поведения системы сетью Петри задержки отсутствуют и поэтому работа блока может быть представлена только одним из хендшейк сигналов, например сигналом «запрос».
4. Частичный порядок событий, определяемый сетью Петри, сохраняется в ее моделирующей схеме.
5. Схема, моделирующая поведение сети Петри, одновременно является и схемой управления.

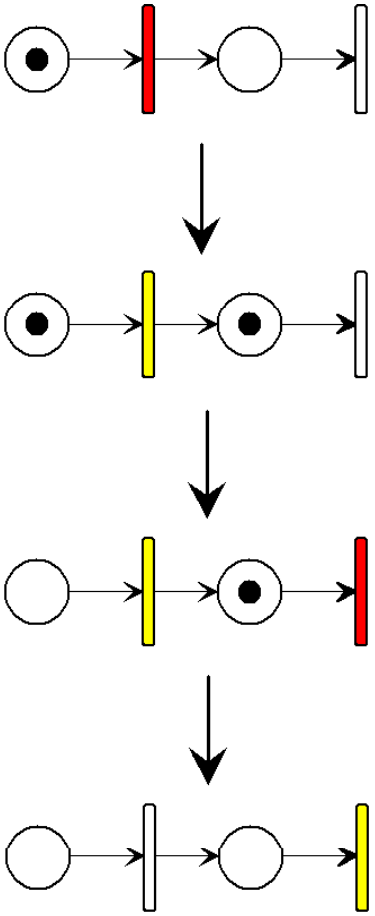
5. Моделирование управления

Моделирование неделимой операции продвижения маркера

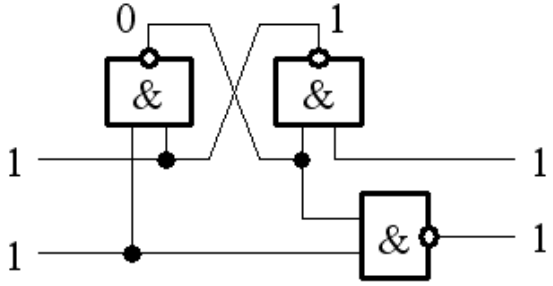
а)



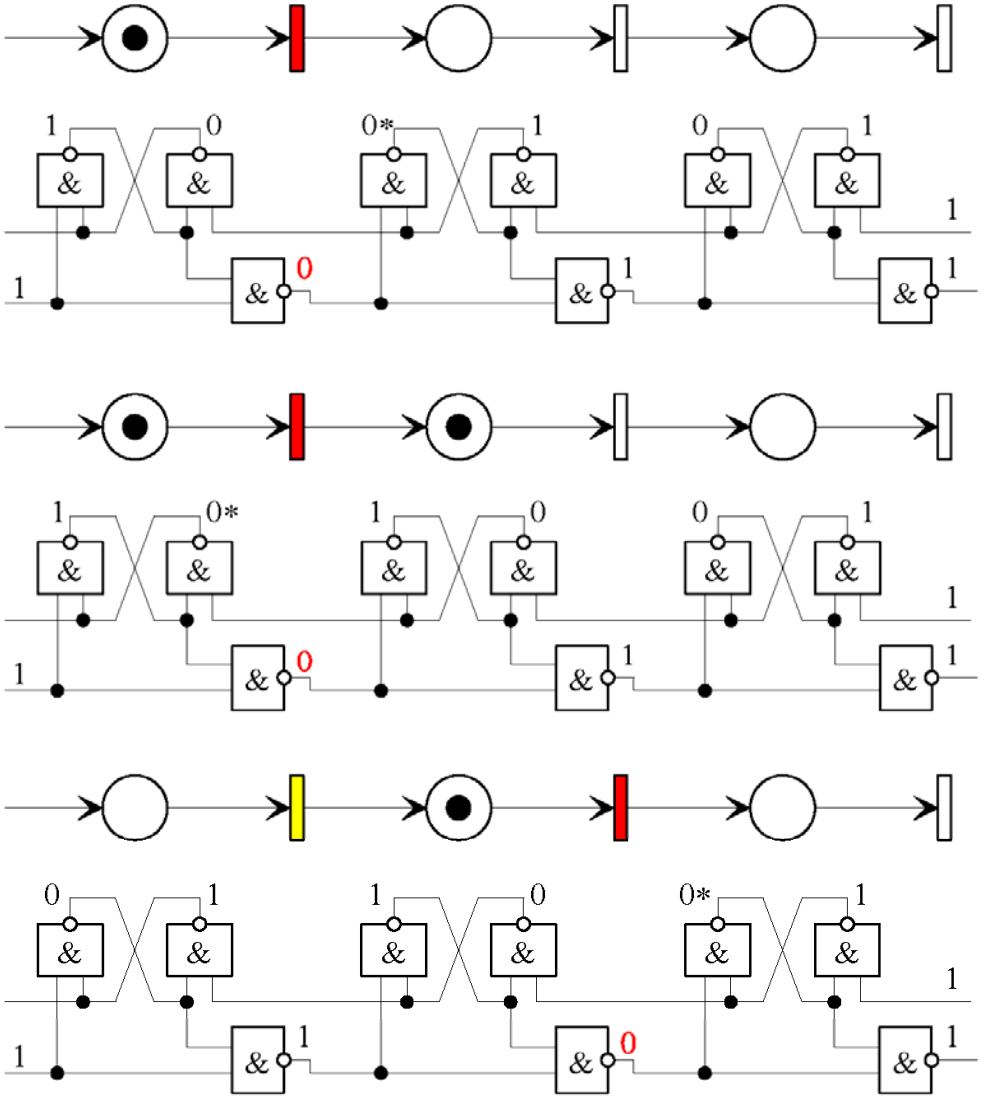
б)



в) *Элемент Давида*



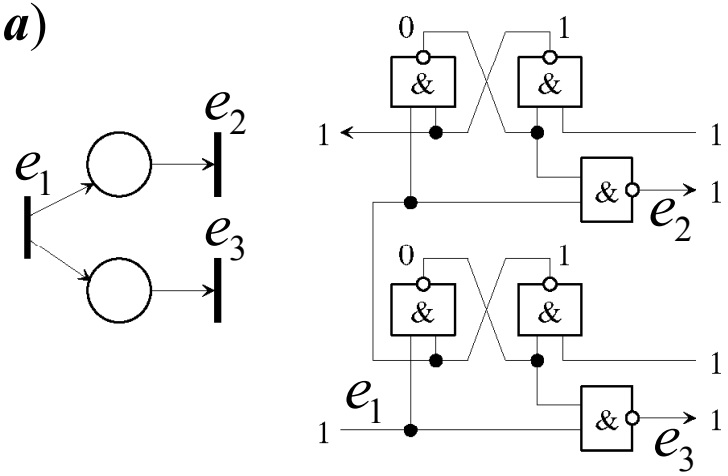
5. Моделирование управления



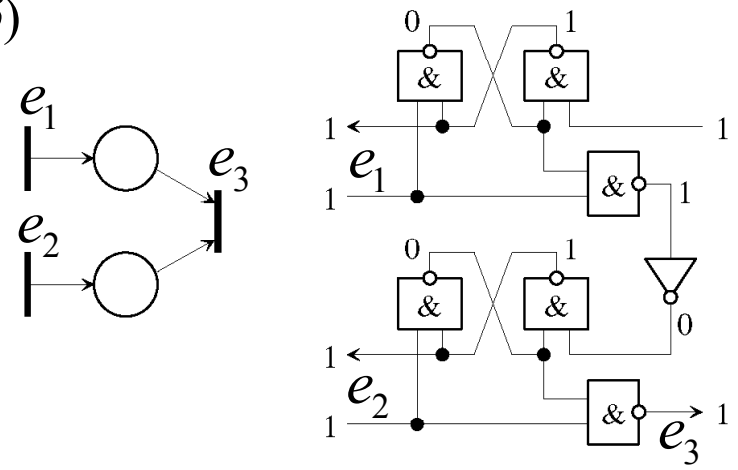
5. Моделирование управления

Разветвитель (а) и синхронизатор (б) (последовательное упорядочение)

а)

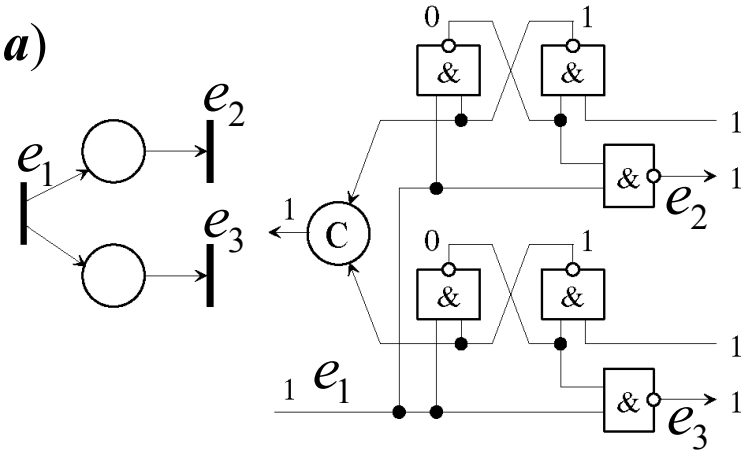


б)

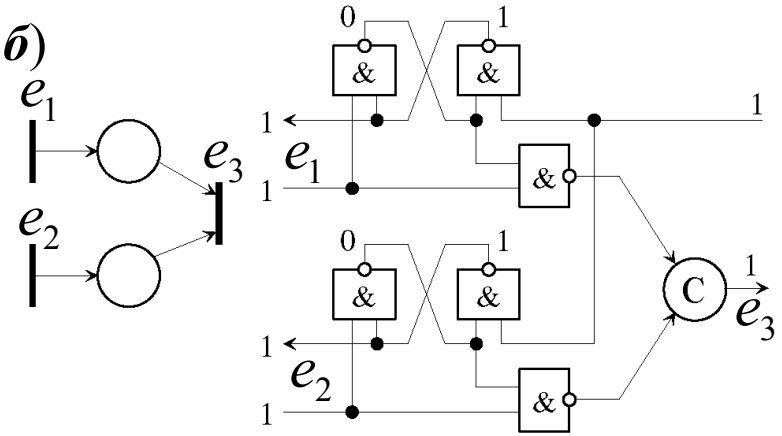


Разветвитель (а) и синхронизатор (б) (параллельная работа)

а)

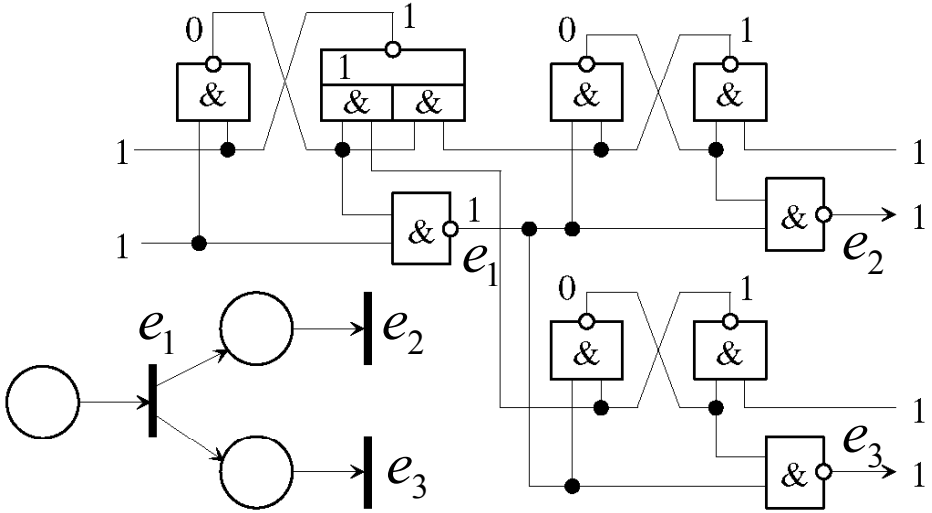


б)

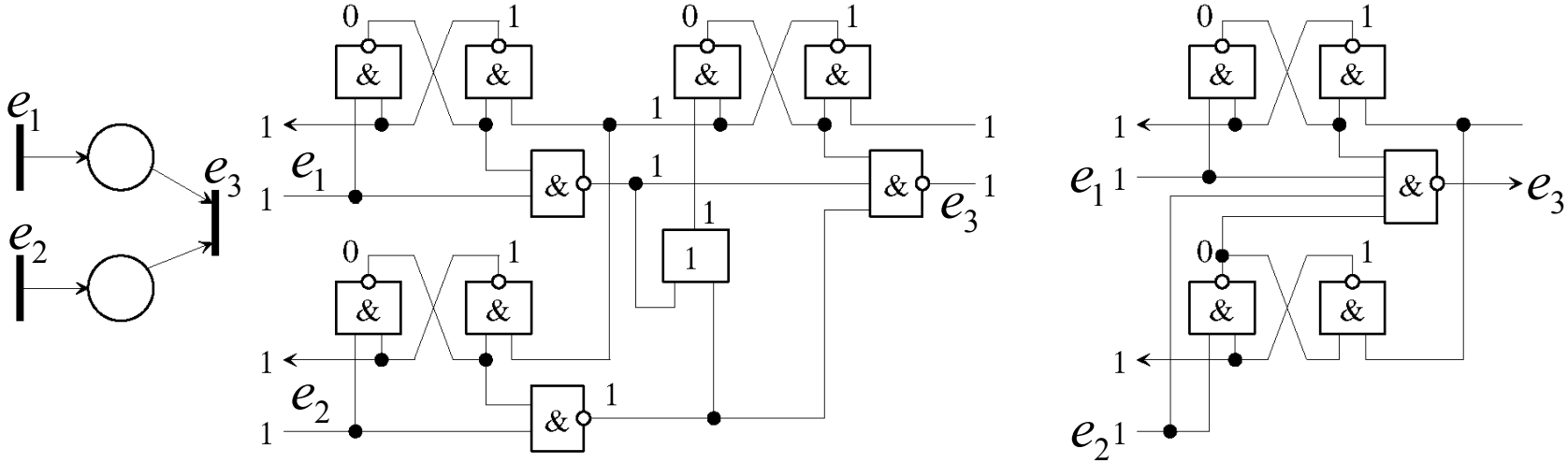


5. Моделирование управления

Другая реализация разветвителя (параллельная работа)

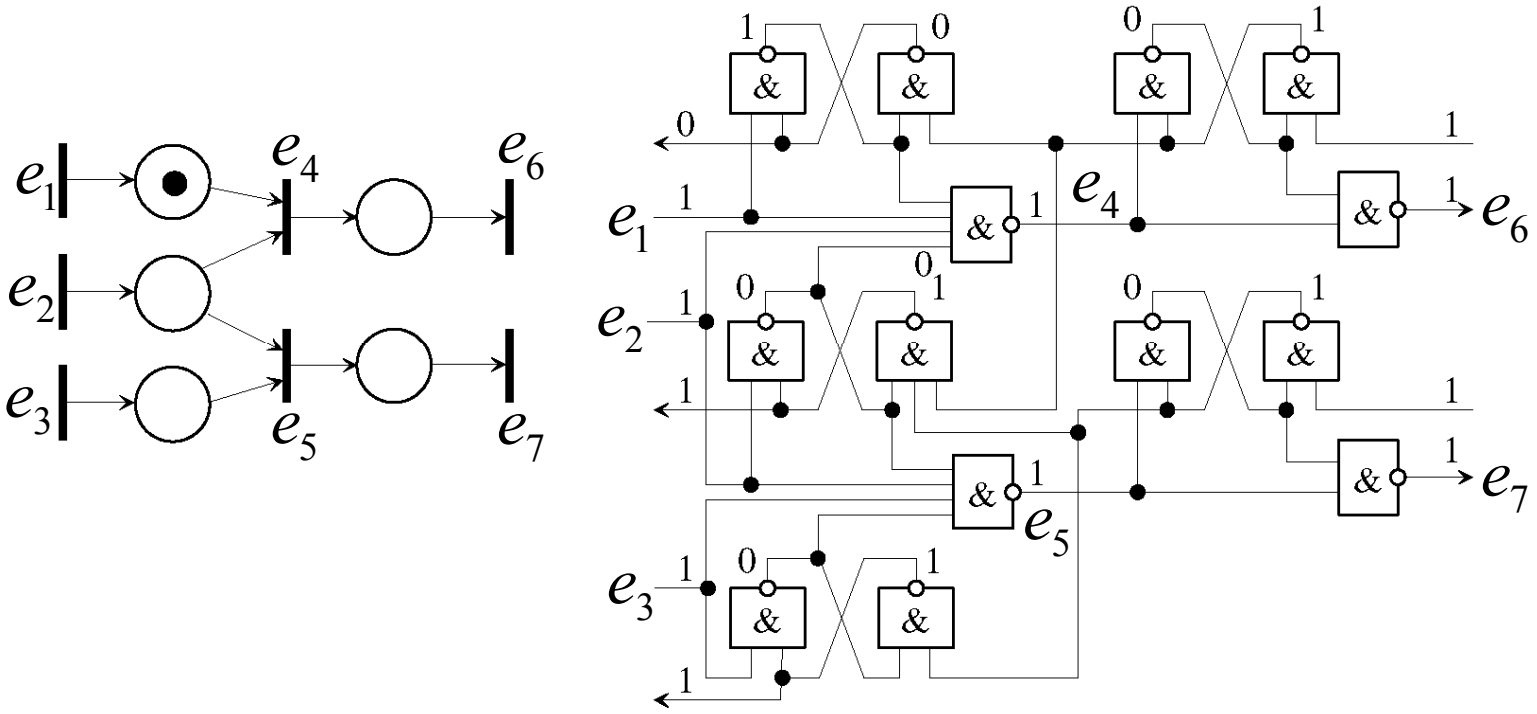


Другие реализации синхронизатора (параллельная работа)



5. Моделирование управления

***Альтернативный разветвитель* (*условный переход*)**



5. Моделирование управления

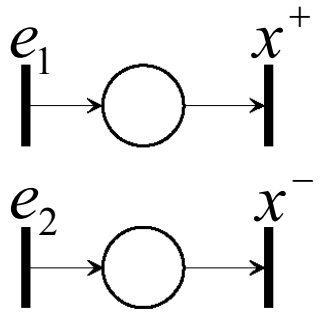
Интерпретация событий в сетях Петри:

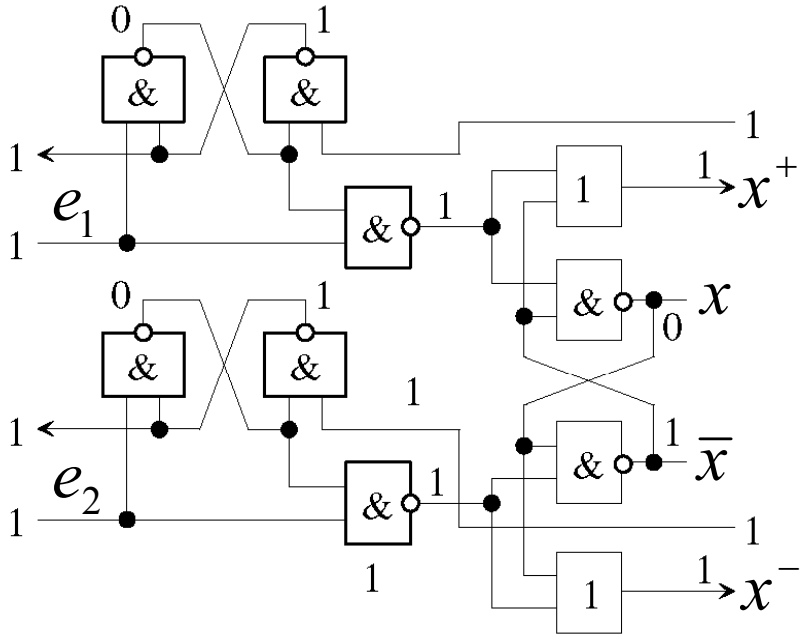
$1 \to 0 \to 1$ $(0 \to 1 \to 0)$ (событие e)

$0 \to 1$ (событие $x+$)

$1 \to 0$ (событие $x-$)

Моделирование событий $x+$, $x-$***:***

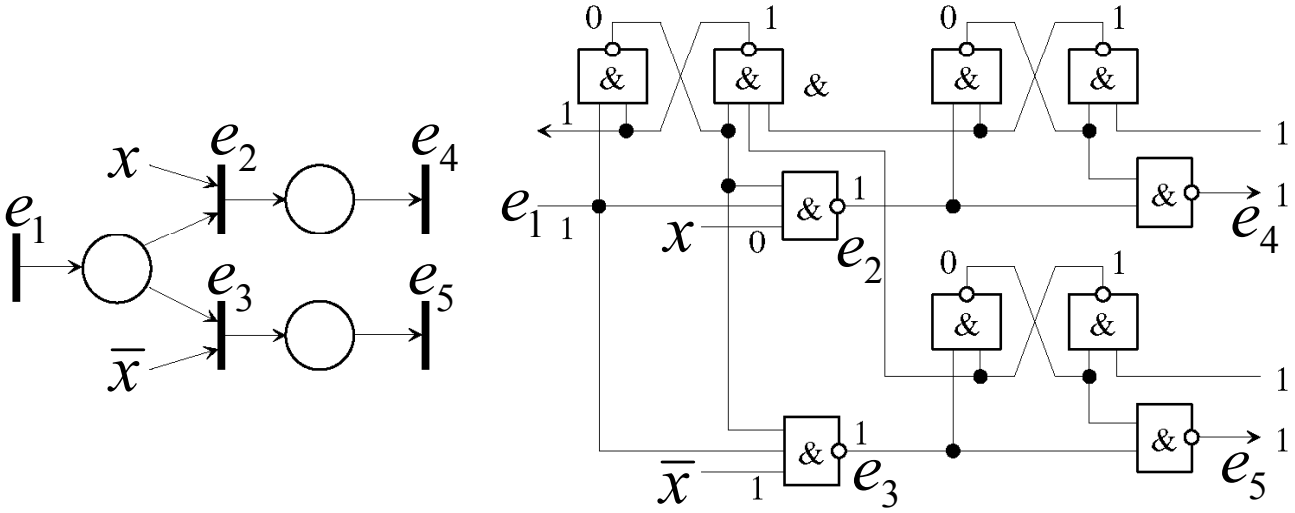




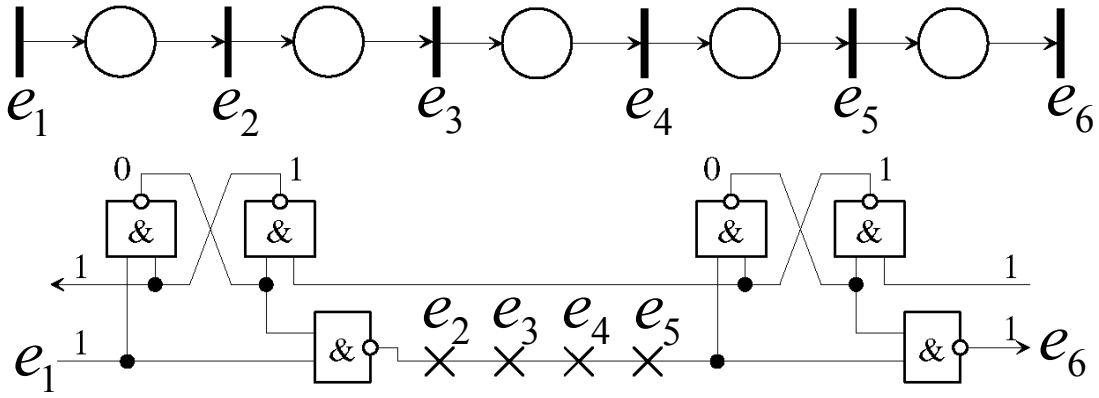
События e_1 и e_2 должны быть ортогональны

5. Моделирование управления

Условный переход, управляемый переменной

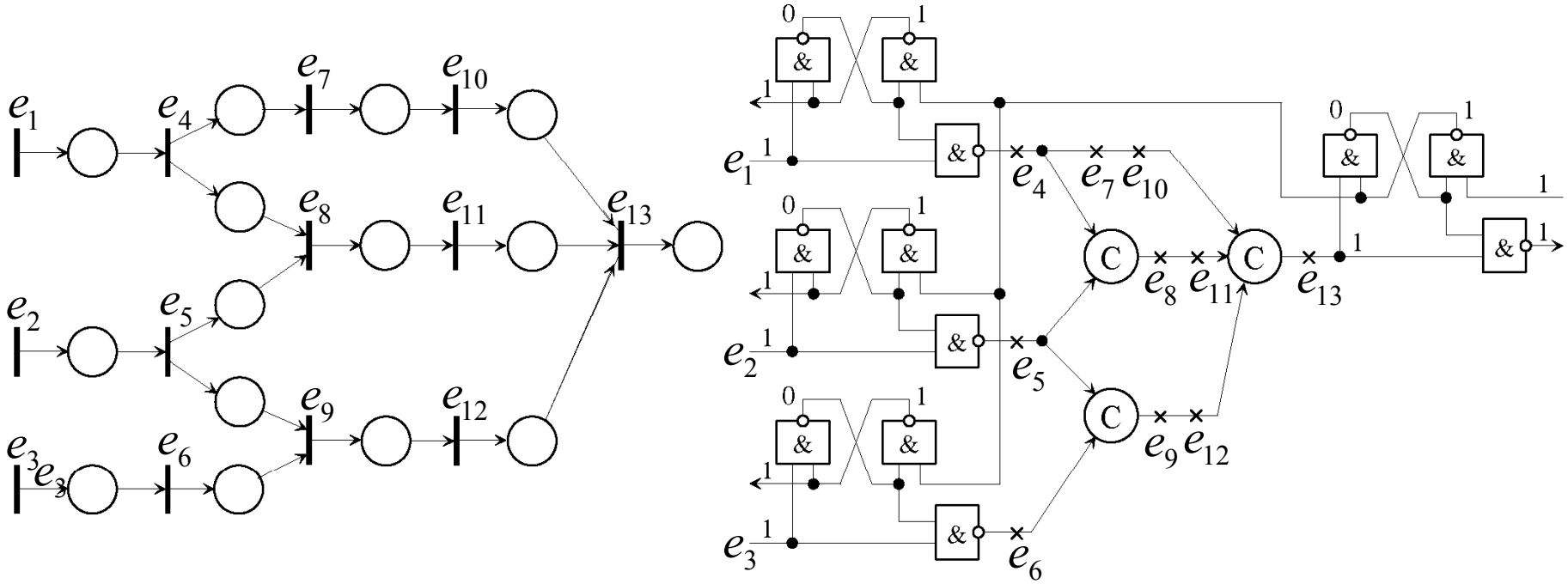


Упрощенная реализация линейного фрагмента



5. Моделирование управления

Пример упрощенной реализации фрагмента сети Петри



5. Моделирование управления

Ограничения, накладываемые этим подходом на вид сетей Петри:

1. Сети Петри должны быть устойчивыми и безопасными.
2. Как правило, условные вершины не должны иметь более одной входной дуги; устойчивые и безопасные сети Петри с таким ограничением называются ***автоматными.*** Допускаются условные вершины с несколькими входными дугами, если входной маркер может поступить только по одной из них (альтернативное ИЛИ).
3. Моделируемая сеть Петри не должна содержать циклов длины меньше трех (в цикле должно быть, как минимум, три условия и три события), так как если цепочку из двух элементов Давида замкнуть на себя, то она не будет работать из-за дедлока (встает в устойчивое состояние).

Последнее ограничение всегда можно обойти, «вставляя» в сеть Петри фиктивные условия и события, не нарушающие порядка срабатываний других событий.

5. Моделирование управления

5.2. Моделирование параллельных асинхронных блок-схем

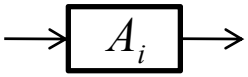
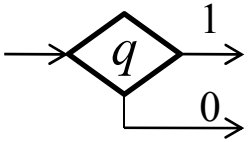
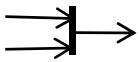
Параллельные асинхронные блок-схемы (ПАБС) были рассмотрены в лекции 1 (слайд 30). Для построения схем, моделирующих поведение ПАБС, необходима дополнительная детализация, связанная с тем, что кроме организации взаимодействия управляющих асинхронными процессами сигналов нужна информация о семантике самих процессов.

Например, может потребоваться многократное использование одного и того же операционного блока в различных частях ПАБС. Для повторного запуска блока необходимо дождаться окончания его работы и использования полученных в нем данных другими блоками.

Во-вторых, следует учитывать двухфазную дисциплину работы управляемых блоков и порядок следования фаз в различных блоках. В зависимости от порядка смены фаз блоков могут быть получены различные реализации схемы управления с различными характеристиками.

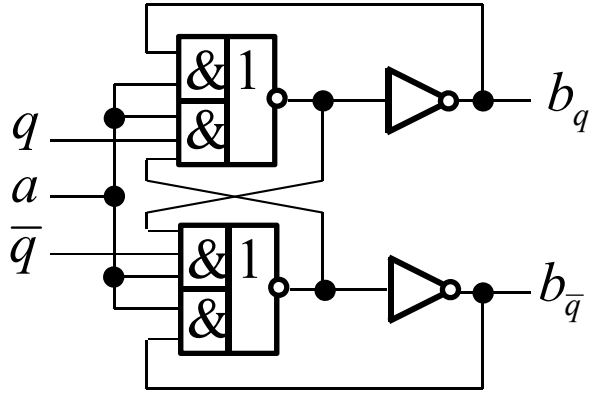
5. Моделирование управления

5.2.1. Реализация управления по стандартным фрагментам

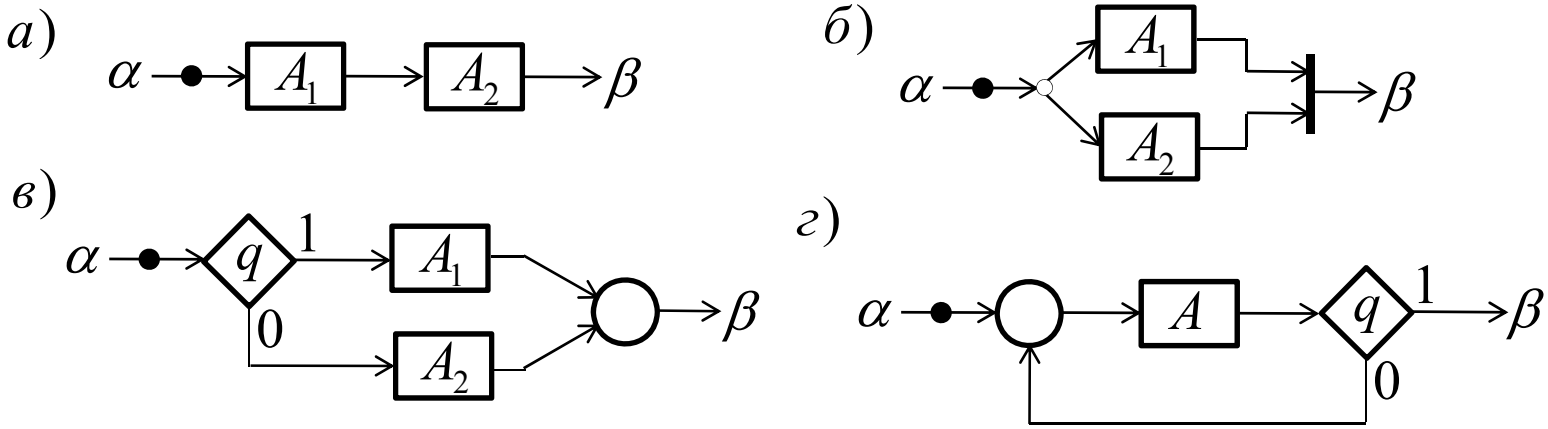
№	Тип вершины ПАБС	Наименование	Реализация
		Дуга	Провод
		Маркированная дуга	Инвертор
1		Оператор	Элемент задержки
2		Условный переход	Переключатель (схема на слайде 15)
3		Альтернативная сборка	Элемент ИЛИ
4		Бифуркатор	Разветвление провода
5		Синхронизатор	*C*-элемент

5. Моделирование управления

Схема переключателя (условного перехода)



Элементарные блок-схемы



5. Моделирование управления

Для ПАБС определим ***операцию композиции***: ПАБС *B* будем называть композицией ПАБС *C* и *D*, если *B* может быть получена из *C* заменой одной из ее вершин типа «оператор» на ПАБС *D*. К полученным таким образом ПАБС снова можно применять операцию композиции и т.д.

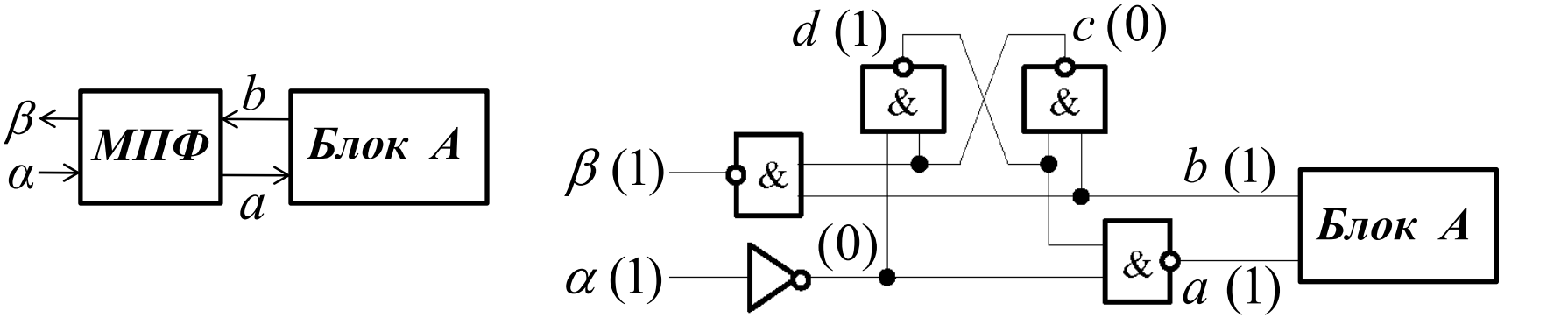
Назовем ПАБС, полученные из элементарных рекурсивным применением операции композиции, ***правильными***. Множество всех правильных ПАБС замкнуто относительно операции композиции. Заметим, что неправильные ПАБС всегда можно свести к алгоритмически эквивалентным правильным.

Такой подход к моделированию ПАБС, к сожалению, не работает даже для правильных ПАБС, поскольку предполагает следующую дисциплину работы блоков: сначала все операционные блоки системы выполняют рабочую фазу в соответствии с порядком, определяемым ПАБС, а затем следует фаза гашения блоков в том же порядке. Но в таком случае не работают циклы.

Вторая проблема заключается в том, что операторы в ПАБС могут повторяться. Как запускать один и тот же оператор из разных мест? Следовательно, нужно менять дисциплину смены фаз и вводить дополнительные модули.

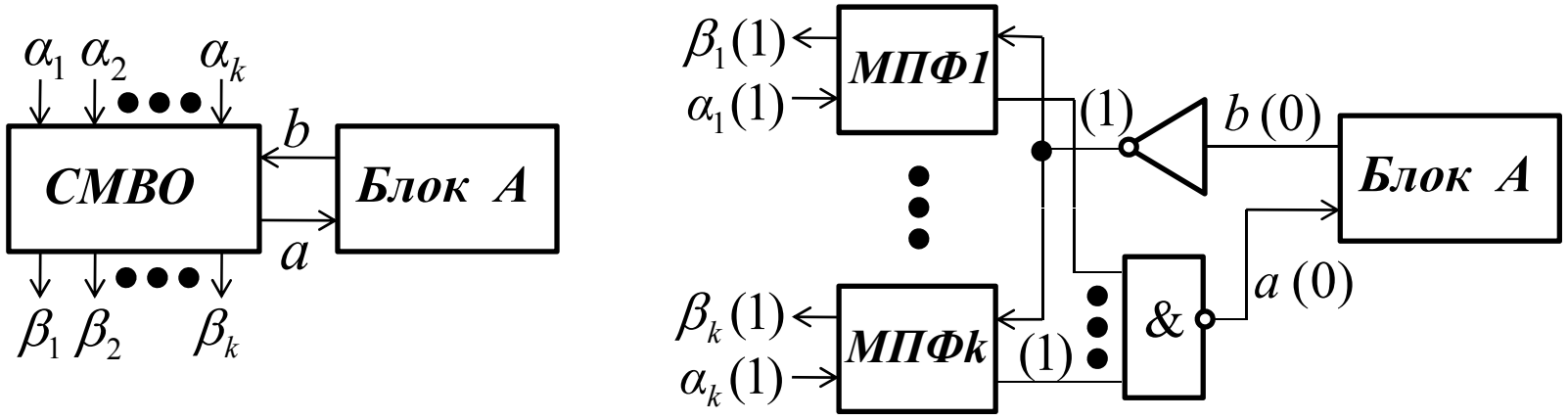
5. Моделирование управления

Модуль для преобразования фазовой дисциплины



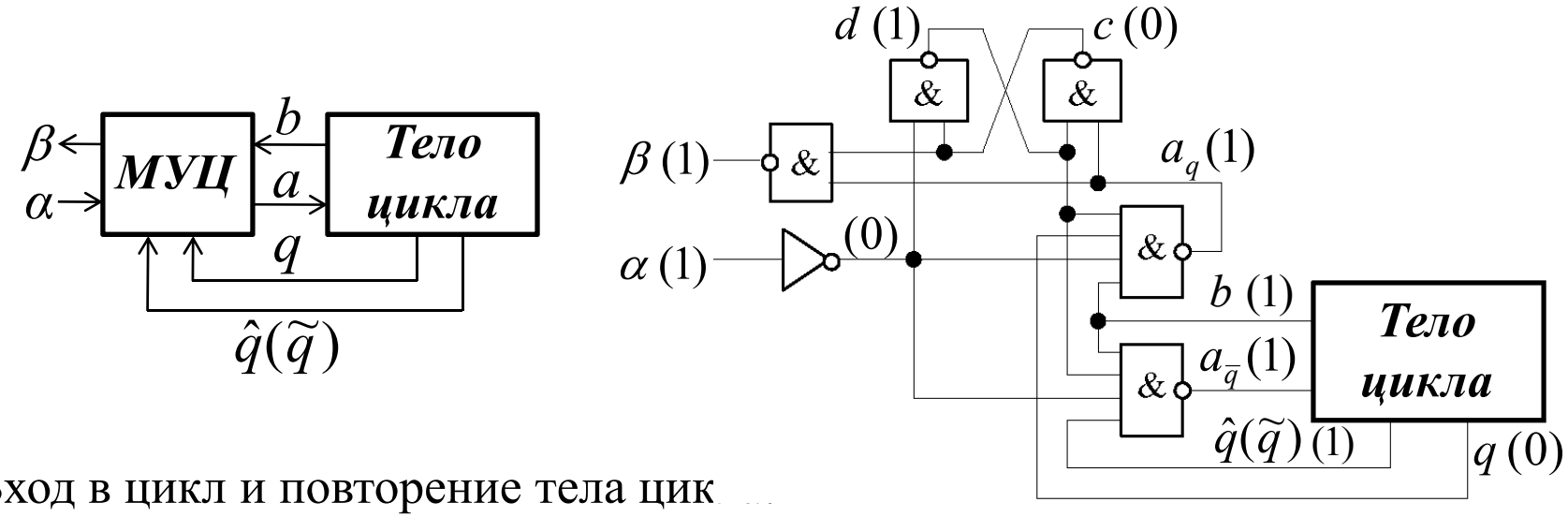
$$-\alpha \rightarrow -a \rightarrow -b \rightarrow +c \rightarrow -d \rightarrow +a \rightarrow +b \rightarrow -\beta \rightarrow +\alpha \rightarrow +d \rightarrow -c \rightarrow +\beta \rightarrow$$

Схема многократного вхождения оператора



5. Моделирование управления

Модуль управления циклом



Вход в цикл и повторение тела цик ...

$$-\alpha \to -a_{\bar{q}} \to -b \to +a_{\bar{q}} \to +b \to \ldots \to -a_{\bar{q}} \to -b \to +a_{\bar{q}} \to +b,$$

выход из цикла: $+b \to -a_q \to +c \to -d \to +a_q \to -\beta,$

фаза гашения: $+\alpha \to +d \to -c \to +\beta.$

Значения сигналов $q,\ \hat{q}(\tilde{q})$ должны изменяться в фазе гашения (когда $b = 0$).

На основе *МУЦ* можно построить схему ***многократного использования цикла*** точно так же, как была построена схема многократного вхождения оператора.

5. Моделирование управления

Рассмотренные модули и схемы достаточны для построения моделирующих схем для класса правильных ПАБС. Однако нужно быть готовым к трудностям, связанным с организацией двухфазной работы самосинхронных блоков, представляющих операторы ПАБС.

Универсальность подхода дает основания предполагать , что получающиеся моделирующие схемы обладают большой избыточностью. С этой точки зрения более эффективным является подход, использующий спусковые функции.

5.2.2. Реализация по спусковым функциям

Спусковой функцией называется предикат, зависящий от состояния процесса и разрешающий активизацию оператора.

ПАБС размечают переменными, представляющими управляющие сигналы. На этих переменных строят спусковые функции для операторов. Составление этих функций содержит неформальные моменты, поэтому экономичность решений зависит от искусства разработчика.

5. Моделирование управления

ПАБС определяют лишь условия инициации операционных блоков. Для учета двухфазной дисциплины работы блоков спусковые функции должны быть дополнены условиями гашения блоков, которые могут быть составлены достаточно произвольно. Далее спусковые функции реализуются методами построения тестеров и индикаторов.

5.3. Функциональная полнота и синтез полумодулярных схем

Существующие подходы к проектированию асинхронных управляющих схем, определяют следующую их классификацию:

- ***Схемы с ограниченными задержками*** **(*Bounded delays*):** разумное ограничение задержек элементов и проводов.

- ***Схемы, не зависящие от скорости*** **(*Speed-independent*):** задержки элементов произвольные, но конечные, разброс задержек в проводах после разветвления не превышает минимальную задержку элемента.

5. Моделирование управления

- ***Схемы, нечувствительные к задержкам*** (***Delay-insensitive***)**:** произвольные, но конечные задержки элементов и проводов. Такие схемы требуют специальных элементов, последовательный запуск параллельных процессов, не допускают разветвлений проводов и дают бешенную избыточность. Практического значения не имеют.

- ***Схемы, квази нечувствительные к задержкам*** (***Quasi-delay insensitive***)**:** схемы, не зависящие от задержек элементов и задержек выделенных (длинных) проводов. Могут быть спроектированы в рамках схем, не зависящих от скорости.

Для нас наибольший интерес будут представлять ***схемы, не зависящие от скорости***. Такие схемы, в свою очередь имеют следующую классификацию:

- ***Полумодулярные схемы*****:** реализуют полумодулярные диаграммы Маллера.
- ***Дистрибутивные схемы*****:** реализуют дистрибутивные диаграммы.
- ***Последовательные схемы*****:** реализуют последовательные диаграммы.

5. Моделирование управления

5.3.1. Постановка задачи

Пусть по диаграмме Маллера построена схема Маллера, заданная системой функций

$$z_i = f_i(z_1,...,z_i,...,z_n), \quad i = 1,2,...,n, \tag{5.1}$$

где f_i – собственная функция *i*-го элемента схемы. Если по этой схеме построить диаграмму на кумулятивных состояниях, которая будет *полумодулярной* , то такую схему будем тоже называть *полумодулярной*.

Если функции z_i настолько сложны, что не могут быть реализованы одним элементом базиса, то необходимо их декомпозировать на подфункции, являющиеся собственными функциями элементов базиса $L = \{\varphi_j\}, \ = 1,2,...,\mathrm{m}$. Однако при этом схема, как правило, перестает быть полумодулярной, поскольку она представляет уже другую схему Маллера.

Если удается найти такую реализацию функций z_i из элементов базиса, что вновь полученная схема является полумодулярной, то будем называть ее ***корректной***.

5. Моделирование управления

Систему функций $L = \{\varphi_j\},\ = 1,2,...,m,$ будем называть ***функционально полной в классе полумодулярных*** (***дистрибутивных*** или ***последовательных***) схем, если для любой схемы из этого класса можно построить ее корректную реализацию на элементах с собственными функциями их$L = \{\varphi_j\}$.

5.3.2. Некоторые свойства полумодулярных схем

Определение 5.1. Схема является ***полумодулярной***, если в ней каждый возбужденный элемент переходит в устойчивое состояние только путем изменения значения своего выхода.

Заметим, что полумодулярная схема является схемой, не зависящей от скорости (задержек элементов). Обратное утверждение не верно. Поэтому следует научиться строить полумодулярные схемы.

Теорема 5.1. *Собственная функция любого элемента z_i полумодулярной схемы либо изотонна относительно переменной z_i, либо от нее не зависит.*

5. Моделирование управления

Доказательство. Разложение Шеннона дает $z_i = z_i\varphi_i(z_i = 1) \vee \bar{z}_i\varphi_i(z_i = 0)$.
Если $\varphi_i(z_i = 1) = 0,\ \varphi_i(z_i = 0) = 0,$ то $z_i = 0$ и не зависит от себя.
Если $\varphi_i(z_i = 1) = 1,\ \varphi_i(z_i = 0) = 1,$ то $z_i = 1$ и от себя не зависит.
Если $\varphi_i(z_i = 1) = 1,\ \varphi_i(z_i = 0) = 0,$ то $z_i = z_i$, т.е. изотонна относительно z_i.
Если $\varphi_i(z_i = 1) = 0,\ \varphi_i(z_i = 0) = 1,$ то $z_i = \bar{z}_i$, т.е. z_i не имеет невозбужденного состояния, что противоречит свойству полумодулярности.

Следствие 5.1. *Собственная функция любого элемента полумодулярной схемы представима в виде* $z_i = z_i\varphi_i(z_i = 1) \vee \varphi_i(z_i = 0) = z_i\overline{R}_i \vee S_i,\ R_iS_i = 0.$
Легко видеть, что это уравнение *RS*-триггера.

5.3.3. Совершенная реализация полумодулярных схем

Предполагается, что для построения схем используются элементы И-ИЛИ-НЕ, собственные функции которых антитонны по входным переменным. Для реализации изотонных функций необходимо иметь, как минимум, два элемента с выходами $z_i, \hat{z}_i$ (или $z_i, \widetilde{z}_i$), т.е. реализации должны быть двухканальными.

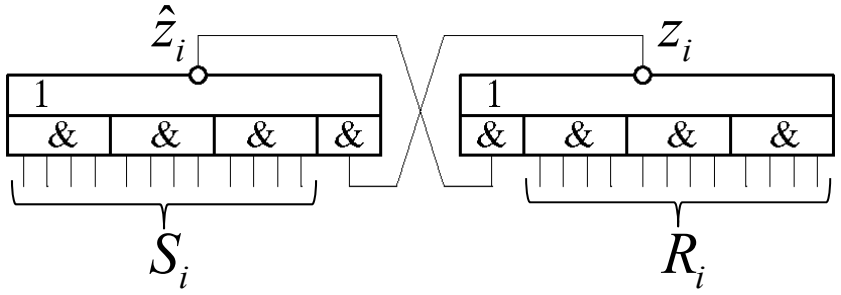
5. Моделирование управления

В полумодулярной схеме Маллера собственная функция любого элемента $z_i = z\overline{R}_i \vee S_i,\ R_iS_i = 0$, может быть представлена в виде

$$z_i = z_i\overline{R}_i \vee \overline{R}_iS_i \vee R_iS_i = \overline{R}_i(z_i \vee S_i) = \overline{R_i \vee \overline{z_i \vee S_i}},$$

где функции R_i и S_i не зависят от z_i. Очевидно, что эта функция может быть реализована с помощью *RS*-триггера:

$$z_i = \overline{R_i \vee \hat{z}_i},\quad \hat{z}_i = \overline{z_i \vee S_i},\ R_iS_i = 0.$$



Таким образом, существует возможность корректной реализации собственных функций элементов $z_i,\ \hat{z}_i$. Заметим, что триггер изменяет свое состояние через транзитное состояние 00, которое не может вызвать возбуждения других элементов схемы. Для того, чтобы эта возможность стала реальностью, сформулируем требования к реализации функций R_i и S_i.

5. Моделирование управления

Пусть полумодулярная схема Маллера находится в некотором состоянии, в котором часть переменных устойчивы, а другая часть, включающая переменную z_i, состоит из возбужденных переменных. В полумодулярной схеме возбуждение переменной z_i не должно сниматься при произвольном порядке переключений других возбужденных переменных до тех пор, пока эта переменная не переключится.

Это возможно в двух случаях:
1) либо функция переменной z_i зависит только от устойчивых переменных,
2) либо она зависит от возбужденных переменных и ее минимальная ДНФ изменяет свое значение при переключении возбужденных переменных, а сокращенная ДНФ сохраняет свое значение (за счет избыточных термов).

Если функция элемента z_i зависит только от устойчивых переменных для всех возможных состояний схемы Маллера, то функции R_i и S_i могут быть реализованы по минимальным ДНФ формам, в противном случае, их следует представлять в сокращенных ДНФ формах.Такую реализацию полумодулярных схем Маллера будем называть ***совершенной***.

5. Моделирование управления

Пример 5.1. Пусть схема из 4 элементов задана следующей диаграммой:

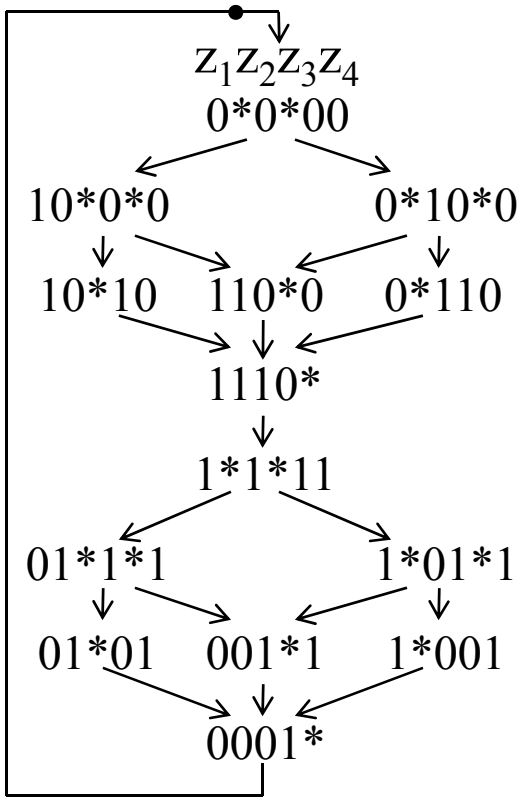


Таблица истинности:

$z_1z_2z_3z_4$	$z_1z_2z_3z_4$
0*0*0 0	1 1 0 0
1 0*0*0	1 1 1 0
0*1 0*0	1 1 1 0
1 1 0*0	1 1 1 0
1 0* 1 0	1 1 1 0
0*1 1 0	1 1 1 0
1 1 1 0*	1 1 1 1
0 0 0 1*	0 0 0 0
1*0 0 1	1 1 0 1
0 1* 0 1	0 0 0 1
0 0 1*1	0 0 0 1
1*0 1*1	0 0 0 1
0 1*1*1	0 0 0 1
1*1*1 1	0 0 1 1

Схема Маллера:

$$z_1 = \bar{z}_4,\ z_2 = \bar{z}_4,$$

$$z_3 = z_1z_2 \vee z_1\bar{z}_4 \vee z_2\bar{z}_4 \vee z_3\bar{z}_4,$$

$$z_4 = z_1z_2z_3 \vee z_4(z_1 \vee z_2 \vee z_3).$$

Представим z_1 и z_2 как:

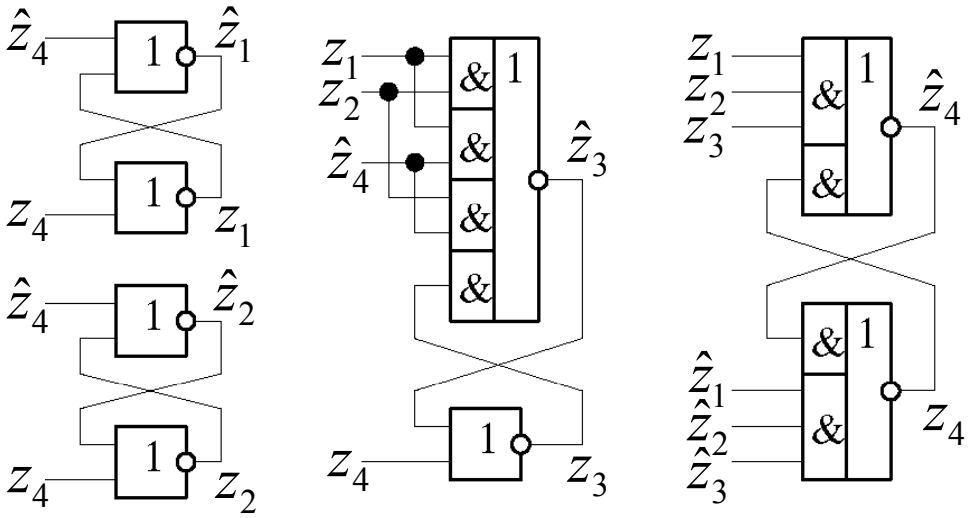
$$z_1 = \bar{z}_4 \vee z_1\bar{z}_4,\ z_2 = \bar{z}_4 \vee z_2\bar{z}_4.$$

Тогда:

$$S_1 = \hat{z}_4,\ R_1 = z_4,$$
$$S_2 = \hat{z}_4,\ R_2 = z_4,$$
$$S_3 = z_1z_2 \vee z_1\hat{z}_4 \vee z_2\hat{z}_4,\ R_3 = z_4,$$
$$S_4 = z_1z_2z_3,\ R_4 = \hat{z}_1\hat{z}_2\hat{z}_3.$$

5. Моделирование управления

Реализация схемы Маллера:



5.3.4. Простые схемы

В процессе циклической работы схемы каждая переменная может переключиться несколько раз прежде, чем схема вернется в исходное состояние.

Построим для схемы диаграмму переходов и для каждой возбужденной переменной припишем порядковый номер ее возбуждения.

5. Моделирование управления

Максимальное множество состояний диаграммы переходов, в которых встречается возбужденное значение (0* или 1*) переменной z_i с одинаковым номером ее возбуждения, назовем ***зоной возбуждения*** этой переменной. Обозначим k-ю зону возбуждения переменной z_i как $A(z_i, k)$.

Состояние диаграммы $\alpha \in A(z_i, k)$ назовем ***минимальным состоянием*** k-й зоны возбуждения переменной z_i, если для любого состояния $\beta \in A(z_i, k)$ ($\beta \neq \alpha$) не существует траектории из β в α, целиком лежащей в $A(z_i, k)$.

Будем говорить, что зона возбуждения $A(z_i, k)$ имеет ***нижнюю грань,*** если она содержит только одно минимальное состояние.

Определение 5.2. Схема, полумодулярная относительно начального состояния α, ***дистрибутивна*** относительно состояния α, если на множестве состояний, достижимых из α, все зоны возбуждения всех ее переменных имеют нижнюю грань. В противном случае схема ***недистрибутивна***.

Зоны возбуждения, не имеющие нижней грани, называются ***детонантными***, а число минимальных состояний в детонантной зоне – ***рангом детонантности***.

5. Моделирование управления

Если зона возбуждения $A(z_i, k)$ имеет нижнюю грань, то именно в состоянии нижней грани происходит начальное возбуждение переменной z_i и оно вызывается одним термом T сокращенной ДНФ ее функций возбуждения R_i или S_i .

В процессе работы схемы переменные терма T могут изменять свои значения прежде, чем переключится z_i . Однако в силу полумодулярности схемы возбуждение переменной z_i должно поддерживаться другими термами функции возбуждения. Такое явление будем называть ***перехватом термов***.

Перехват термов существенно затрудняет решение задачи декомпозиции функций возбуждения элементов, так как между термами, участвующими в перехвате, возможны логические состязания.

Определение 5.3. Схему, полумодулярную относительно начального состояния, будем называть ***простой***, если все зоны возбуждения переменных на множестве достижимых состояний свободны от перехватов термов.

В совершенной реализации простой схемы функции возбуждения R_i и S_i могут быть реализованы по минимальным ДНФ.

5. Моделирование управления

Если в зоне возбуждения $A(z_i, k)$ изменяет свое значение хотя бы одна переменная, являющаяся непосредственной причиной возбуждения переменной z_i, то в функциях возбуждения z_i наверняка имеется перехват. Такой перехват, вызванный изменениями входной переменной для зоны возбуждения, назовем ***сложным*** перехватом. Если же в зоне не изменяется ни одной такой входной переменной, то перехват назовем ***простым***.

Теорема 5.2. *По любой полумодулярной схеме можно построить эквивалентную ей полумодулярную простую схему, причем если исходная схема дистрибутивна, то и простая схема будет дистрибутивной.*

Доказательство этой теоремы конструктивно, т.е. дает метод устранения простых и сложных перехватов.

Отметим, что любую недистрибутивную полумодулярную схему можно эквивалентно преобразовать к такому виду, что детонантные переходы будут иметь ранг два с функциями возбуждения элементарного вида $z_l^{\alpha} \vee z_m^{\beta}$.

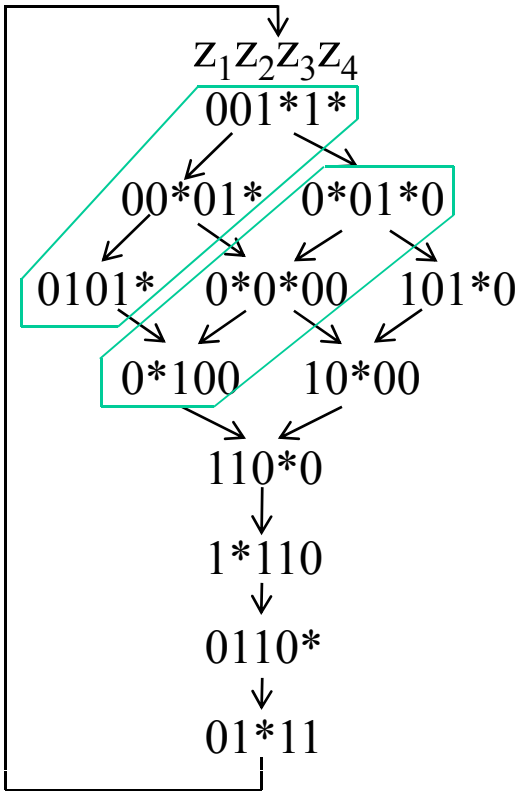
Рассмотрим метод устранения перехватов на примере.

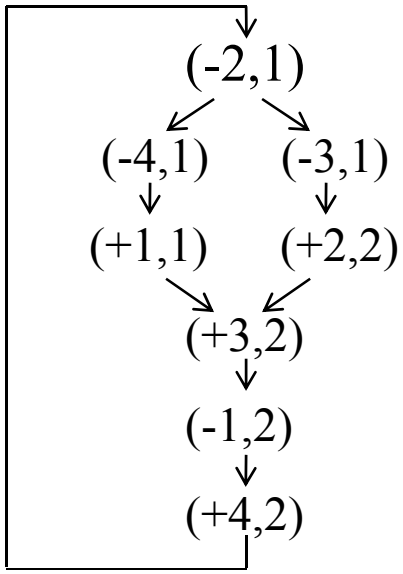
5. Моделирование управления

Пример 5.2. Задана схема системой уравнений:

$$z_1 = (\bar{z}_2 \vee \bar{z}_3)\bar{z}_4,\ z_2 = \bar{z}_3 \vee z_2\bar{z}_4,\ z_3 = (z_1 \vee z_3)z_2,\ z_4 = \bar{z}_1 z_2 z_3.$$

а) ***Диаграмма переходов*** ***Диаграмма изменений***





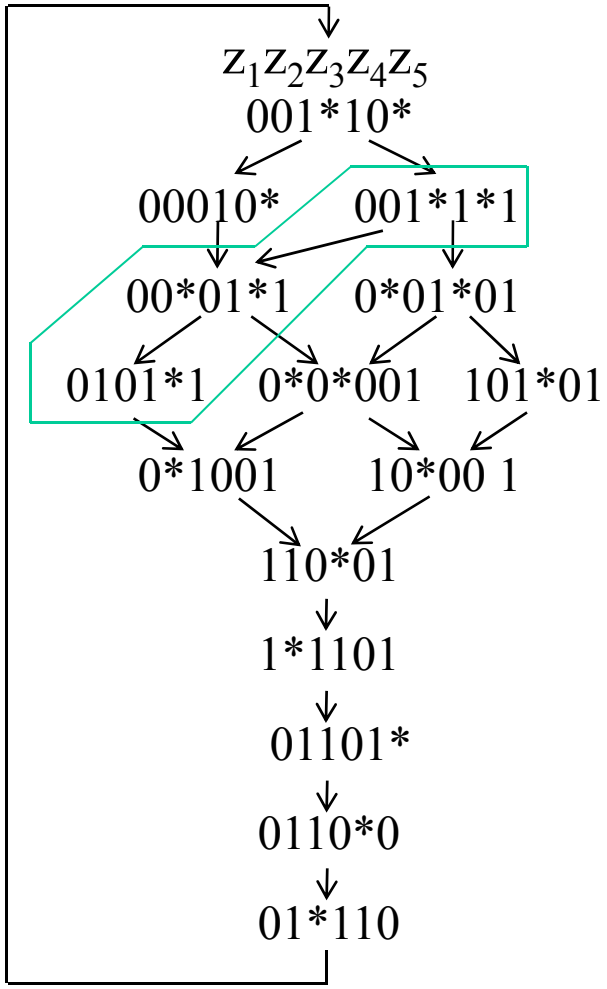
В схеме два перехвата термов. Первый в зоне $A(z_4 = 1^*)$ для входной переменной z_2 является сложным. В этой зоне входные наборы для z_4 изменяются как $101 \to 100 \to 110$.

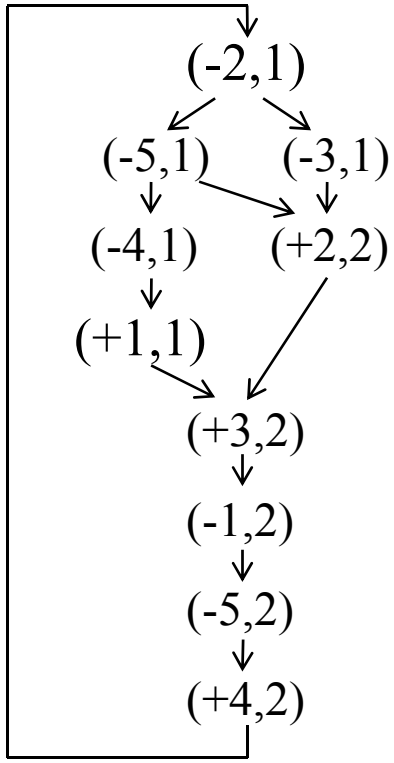
Второй в зоне $A(z_1 = 0^*)$, в которой значение $z_1 = 0^*$ поддерживается за счет перехвата терма $\bar{z}_2\bar{z}_4 = 1$ термом $\bar{z}_3\bar{z}_4 = 1$, является простым.

5. Моделирование управления

Устранение сложного перехвата.

б) Диаграмма переходов Диаграмма изменений





Вводим дополнительную переменную z_5 так, чтобы она не возбуждалась в зоне $A(z_4 = 1^*)$.

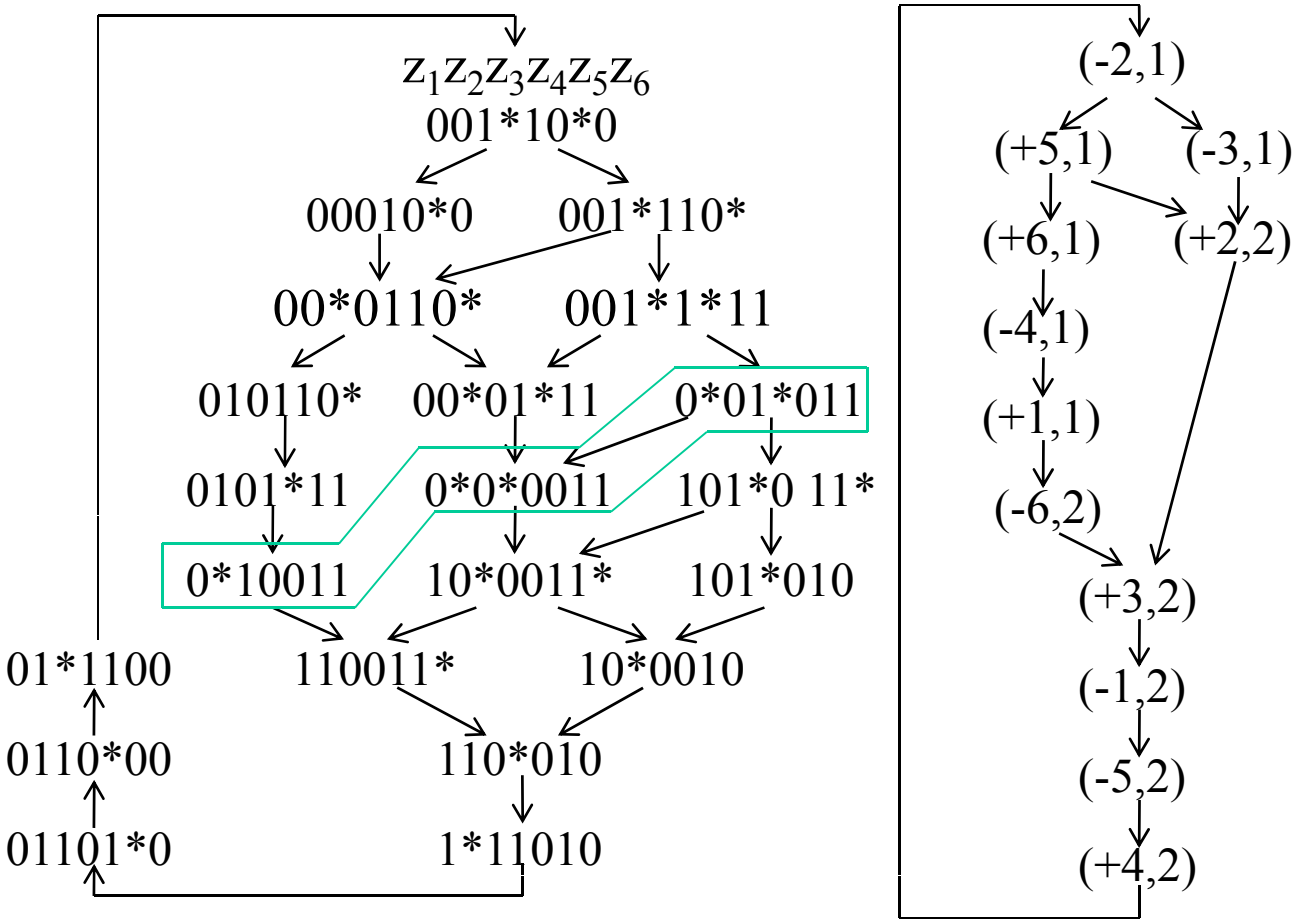
Собственные функции элементов будут иметь вид:

$$z_1 = (\bar{z}_2 \vee \bar{z}_3)\bar{z}_4,$$
$$z_2 = \bar{z}_3 z_5 \vee z_2\bar{z}_4,$$
$$z_3 = (z_1 \vee z_3)z_2,$$
$$z_4 = \bar{z}_5,$$
$$z_5 = z_1 \vee \bar{z}_2 \vee \bar{z}_3.$$

5. Моделирование управления

Устранение простого перехвата.

б) Диаграмма переходов ***Диаграмма изменений***



Введем еще одну дополнительную переменную z_6 так, чтобы она не возбуждалась в зоне $A(z_1 = 0^*)$.

Система уравнений, задающая схему без перехватов, имеет вид:

$$z_1 = (\bar{z}_2 \vee \bar{z}_3 \vee z_6)\bar{z}_4,$$
$$z_2 = \bar{z}_3 z_5 \vee z_2 \bar{z}_4,$$
$$z_3 = (z_1 \bar{z}_6 \vee z_3) z_2,$$
$$z_4 = \bar{z}_5 \bar{z}_6 \vee z_4 \bar{z}_6,$$
$$z_5 = z_1 \vee \bar{z}_2 \vee \bar{z}_3,$$
$$z_6 = z_4 z_5 \vee z_6 \bar{z}_1.$$

Логическое Проектирование Асинхронных Схем

Лекция 4

5. Моделирование управления

5.3.5. Реализация дистрибутивных и последовательных схем

Последовательные схемы относятся к классу дистрибутивных. Действительно, каждая зона возбуждения переменных последовательной схемы состоит из единственного состояния, которое и является нижней гранью.

Теорема 5.3. *Элементы И-НЕ* (*ИЛИ-НЕ*) *с неограниченным числом входов являются функционально полным базисом в классе дистрибутивных схем.*

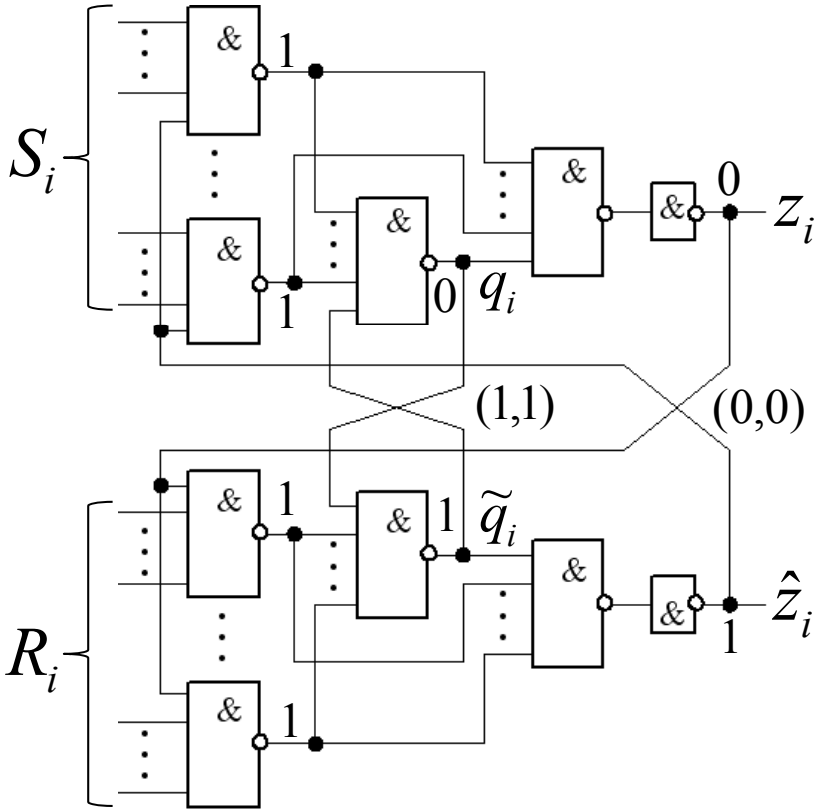
Иными словами любая дистрибутивная схема может быть построена из элементов И-НЕ (двойственная схема на элементах ИЛИ-НЕ). Докажем теорему, предложив метод построения таких схем.

Пусть задана произвольная дистрибутивная диаграмма переходов.

1. Убираем в диаграмме все возможные перехваты термов и переменных.
2. Применим подход совершенной реализации элементов схемы с использованием триггеров на элементах И-НЕ и строим функции возбуждения R_i и S_i так, как это показано на следующей схеме.

5. Моделирование управления

Базовая конструкция для реализации дистрибутивных схем



Элементы И-НЕ первого столбца реализуют термы функций S_i и R_i, в каждой из которых только один терм может принимать значение 1, т.е. выход только одного элемента первого столбца может принимать значение 0. Это обеспечивается отсутствием перехватов и переключением пар входных сигналов $(z_j, \hat{z}_j)$ через состояние (0,0).

Выходные сигналы триггера не могут быть использованы в качестве выходов $(q_i, \tilde{q}_i)$ схемы, так как переключаются через состояние (1,1). Выходные сигналы $(z_i, \hat{z}_i)$ переключаются через состояние (0,0) за счет обратных связей. Эти же связи обеспечивают выполнение условия $R_i S_i = 0$.

5. Моделирование управления

5.5. Синтез полумодулярных схем в ограниченных базисах

Теорема 5.4. *Элемент 2И-НЕ2* (*2ИЛИ-НЕ2*) *являются функционально полным базисом в классе дистрибутивных схем.*

Иными словами, на двухвходовых элементах И-НЕ (ИЛИ-НЕ) с нагрузочной способностью, равной двум, может быть корректно реализована любая дистрибутивная схема.

Доказательство конструктивное, т.е. предлагается метод построения дистрибутивных схем в указанном базисе. Его можно найти в [2]. Поскольку схемы получаются очень сложными и не имеющими практического значения, оно здесь не приводится.

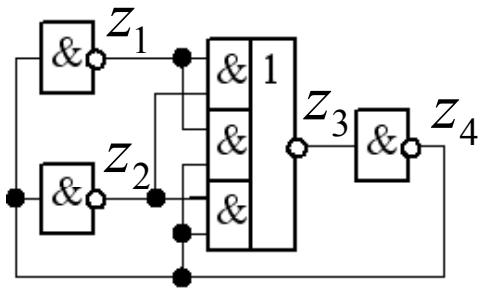
Рассмотрим вопрос о реализации полумодулярных схем в ограниченном базисе. Таким схемы делятся на два класса: дистрибутивные и недистрибутивные. Значит остается невыясненным вопрос о реализации недистрибутивных схем.

5. Моделирование управления

Достаточно исследовать реализации простых полумодулярных схем, ранг детонантности которых не превосходит двух, а функции возбуждения детонантных переходов имеют простейшую форму: $z_r^{\alpha} \vee z_m^{\beta}$.

Начнем с простого примера построения из простых элементов С-элемента с инверторами на входах.

***а*) Исходная схема**



***б*) Некорректная реализация**

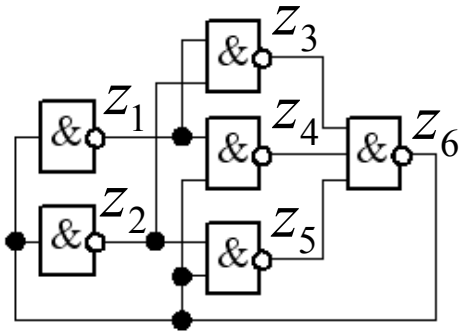
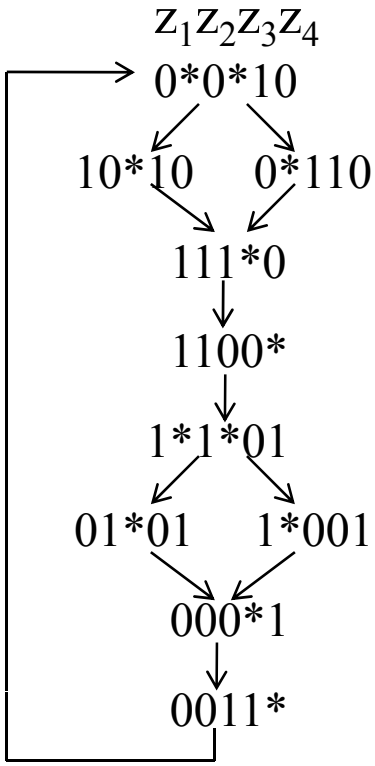


Схема (*а*) описывается дистрибутивной (последовательной) диаграммой переходов и, следовательно, может быть реализована на элементах И-НЕ. Однако в такой реализации все сигналы представлены парофазно. Для автономной схемы это допустимо. Ситуация изменяется если сигналы должны быть однофазными.

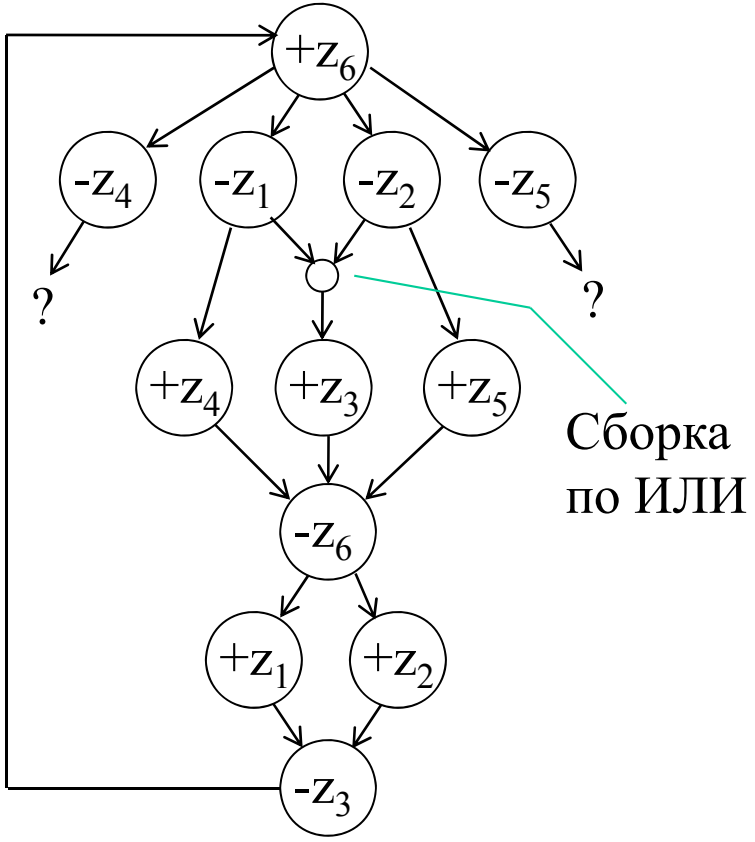
5. Моделирование управления

а) Диаграмма переходов исходной схемы



Эта диаграмма дистрибутивна.

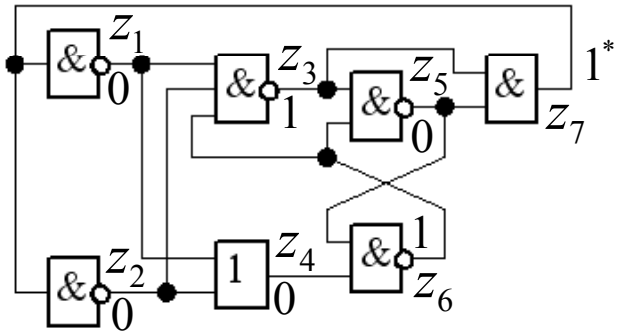
б) Сигнальный граф некорректной реализации



Граф описывает неполумодулярную схему.

5. Моделирование управления

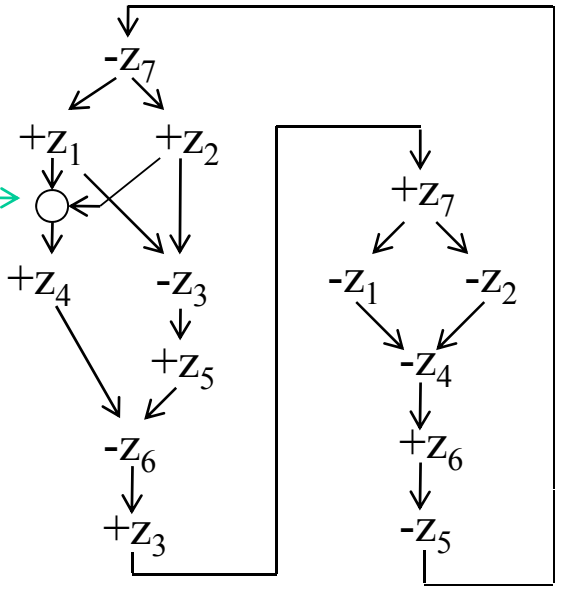
в) Корректная реализация на простых элементах С-элемента с инверторами на входах



Эта схема может быть реализована на элементах 2И-НЕ2 и 2ИЛИ-НЕ2. Для этого трехвходовый элемент реализуется каскадным включением трех элементов, а элементы без инверсии с помощью двух последовательных элементов.

г) Сигнальный граф, иллюстрирующий работу этой схемы

Сборка по ИЛИ →



Типичный пример недистрибутивной схемы. Иногда целесообразно искусственно вводить недистрибутивность с целю увеличения параллельности.

5. Моделирование управления

Можно доказать, что для любой недистрибутивной полумодулярной диаграмму переходов с произвольным рангом детонантности можно построить моделирующую ее диаграмму, в которой ранг детонантности не превышает 2.

Существует также схема реализации детонантной переменной ранга два, содержащая элемент 2ИЛИ-НЕ2. Она строится приблизительно также как и схема переменной для простых дистрибутивных схем.

Таким образом справедлива следующая теорема.

Теорема 5.5. *Система из элементов 2И-НЕ2 и 2ИЛИ-НЕ2 функционально полна в классе полумодулярных схем.*

Доказательство можно найти в [2].

Вопрос о минимальности базиса реализации представляет чисто теоретический интерес. Наиболее предпочтительный базис для построения полумодулярных схем состоит из элементов И-ИЛИ-НЕ и чем более функционально мощные элементы, тем более приемлемыми получаются схемы.

5. Моделирование управления

Для решения практических задач синтеза были разработаны методы, позволяющие по формальным описаниям поведения строить корректные описания с помощью введения дополнительных переменных, пригодные для реализации в выбранном базисе, а затем строить самосинхронные схемы, отвечающие некоторым критериям качества.

Эти методы легли в основу алгоритмов системы автоматизированного анализа и синтеза самосинхронных схем “Petrify”, доступную через интернет:

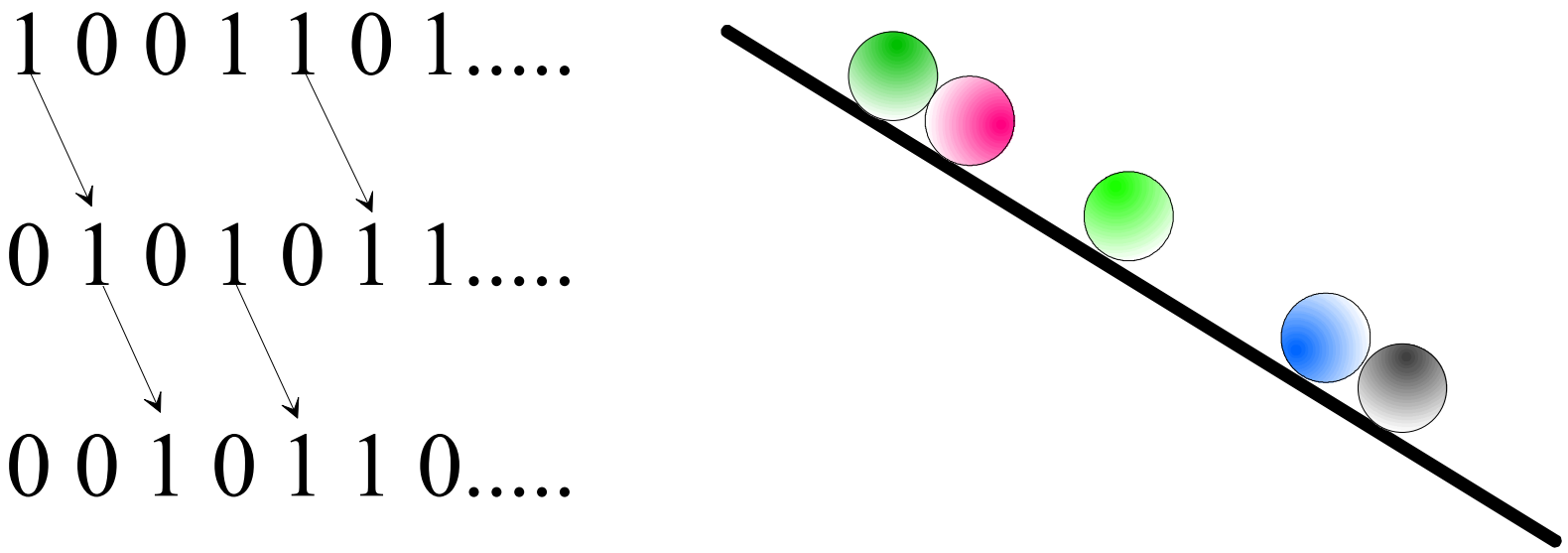
http://www.lsi.upc.edu/~jordicf/petrify/

Для ручного применения они достаточно сложны, так как многие из них требуют большого перебора.

5. Моделирование управления

5.4. Моделирование конвейерных процессов

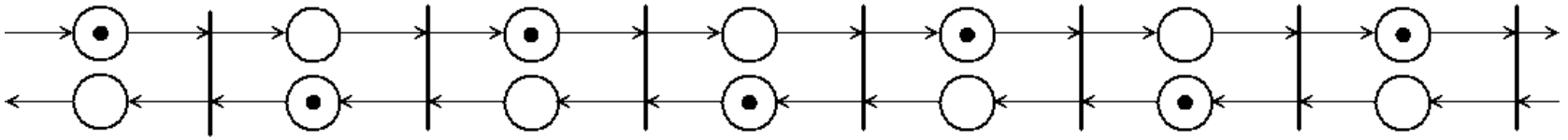
Понятие асинхронного конвейера и его отличие от синхронного



Продвижение информации по конвейеру определяется внутренней динамикой, а не синхротактами.

5. Моделирование управления

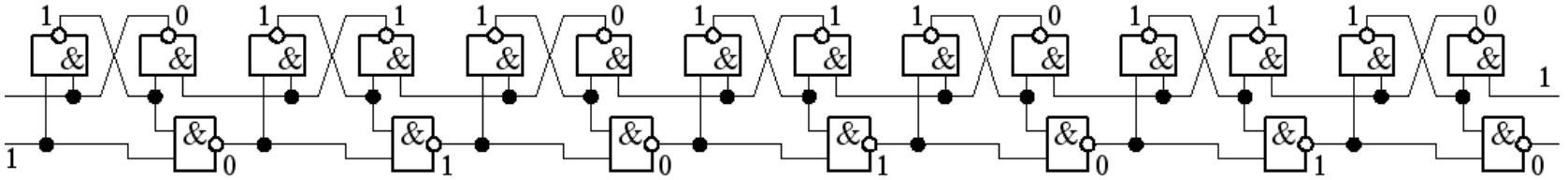
Конвейерная сеть Петри



Допускает плотное заполнение: 1111111

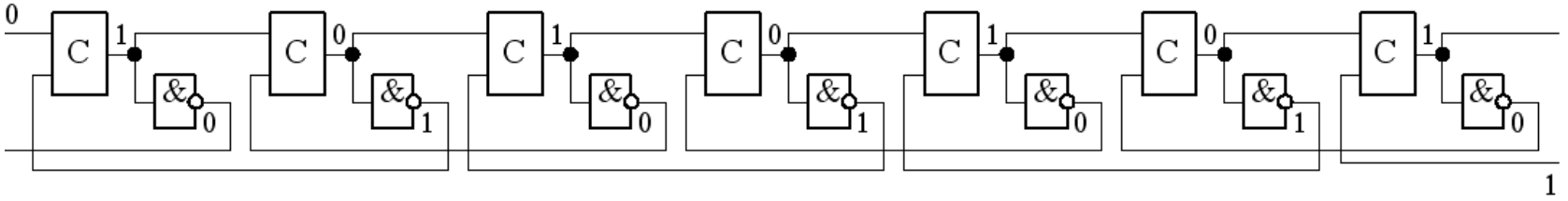
Моделирующие схемы:

а) На ячейках Давида



Полуплотное заполнение: 1Г1Г1Г1. Максимальная пропускная способность достигается при заполнении на 1/4, например, при 011001100.

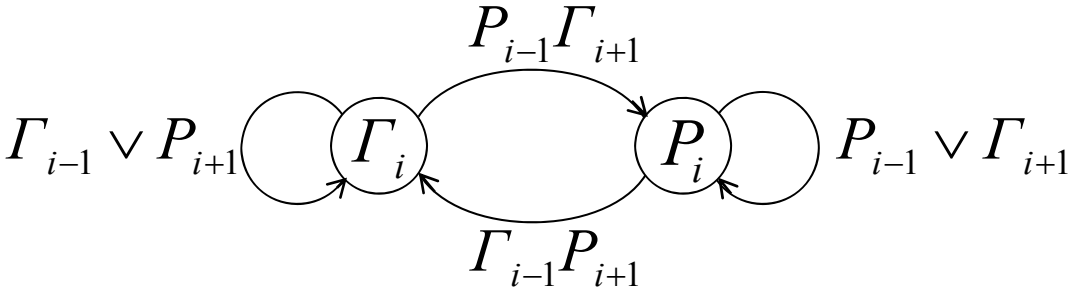
б) На С-элементах (с неплотным заполнением)



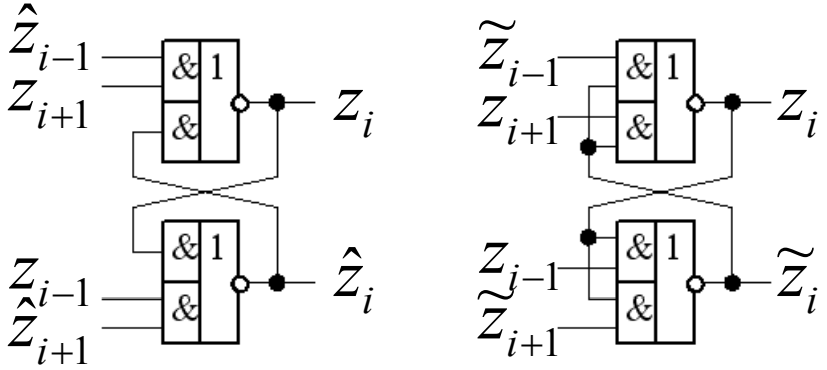
Максимальная пропускная способность достигается при заполнении на 1/4.

5. Моделирование управления

Возможна и другая реализация одномерного конвейера. Рассматривая конвейерное взаимодействие трех последовательных ячеек, каждое из которых имеет два состояния (например, рабочее P и гашения Γ), можно построить граф переходов i-ячейки памяти.



Непосредственно по этому графу можно построить схему ячейки неплотного конвейера:



$z_i = 1$ – это P_i

$z_i = 0$ – это Γ_i

5. Моделирование управления

Отличие конвейера из таких элементов от конвейера из элементов Давида и С-элементов заключается в следующем.

Пусть ячейки находятся в состоянии *Г* и вход конвейера поступает сигнал *Р*. Этот сигнал начинает распространяться по конвейеру с максимальной скоростью, оставляя за собой все ячейки в состоянии *Р*. Только после того, как на входе сигнал *Р* сменится на *Г* по ячейкам конвейера будет распространяться состояние *Г*. В заполненном конвейере состояния *Р* и *Г* чередуются. Такой конвейер может быть использован, например, для управления буферной памятью типа *FIFO*.

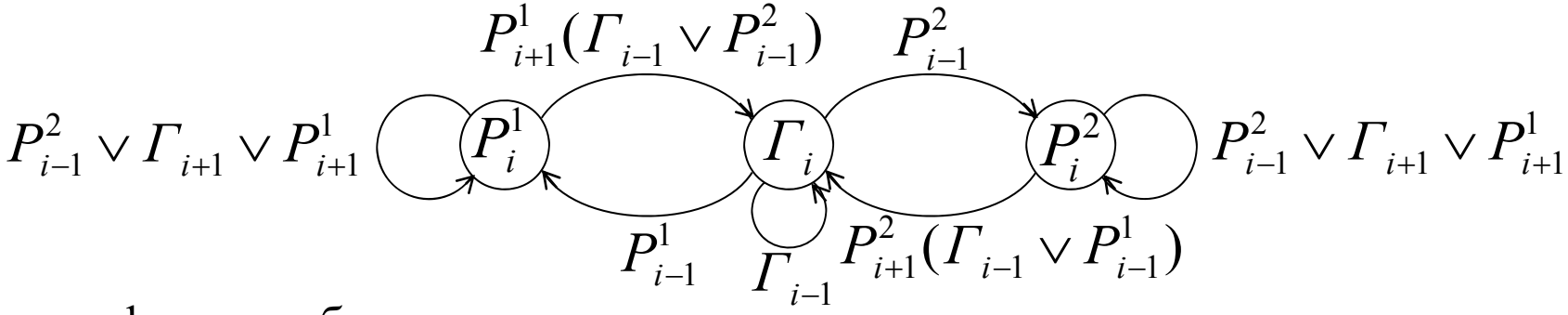
Максимальная пропускная способность такого конвейера также достигается при его заполнении на 1/4.

Существуют ли такие схемы конвейера которые допускают его плотное заполнение? Пропускная способность таких конвейеров должна возрасти до 1/2.

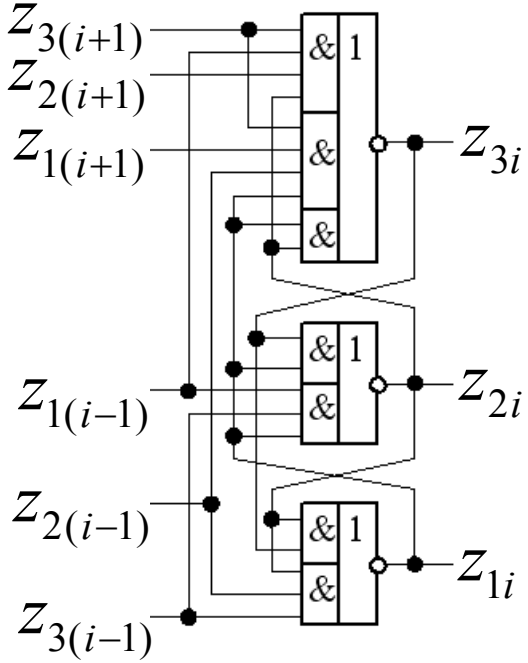
Для построения такого конвейера необходимо различать два смежных рабочих состояния его ячеек. Обозначим их соответственно P^1 и P^2.

5. Моделирование управления

Построим граф переходов такой ячейки.



Это граф трехстабильного триггера.



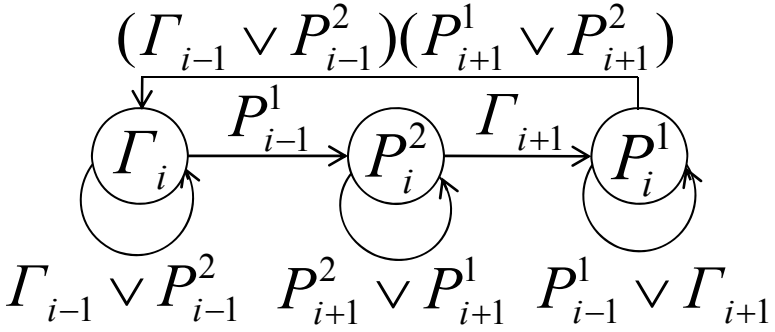
В этой реализации состоянию Γ соответствует набор 110 значений переменных z_1, z_2, z_3, состоянию P^1 – набор 011, состоянию P^2 – набор 101.

Смена состояний осуществляется через наборы, содержащие только одну единицу.

$$P^1_{i+1}(\Gamma_{i-1} \vee P^2_{i-1}) = z_{2(i+1)} z_{3(i+1)} z_{1(i-1)}$$
$$P^2_{i+1}(\Gamma_{i-1} \vee P^1_{i-1}) = z_{1(i+1)} z_{3(i+1)} z_{2(i-1)}$$

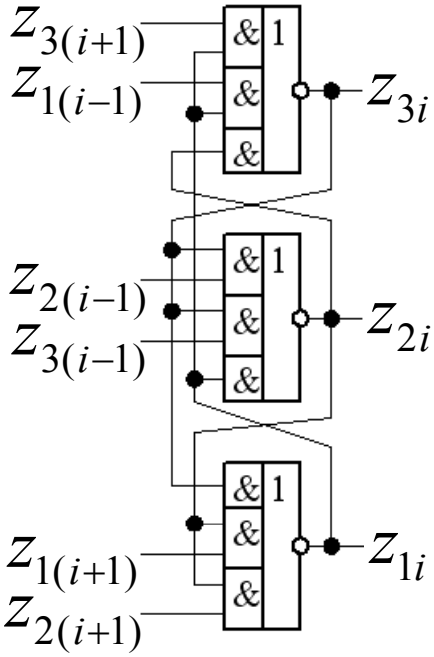
5. Моделирование управления

Построим другой граф переходов ячейки плотного конвейера.



Скорость распространения сигнала по конвейеру в два раза ниже, чем в предыдущем случае

Схема ячейки, соответствующей этому графу существенно проще.



В этой реализации состоянию Γ соответствует набор 001 значений переменных z_1, z_2, z_3,
состоянию P^1 – набор 100,
состоянию P^2 – набор 010.
Смена состояний осуществляется через наборы, содержащие две единицы.

$$(\Gamma_{i-1} \vee P^2_{i-1})(P^1_{i+1} \vee P^2_{i+1}) = \bar{z}_{1(i-1)}\bar{z}_{3(i+1)},$$
$$P^1_{i-1} = \bar{z}_{2(i-1)}\bar{z}_{3(i-1)},\ \Gamma_{i+1} = \bar{z}_{1(i+1)}\bar{z}_{2(i+1)}.$$

5. Моделирование управления

Если некоторый управляющий процесс задан сетью Петри, то вводя в сеть дополнительные цепочки условий, направленные в сторону, обратную распространению процесса, можно получить конвейерную сеть Петри с тем же упорядочением событий, что и исходная.

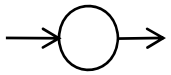
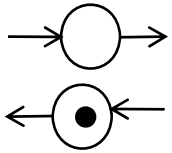
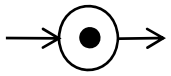
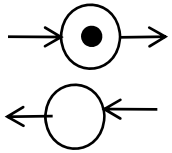
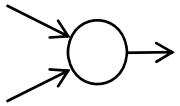
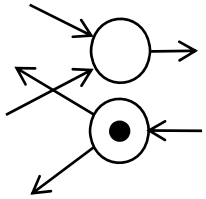
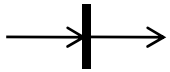
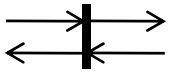
Такая конвейеризация может дать существенное увеличение производительности асинхронного процесса, если в исходной сети Петри нет петель, содержащих менее четырех событий.

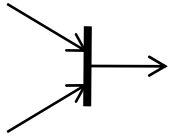
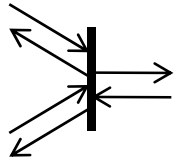
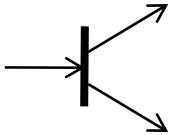
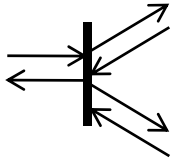
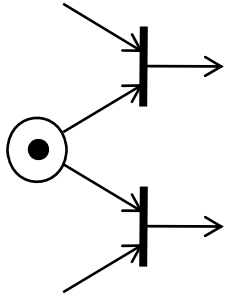
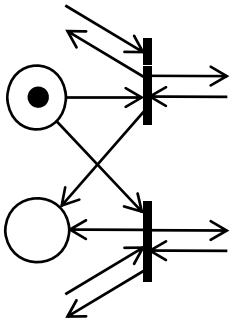
Будем рассматривать только *простые* сети Петри, т.е. устойчивые и безопасные сети, в которых каждое событие не имеет более одного условия, общего с другим событием. Это резко сокращает количество фрагментов, по которым осуществляется конвейеризация сети.

В общем случае формальный переход к конвейерному процессу не гарантирует сохранения устойчивости сети. В таких случаях следует вводить дополнительные условия и события, делающие сеть устойчивой.

Фрагменты сетей Петри и их конвейерные аналоги приведены в следующей таблице.

5. Моделирование управления

Фрагмент сети Петри	Конвейерный аналог
	
	
	
	

Фрагмент сети Петри	Конвейерный аналог
	
	
	

5. Моделирование управления

Процесс асинхронного управления, описываемый сетью Петри, назовем задачей. В конвейерной сети Петри решается сразу много задач управления. При конвейеризации сетей Петри возникают трудности, связанные с необходимостью сохранять свойства устойчивости, безопасности и живости, присущие исходной сети.

Приходится все время следить, чтобы события одной задачи не обгоняли событий другой задачи, чтобы правильно работали циклы и не возникали дедлоки и т.д.

Заметим, что построенные по обычным сетям Петри моделирующие схемы из элементов Давида обладают свойством конвейера. И если обычные (неконвейерные) сети Петри дополнить средствами решения проблем конвейеризации, то построенные по ним моделирующие схемы будут схемами конвейерного управления.

Таким образом, построение конвейерных схем Петри оправдано лишь для выявления проблем конвейеризации.

6. Композиция самосинхронных схем

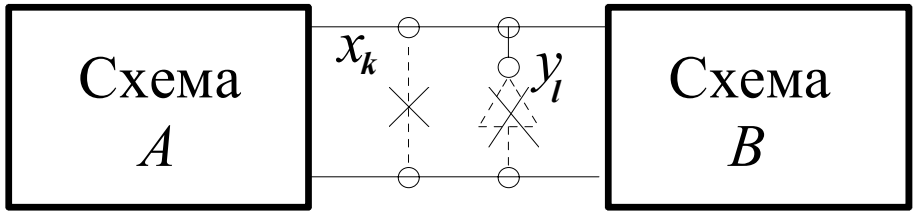
Специфика самосинхронных схем позволяет использовать методы композиции, нетипичные для синхронных и классических асинхронных схем.

Теорема 6.1. *Пусть имеются две полумодулярные схемы Маллера A и B такие, что в схеме A, можно выделить элемент* x_k *и элементы* $x_i = f_i(x_k, X)$, $1 \le i \le n$, *а в схеме B – инвертор* y_l *и элементы* $y_j = \varphi_j(y_l, Y)$, $1 \le j \le m$. *Тогда схема C, представляющая собой композицию схемы A и схемы B без инвертора* y_l, *такая что* $x_i = f_i(x_k \equiv \bar{y}_l, X)$, $y_j = \varphi_j(y_l \equiv x_k, Y)$, $1 \le i \le n$, $1 \le j \le m$, *также является полумодулярной.*

Эта теорема известна как теорема Маллера о соединении полумодулярных схем. Ее смысл может быть объяснен следующим образом. В схеме *A* выделяем провод между выходом какого-либо элемента x_i и входами *n* элементов с которыми он связан, и разрываем его, образуя два полюса. В схеме *B* выделяем инвертор y_l и удаляем его. После этого соединяем схемы так, как это показано на рисунке. Получившаяся схема также является полумодулярной.

Доказательство основано на том, что такое соединение не нарушает свойства полумодулярности.

6. Композиция самосинхронных схем



Если инвертор в схеме *B* отсутствует, то в ней всегда можно отсоединить провод от выхода какого-либо элемента и вставить между проводом и этим выходом два последовательно соединенных инвертора, так как пара инверторов представляет собой просто задержку, которая не оказывает влияния на поведение полумодулярной схемы.

Теорема 6.2. *Пусть имеются две полумодулярные схемы Маллера A и B такие, что в схеме A, можно выделить элемент* x_k *и элементы* $x_i = f_i(x_k, X)$, $1 \le i \le n$, *а в схеме B – элемент* y_l *и элементы* $y_j = \varphi_j(y_l, Y)$, $1 \le j \le m$. *Тогда схема C, представляющая собой композицию схемы A и схемы B без элемента* y_l, *такая что* $x_i = f_i(x_k \equiv \bar{y}_l, X)$, $y_j = \varphi_j(y_l \equiv x_k, Y)$, $1 \le i \le n$, $1 \le j \le m$, *также является полумодулярной.*

6. Композиция самосинхронных схем

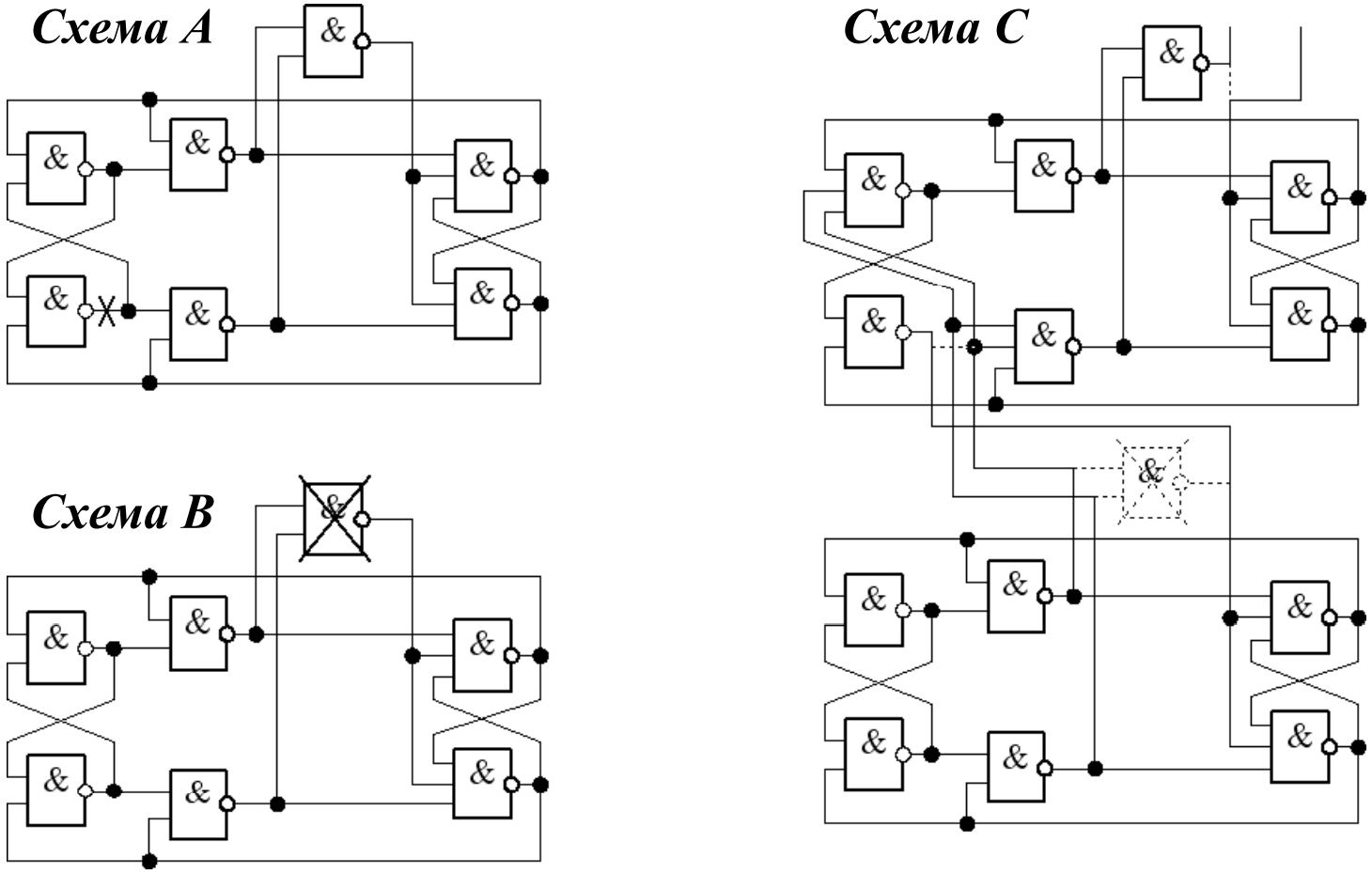
Это обобщение теоремы Маллера можно доказать следующим образом. Представим элемент y_l в виде безынерционного логического элемента и инвертора, что не противоречит гипотезе о задержках. В соответствии с теоремой 6.1 инвертор удаляем. Остается безынерционная функция $\overline{y}_l$, которая должна быть реализована на входах элементов $x_i = f_i(x_k \equiv \overline{y}_l, X)$ схемы A. Далее, отсоединенный выход элемента x_k в схеме A соединяется со входами y_l элементов $y_j = \varphi_j(y_l \equiv x_k, Y)$ схемы B. Все указанные соединения не нарушают свойства полумодулярности получившейся схемы.

В результате такой композиции сложность композируемой схемы может оказаться меньше суммарной сложности композируемых схем (например, по числу элементов). Кроме того, получившаяся схема может иметь большее быстродействие.

Такие эффекты, как правило, недостижимы в традиционной схемотехнике.

Проиллюстрируем применение такой композиции па примере построения счетчика из двух счетных триггеров.

6. Композиция самосинхронных схем



Теорему 6.2 можно расширить до замещения в полумодулярных схемах целых подсхем. Рассмотрим пример возможной композиции двух разрядов счетчика.

6. Композиция самосинхронных схем

Схема A

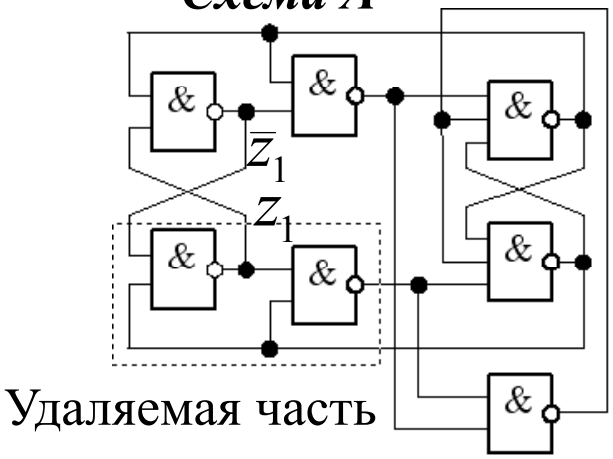


Схема B

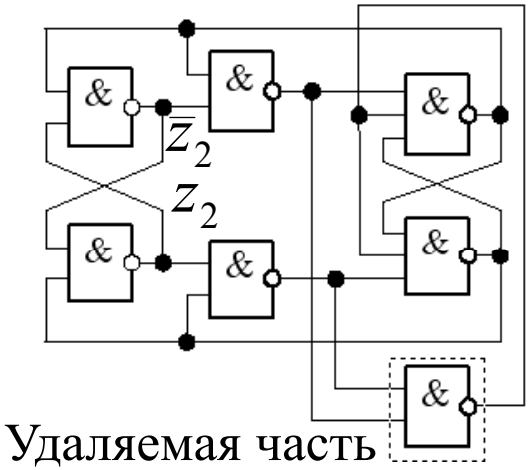
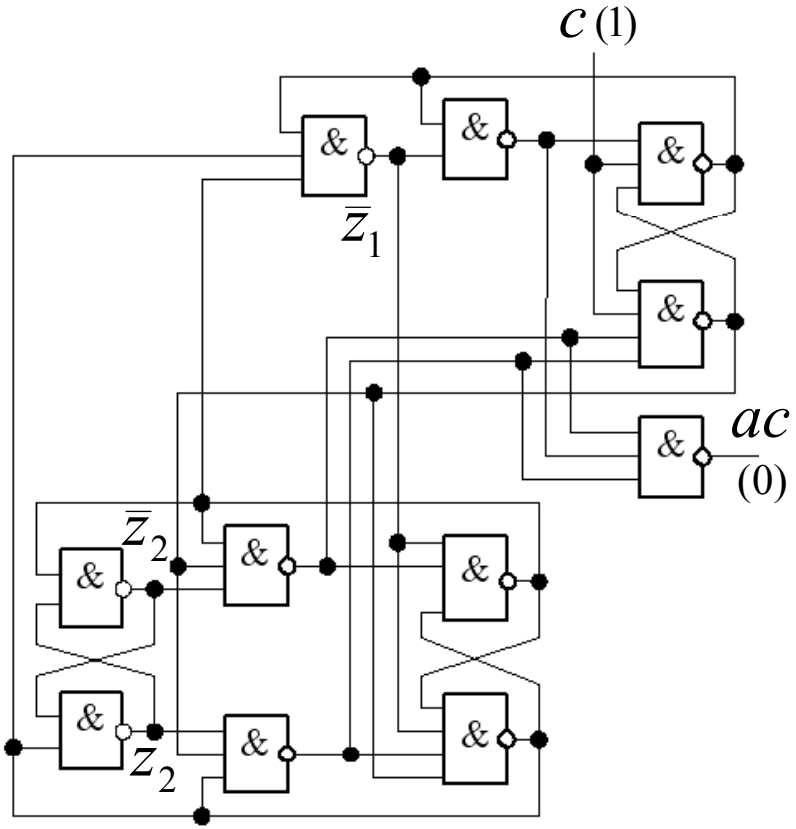


Схема C



6. Композиция самосинхронных схем

Особенностью многих полумодулярных схем является парофазное представление сигналов (переменных).

Будем называть схему *монотранзитной*, если при любом переходе все ее парафазные переменные имеют одно и то же транзитное состояние (нулевое или единичное), и *совершенной*, если транзитные состояния переменных не вызывают возбуждения элементов схемы. Композицию таких схем можно строить в соответствии со следующей теоремой.

Теорема 6.3. *Пусть имеются две полумодулярные монотранзитные совершенные схемы A и B такие, что в схеме A, можно выделить совершенную подсхему (триггер) с парофазным выходом* $x_k, \hat{x}_k$ *и триггеры с парофазными выходами* $x_i, \hat{x}_i$ и функциями возбуждения

$$S_i^A = S_i^A(x_k, \hat{x}_k, X),\ R_i^A = R_i^A(x_k, \hat{x}_k, X),\ 1 \le i \le n,$$

а в схеме B – совершенную подсхему (триггер) $y_l, \hat{y}_l$ *с функциями возбуждения* S_l^B, R_l^B *и триггеры с парофазными выходами* $y_j, \hat{y}_j$ *и функциями возбуждения*

$$S_j^B = S_j^B(y_l, \hat{y}_l, Y),\ R_j^B = R_j^B(y_l, \hat{y}_l, Y),\ 1 \le j \le m.$$

6. Композиция самосинхронных схем

Тогда схема C, представляющая собой композицию схемы A и схемы B без триггера $y_l, \hat{y}_l$, *такая что*

$$S_i^A = S_i^A(x_k \equiv S_l^B, \hat{x}_k \equiv R_l^B, X),\ R_i^A = R_i^A(x_k \equiv S_l^B, \hat{x}_k \equiv R_l^B, X),$$
$$S_j^B = S_j^B(y_l \equiv x_k, \hat{y}_l \equiv \hat{x}_k, Y),\ R_j^B = R_j^B(y_l \equiv x_k, \hat{y}_l \equiv \hat{x}_k, Y),$$
$$1 \le i \le n,\ 1 \le j \le m,$$

также является полумодулярной.

Эта теорема является аналогом теоремы 6.2 для совершенных реализаций, в котором вместо элемента удаляется подсхема, состоящая из триггера и его функций возбуждения.

Рассмотренные методы композиции, являющиеся по существу обобщениями теоремы Маллера, не покрывают всех возможностей соединения полумодулярных схем.

7. Организация интерфейсов

Схемы, независящие от скорости (например, полумодулярные) , сохраняют свою работоспособность при любых конечных задержках элементов до тех пор, пока выполняется принятая гипотеза о задержках в проводах.

Схемы, нечувствительные к задержкам (в элементах и проводах), практически не могут быть реализованы.

Возникает проблема длинных проводов.

Для решения этой проблемы предлагается всю площадь кристалла, разбивать на множество *эквихронных* зон. Во всех точках каждой эквихронной зоны время предполагается одинаковым, т.е. задержками в проводах внутри эквихронной зоны можно пренебречь.

Разработанные самосинхронные устройства или их модули располагаются в экихронных зонах (каждый в своей), а взаимодействие между различными устройствами или модулями осуществляется по длинным линиям на основе интерфейсов.

7. Организация интерфейсов

В широком смысле под интерфейсом (или сопряжением) понимается совокупность правил, обеспечивающих совместное функционирование аппаратных и программных средств. Иногда интерфейсом называют и сами технические средства, созданные по этим правилам.

Интерфейсы принято разделять на три уровня: *механический* (конструктивный), *электрический* и *логический*. Нас будет интересовать логический уровень, обеспечивающий координацию информационных потоков между устройствами. На этом уровне реализуется *протокол* информационного обмена.

Под протоколом обычно понимают свод правил, обеспечивающий упорядоченный информационных обмен между объектами. Теория протоколов изучает способы задания, анализа и синтеза интерфейсного взаимодействия.

Протоколы информационного обмена обычно ориентированы на выполнение множества функций, основными из которых являются функции *передачи* и *приема* информационных наборов. Именно эти функции мы и будем рассматривать, оставив в стороне функции назначения ведущего и адресации ведомого.

7. Организация интерфейсов

7.1. Передача-прием наборами ПК

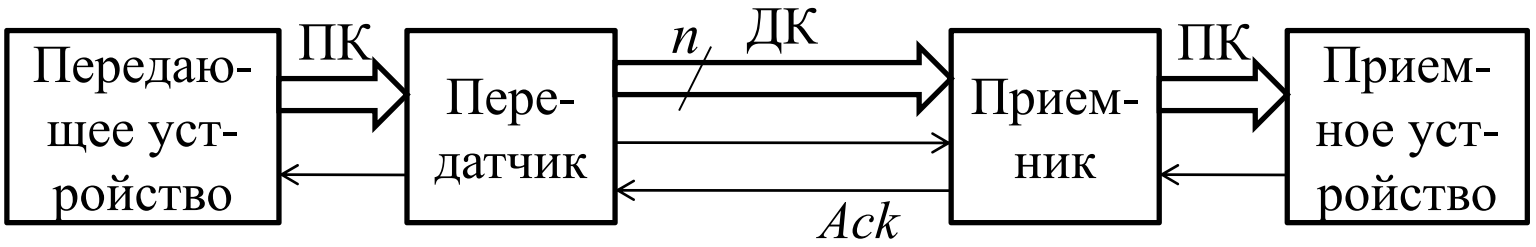
Желательно, чтобы взаимодействие всех блоков было построено по принципу конвейера. Для этого все блоки должны иметь регистры данных. Если передатчик может работать на несколько приемников, то драйверы линий связи должны иметь трехстабильный выход. При выбранном приемнике они могут быть переведены в двухстабильный режим.

Достоинство. Это самый простой способ передачи. Легко строятся тестеры парофазного кода. Простой протокол передачи-приема.

Недостатки. Для передачи наборов n-разрядного двоичного кода требуется $(2n+1)$ линий связи. Передача одного набора осуществляется за четыре обращения к линиям связи (мала скорость передачи).

7. Организация интерфейсов

7.2. Передача наборами безызбыточного двоичного кода



Дополнительная линия данных от передатчика к приемнику необходима для передачи нулевого кода. Протоколы взаимодействия передающего устройства с передатчиком и приемника с приемным устройством тривиальны (конвейеризированное хендшейк взаимодействие).

Для организации взаимодействия, независящего от задержек линии связи, используется побитное квитирование разрядов двоичного кода.

Протокол передачи-приема по линиям связи:

1) На информационных линиях связи находится набор из всех нулей. Передатчик устанавливает на линиях связи двоичный набор и после проверки установки посылает 1 по дополнительной линии.

7. Организация интерфейсов

2) Приемник, получив 1 по доп. линии, принимает двоичный код в регистр и после приема каждого разряда со значением 1 квитирует принятые единицы, переводя соответствующие линии в состояние 0, и посылает ответ *Ack=1*.

3) Передатчик, получив сигнала *Ack=1*, проверяет все нули на основных линиях и снимает 1 с доп. линии.

4) Приемник, получив 0 по доп. линии, заканчивает прием информации и отвечает изменением состояния сигнала *Ack=0*.

5) Передатчик в ответ на сигнал *Ack=0* пытается установить на основных линиях единичный набор. При этом на неквитированных линиях устанавливается инверсный набор переданного двоичного кода. Передатчик проверяет установку инверсного кода, после чего устанавливает 1 на доп. линии.

6) Приемник, получив 1 по доп. линии, снимает квитирование основных линий, что вызывает появление на них всех единиц, и отвечает изменением сигнала *Ack=1*.

7) Передатчик, получив сигнал *Ack=1*, проверяет все единицы на информационных линиях, после чего устанавливает на них все нули, включая доп. линию.

8) Приемник проверяет все нули на информационных линиях и отвечает изменением значения сигнала *Ack=0*.

Достоинство. Для передачи наборов *n*-разрядного двоичного кода требуется (*n*+2) линий связи (минимальная избыточность), т.е. обеспечивает минимальную площадь, занимаемую линиями связи.

Недостатки. Передача одного набора двоичного кода осуществляется за 8 обращений к линиям связи (в два раза хуже, чем при передаче наборов ПК). Усложненная схема управления передатчиком и приемником.

7.3. Передача наборами ССК, отличного от ПК

Код с идентификатором (КИ, или код Бергера) или оптимальный равновесный код (ОРК) могут использоваться лишь для сокращения числа линий связи. Для передачи наборов *n*-разрядного двоичного кода они требуют соответственно

7. Организация интерфейсов

$(n+]\log_2(n+1)[+1)$ и $(n+]\log_2 n[+1)$ линий связи. Эти коды, как и ПК, являются двухфазными и не выигрывают по скорости передачи, однако их использование в значительной степени усложняет схемы передатчика и приемника, так как требует включения в них схем кодирования, тестера и декодирования. Даже при небольших значениях n эти дополнительные схемы могут иметь очень высокую сложность.

Покажем на простом примере возможность реализации интерфейсных устройств, использующих ОРК.

Полубайтный обмен. Для передачи 16 возможных комбинаций четырехразрядного двоичного кода можно использовать ОРК $C_6^3 = 20$ и передавать рабочие комбинации по 6 информационным линиям.

Один из вариантов кодирования приведен в таблице. Нулевой набор являются спейсером. Следующие за ним 16 наборов кодируют двоичные числа от 0 до 15 и являются рабочими. Остальные 4 набора не используются. При желании их можно использовать для кодирования управляющих символов.

7. Организация интерфейсов

$p_1p_2p_3p_4p_5p_6$	$y_1y_2y_3\hat{y}_3y_4\hat{y}_4$
0 0 0 0 0 0	0 0 0 0 0 0
0 0 0 1 1 1	0 0 0 1 0 1
0 0 1 0 1 1	0 0 1 0 0 1
0 0 1 1 0 1	0 0 0 1 1 0
0 0 1 1 1 0	0 0 1 0 1 0
1 0 0 0 1 1	1 0 0 1 0 1
1 0 1 0 1 0	1 0 1 0 0 1
1 0 0 1 0 1	1 0 0 1 1 0
1 0 1 1 0 0	1 0 1 0 1 0
0 1 0 1 0 1	0 1 0 1 0 1
0 1 0 0 1 1	0 1 1 0 0 1
0 1 1 1 0 0	0 1 0 1 1 0
0 1 1 0 1 0	0 1 1 0 1 0
1 1 0 0 0 1	1 1 0 1 0 1
1 1 0 0 1 0	1 1 1 0 0 1
1 1 0 1 0 0	1 1 0 1 1 0
1 1 1 0 0 0	1 1 1 0 1 0
1 0 0 1 1 0	1 0 0 0 1 1
1 0 1 0 0 1	1 0 1 1 0 0
0 1 0 1 1 0	0 1 1 1 0 0
0 1 1 0 0 1	0 1 0 0 1 1

Не используются

Только этот вариант кодирования дает простые функции декодирования.

Непосредственно по таблице получаем:

$$y_1 = p_1,\ y_2 = p_2,$$

$$y_3 = \overline{p_1p_3 \vee p_2p_5 \vee p_3p_5},$$

$$\hat{y}_3 = \overline{p_1p_6 \vee p_2p_4 \vee p_4p_6},$$

$$y_4 = \overline{p_1p_4 \vee p_2p_3 \vee p_3p_4},$$

$$\hat{y}_4 = \overline{p_1p_5 \vee p_2p_6 \vee p_5p_6}.$$

7. Организация интерфейсов

Схема декодирования

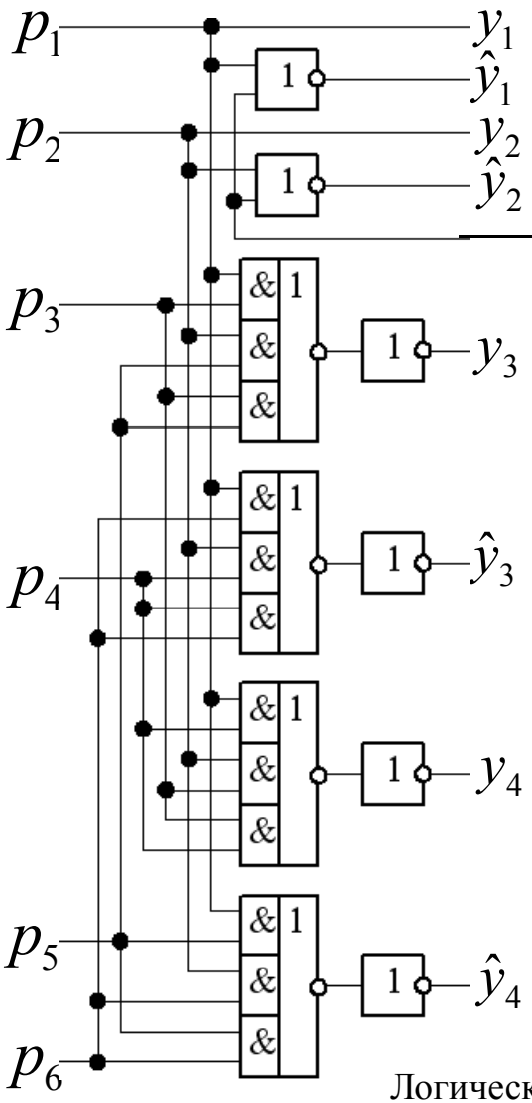
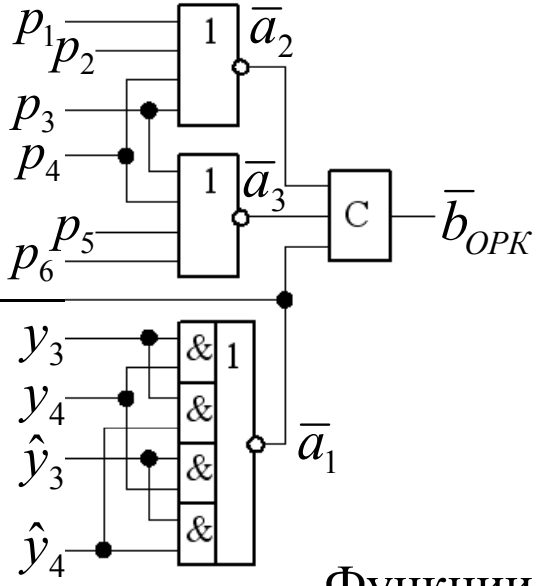


Схема тестера ОРК



Функции тестера:

$$\bar{a}_1 = \overline{y_3 y_4 \vee y_3 \hat{y}_4 \vee \hat{y}_3 y_4 \vee \hat{y}_3 \hat{y}_4},$$

$$\bar{a}_2 = \overline{p_1 \vee p_2 \vee p_3 \vee p_4},\ \bar{a}_3 = \overline{p_3 \vee p_4 \vee p_5 \vee p_6},$$

$$\bar{b}_{OPK} = \bar{a}_1 \bar{a}_2 \bar{a}_3 \vee \bar{b}_{OPK}(\bar{a}_1 \vee \bar{a}_2 \vee \bar{a}_3).$$

7. Организация интерфейсов

Сигнал $\overline{a}_1$ тестера принимает значение 0, когда на входах приемника установлен набор ОРК. Этот сигнал используется для разрешения преобразования однофазных сигналов p_1, p_2 на входе декодера в парофазные $y_1, \hat{y}_1$ и $y_2, \hat{y}_2$ на его выходах.

На элементах $\overline{a}_2, \overline{a}_3$ проверяется установка спейсера на входах приемника.

Декодированный парофазный код записывается в регистр приемника. Выходной сигнал индикатора этого регистра и выходной сигнал $\overline{b}_{ОРК}$ участвуют в выработке сигнала ответа передатчику об установке и приеме очередного рабочего набора и о приеме спейсера.

В регистр приемника может быть записан новый парофазный код только после того, как приемное устройство перепишет предыдущий. Таким образом организуется пайплайновое взаимодействие приемника с приемным устройством.

Кодирующее устройство передатчика имеет почти такую же структуру и не содержит тестера *ОРК*.

7. Организация интерфейсов

Побайтный обмен. Может быть построен очевидным образом из двух схем полубайтного обмена. В этом случае для передачи байта информации потребуется 12 информационных линий и одна линия для сигналов-квитанций.

Передача-прием в изменениях. Этот способ удобен при передаче пакетов информации. Двухфазную дисциплину передачи наборов можно заменить однофазной, если не передавать спейсер. Это приводит к удвоению скорости передачи.

В качестве спейсера на передающем и приемном конце используется предыдущий набор. Для этого в передатчик и приемник должны быть введено по одному дополнительному регистру, хранящему предыдущий набор.

Наборы, передаваемые по линиям отличаются от наборов ССК, так как передается не сам набор, а лишь изменения сигналов на тех линиях, которые указывает набор *ССК*. На приемном конце поразрядно суммируя по модулю два два смежных набора, выделяют набор ССК. Каждое изменение сигнала ответа передающего устройства должно вызывать передачу следующего набора.

Логическое Проектирование Асинхронных Схем

Лекция 5

8. Самосинхронная схемотехника

В этом разделе рассмотрим некоторые самосинхронные схемы, которые могут оказаться полезными в практике проектирования. Любая самосинхронная схема может работать в синхронном окружении. Для этого на вход фазового сигнала подаются синхросигналы, а выход фазового сигнала либо не используется, либо используется для контроля правильности функционирования схемы (если схема отвечает в каждом синхротакте изменением этого сигнала, то она функционирует правильно).

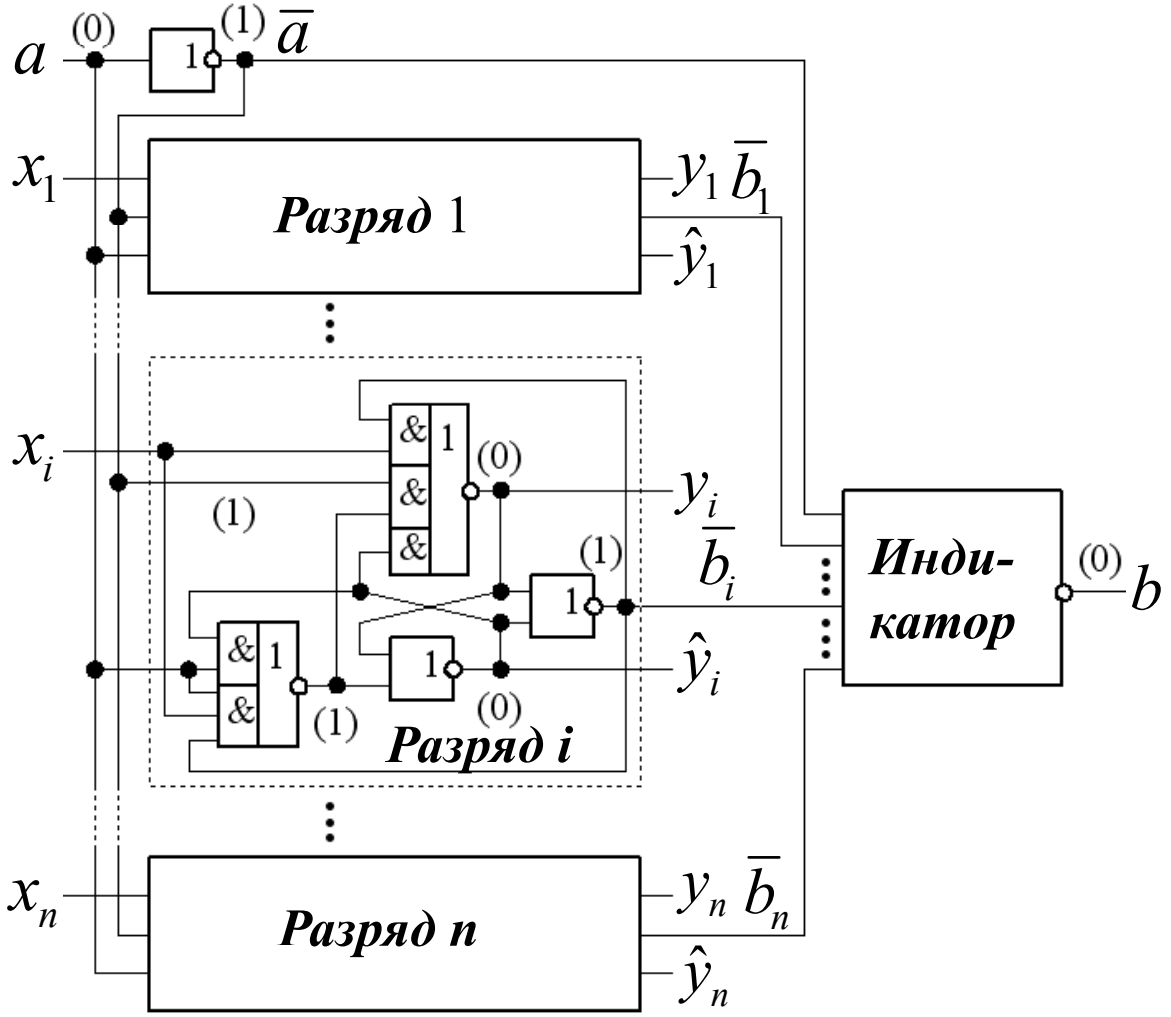
8.1. Регистр-преобразователь однофазных сигналов в парафазные

Может использоваться для стыковки самосинхронных устройств с устройствами других типов. Основные требования к преобразователю:

1) Входные данные принимаются по сигналу готовности (сопровождения).
2) После однократного приема данных преобразователь становится нечувствительным к информационным входам (отсечка от входов).
3) При снятии сигнала готовности информационные выходы преобразователя устанавливаются в состояние спейсера.

8. Самосинхронная схемотехника

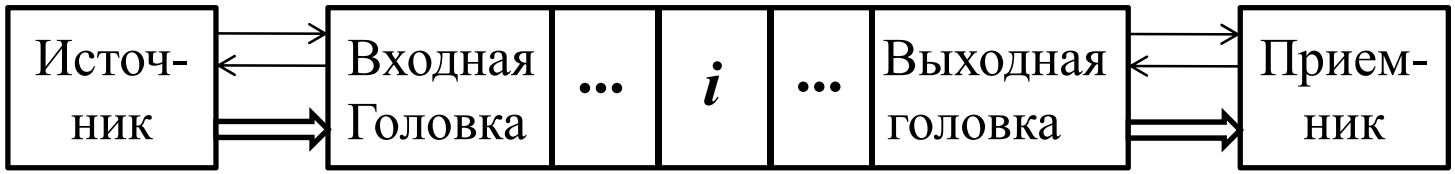
Схема регистра-преобразователя



8. Самосинхронная схемотехника

8.2. Конвейерные регистры

Конвейерный регистр – это последовательный регистр, который благодаря своим динамическим свойствам может выполнять роль буфера (накопителя) между источником и приемником двоичной информации. Его структурная схема имеет вид:



Бит информации, выданный источником, пробегает по всем ячейкам регистра. Источник и приемник могут работать независимо до тех пор, пока регистр либо полностью не освободится, либо полностью заполнится.

Параметры конвейерного регистра:

1) максимальное число бит информации, которое может быть записано в *n*-разрядный регистр (емкость);
2) пропускная способность (величина, обратная интервалу между битами).

8. Самосинхронная схемотехника

8.2.1. Неплотные регистры

Граф переходов i-й ячейки

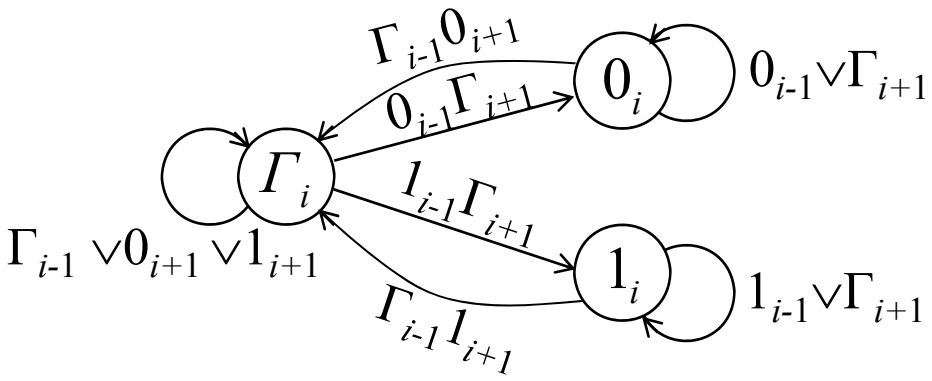
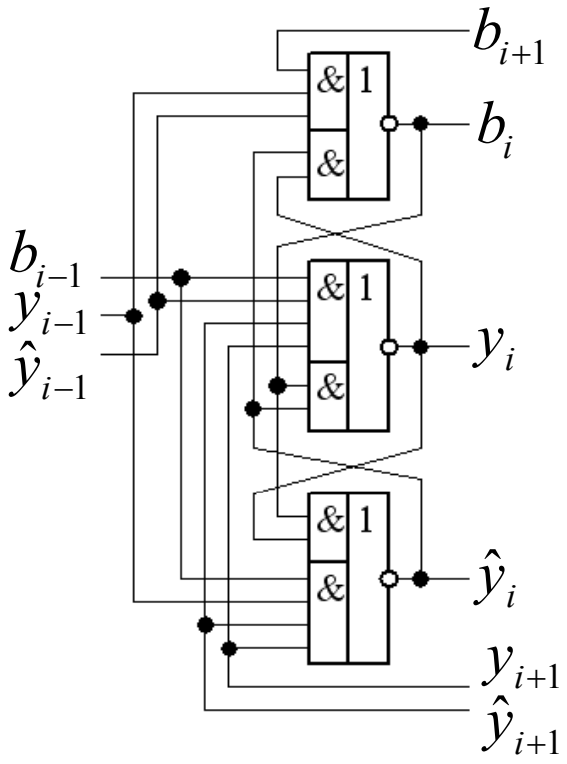


Схема ячейки



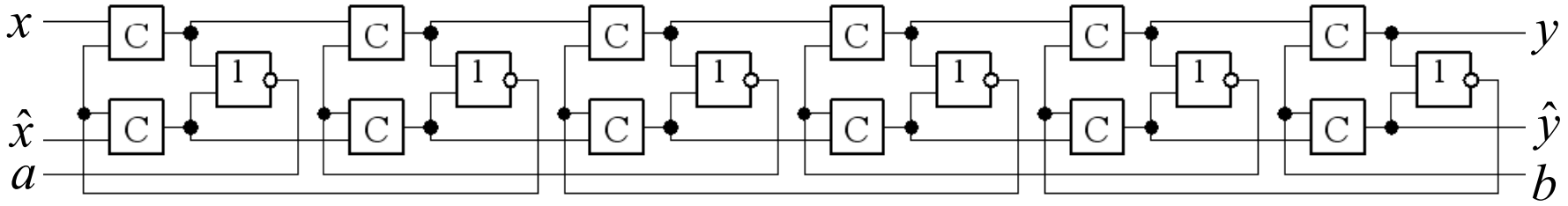
Кодировка: 011 – Г; 101 – 0; 110 – 1.
В транзитном состоянии два нуля.

Максимальное заполнение – $]n/2[$.

Пропускная способность $-1/(4t)$, где $t = 2\tau_g$.
Достигается при повторяющейся конфигурации:
...РРГГРРГГ... , т.е. при заполнении на 25%.

8. Самосинхронная схемотехника

Другой вариант схемы неплотного регистра



8.2.2. Полуплотные конвейерные регистры

Граф переходов i-й ячейки

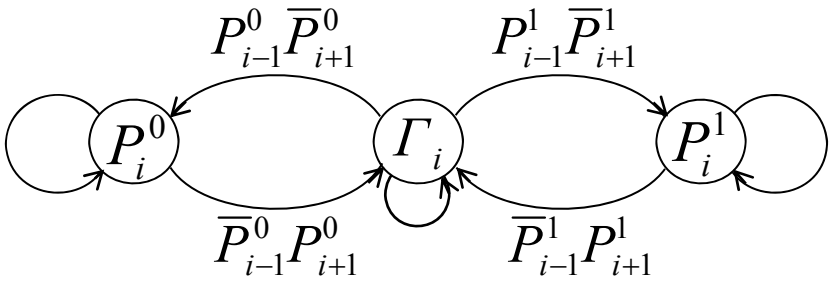


Таблица кодирования

	x_i	$\hat{x}_i$	$\hat{y}_i$	y_i
Γ_i	0	1	1	0
P_i^1	1	0	1	0
P_i^0	0	1	0	1
–	1	0	0	1

Учитывая, что $\overline{P}_i^0 = \Gamma_i \vee P_i^1 = \hat{y}_i$, $\overline{P}_i^1 = \Gamma_i \vee P_i^0 = \hat{x}_i$
и $P_i^0 = x_i$, $P_i^1 = y_i$, построим схему ячейки регистра.

8. Самосинхронная схемотехника

Схема ячейки

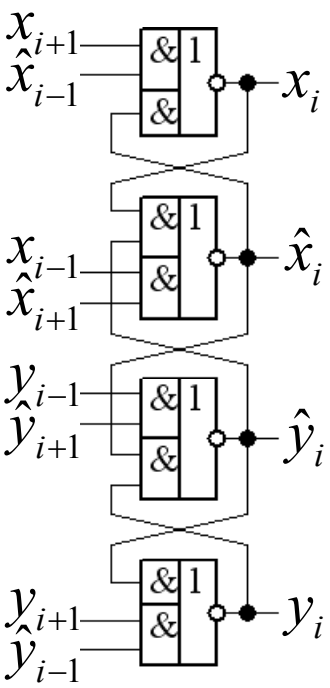


Схема имеет три устойчивых состояния.

Характеристики полуплотного конвейера:

– информационная емкость в пределах от $n/2$ до n; среднее значение для бернулиевской последовательности – $3n/4$;

– пропускная способность от $1/(4t)$ при конфигурации $\ldots\Gamma_i\Gamma_{i+1}P^{\sigma}_{i+2}P^{\sigma}_{i+3}\Gamma_{i+4}\Gamma_{i+5}P^{\sigma}_{i+6}P^{\sigma}_{i+7}\ldots$ до $1/(3t)$ при конфигурации

$$\ldots\Gamma_i P^{\sigma}_{i+1}P^{\sigma}_{i+2}\Gamma_{i+3}P^{\bar{\sigma}}_{i+4}P^{\bar{\sigma}}_{i+5}\Gamma_{i+6}P^{\sigma}_{i+7}P^{\sigma}_{i+8}\ldots;$$

среднее значение пропускной способности – $7/(24t)$; $t = 2\tau_g$.

Начальная установка конвейера может быть осуществлена методом выкачки информации.

8. Самосинхронная схемотехника

8.2.3. Плотные конвейерные регистры

Плотный регистр целесообразен в случае, если схема его разряда проще двух разрядов неплотного регистра.

Граф переходов i-й ячейки

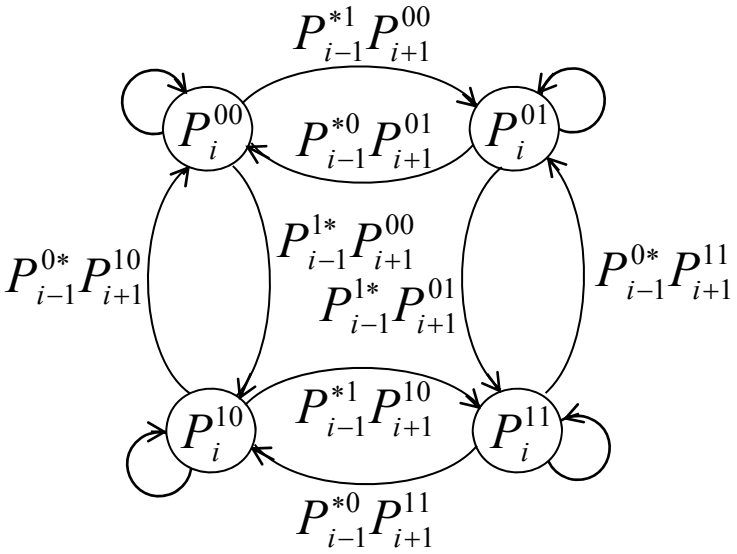


Таблица кодирования

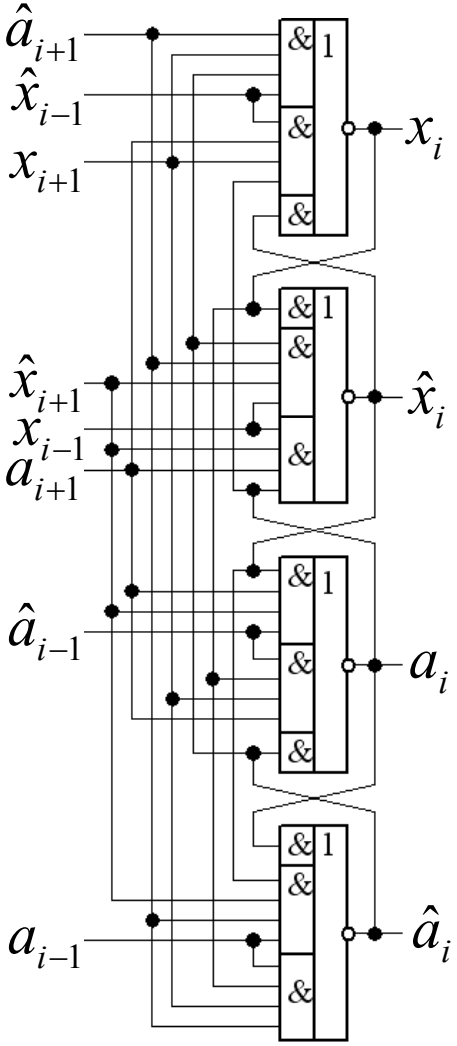
	$\hat{x}_i$	x_i	$\hat{a}_i$	a_i
P_i^{00}	1	0	1	0
P_i^{10}	0	1	1	0
P_i^{01}	1	0	0	1
P_i^{11}	0	1	0	1

Первый верхний индекс в состоянии ячейки – значение бита информации, а второй – значение метки. Приняты следующие обозначения:

$$P_{i-1}^{*\sigma} = P_{i-1}^{1\sigma} \vee P_{i-1}^{0\sigma};\ P_{i-1}^{\sigma*} = P_{i-1}^{\sigma 1} \vee P_{i-1}^{\sigma 0}.$$

8. Самосинхронная схемотехника

Схема ячейки



После включения питания начальная установка регистра может осуществляться выкачкой из него неверной информации.

Поскольку нет состояния гашения в пустом регистре всегда хранится один бит информации.

Пропускная способность, равная $1/(2t) = 1/(4\tau_g)$, достигается при конфигурации, в которой каждый бит информации занимает в регистре по две ячейки.

Существуют более экономичные схемы плотных регистров, но с меньшей пропускной способностью.

Аналогичным образом могут быть построены входные и выходные головки всех типов конвейерных регистров.

8. Самосинхронная схемотехника

8.3. Организация самосинхронной статической памяти

Элемент статической памяти

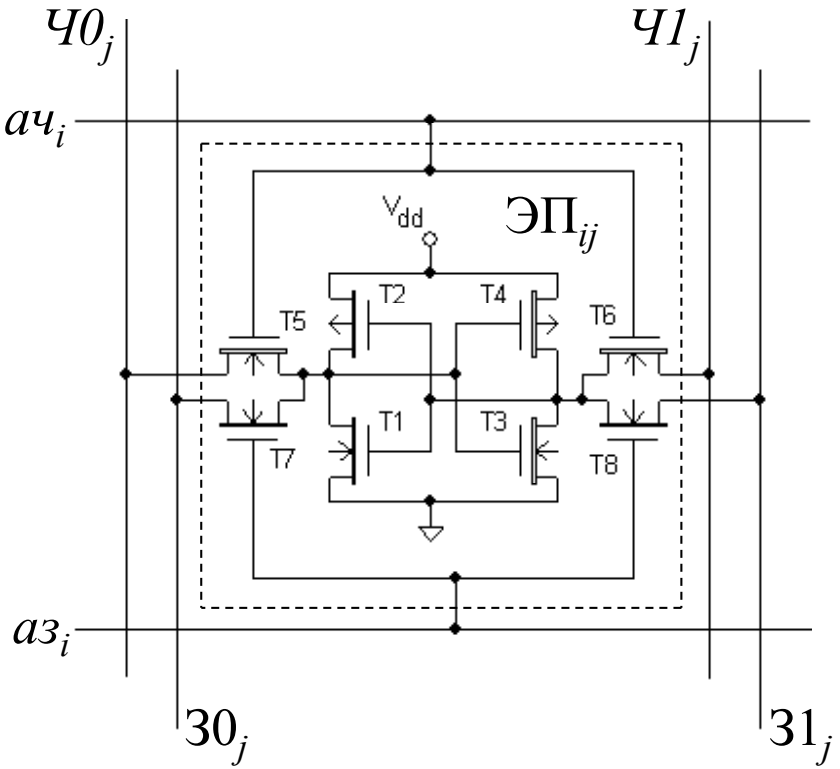
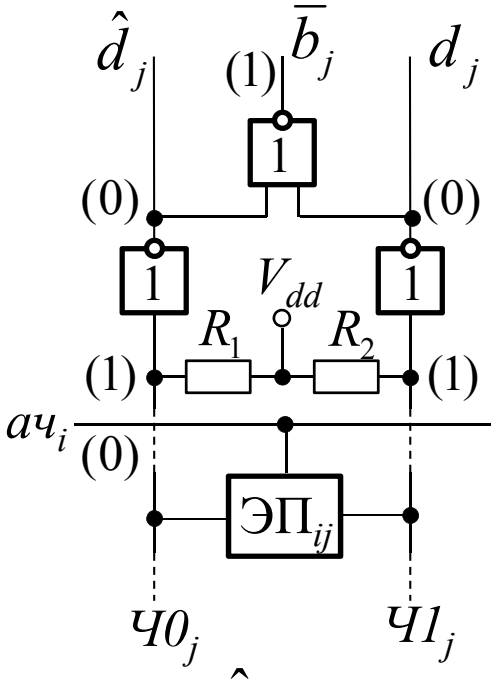


Схема чтения j-го разряда i-го слова

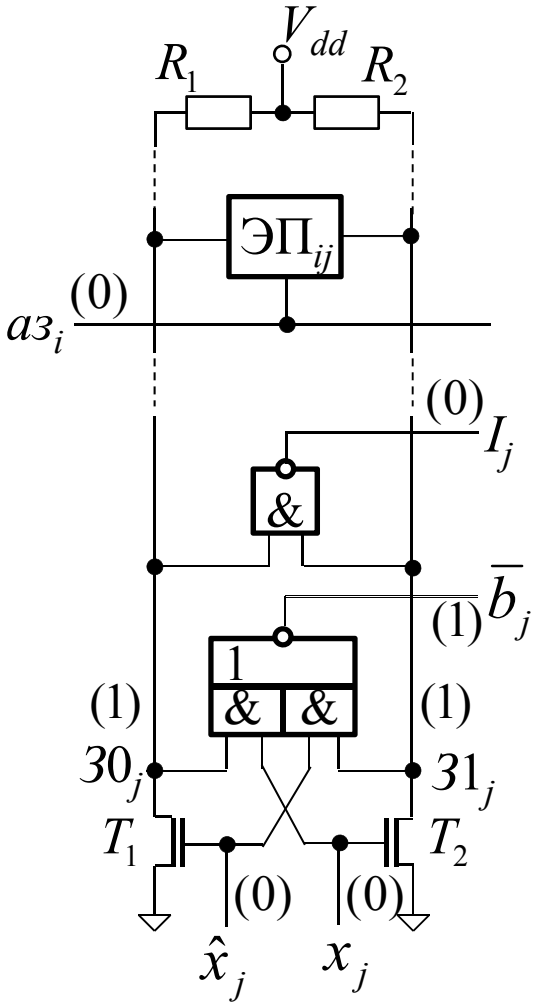


$$\rightarrow +ач_i \rightarrow \hat{d}_j \neq d_j \rightarrow -\bar{b}_j$$

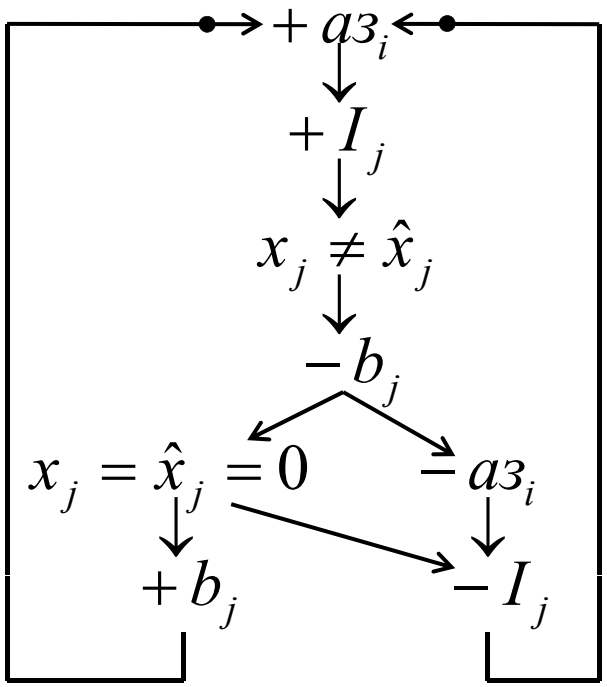
$$+\bar{b}_j \leftarrow \hat{d}_j = d_j \leftarrow -ач_i \leftarrow$$

8. Самосинхронная схемотехника

Схема записи в j-й разряд i-го слова



Сигнальный граф ее работы



Такая статическая память может быть использована, например, для реализации операционных регистров процессоров. Возможно построение многопортовой памяти.

9. Формальные методы синтеза самосинхронных схем

9.1. Реализуемость описаний

Будем использовать язык диаграмм изменений (ДИ) как средство спецификации исходного поведения.

Очевидно, что по любой полумодулярной ДП можно построить ДИ, которая отличается от сигнального графа только тем, что допускает использование вершин типа ИЛИ. Обратный переход возможен лишь для так называемых правильных ДИ, удовлетворяющих двум ограничениям.

1. Поскольку состояния полумодулярной ДП представлены бинарными наборами, то любая последовательность состояний *корректна по переключениям* переменных: для каждой переменной изменения ее значений чередуются (за переходом 0 – 1 следует переход 1 – 0) . Тем же свойством должна обладать и ДИ, соответствующая ДП.

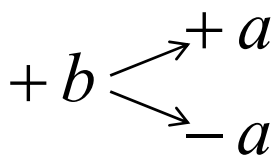
2. В двух параллельных ветвях ДИ не должна изменяться одна и та же переменная, так как переменная не может быть параллельной самой себе (не должно быть *авто-параллельности*).

9. Формальные методы синтеза самосинхронных схем

Некорректность по переключению

$$+a \rightarrow -b \rightarrow +a$$

Авто-параллельность



Определение 9.1.1. ДИ называется *правильной*, если все ее переменные корректны по переключениям и не являются автопараллельными.

Доказано, что языки правильных ДИ и полумодулярных ДП эквивалентны, поэтому некоторые понятия, присущие ДП, распространим на ДИ, например, понятие полного состояния ДП.

В обсуждаемых методах на базис реализации накладывается единственное требование – он должен быть антитонным.

Определение 9.1.1. Состояния *v* и *w* ДИ (ДП) называются *непосредственно конфликтными*, если они представлены одинаковыми наборами с разной разметкой возбуждений.

9. Формальные методы синтеза самосинхронных схем

Свойство 9.1.1. Полумодулярная ДП (ДИ) реализуема схемой, если и только если она не содержит непосредственно конфликтных состояний.

Доказательство очевидно, поскольку существует процедура построения по такой ДИ схемы Маллера.

Следовательно анализ описаний на некорректность связан с нахождением *непосредственно конфликтных состояний.* Однако не только такие состояния являются недопустимыми, но и состояния, которые могут служить причиной их возникновения.

Определение 9.1.2. Состояния *v* и *w* ДИ (ДП) называются *конфликтными*, если они представлены одним и тем же набором двоичных переменных (с одинаковыми возбуждениями), но имеют различную пост-историю.

В событиях, следующих за такими состояниями, обязательно должны появиться непосредственно конфликтные состояния.

9. Формальные методы синтеза самосинхронных схем

Определение 9.1.3. Изменения ДИ называются *конфликтными*, если зоны возбуждения соответствующей ДП содержат конфликтные состояния.

ДИ (ДП) называется конфликтной (или противоречивой), если она содержит конфликтные изменения (состояния).

Утверждение 9.1.1. Правильная ДИ реализуема самосинхронной схемой в неограниченном базисе, если она бесконфликтна.

Итак,

1) конфликты свидетельствуют о неполноте исходного описания и могут быть классифицированы как ее некритичная некорректность, так как устраняются введением в описание новых переменных;
2) присутствие конфликтов в исходном описании говорит о том, что по нему невозможно построить самосинхронную схему.

Для того, чтобы бесконфликтное описание могло быть реализовано самосинхронной схемой в антитонном базисе, необходимо ввести дополнительные ограничения.

9. Формальные методы синтеза самосинхронных схем

Определение 9.1.4. Состояния v и w ДП назовем *анормальными*, если они представлены сравнимыми наборами и существуют переменные x_i, имеющие в этих состояниях различные устойчивые значения.

Например, состояния 1*00 и 1*01 не являются нормальными, так как они сравнимы и последняя переменная имеет различные значения.

Определение 9.1.5. Изменения в ДИ назовем *анормальными*, если соответствующие им зоны возбуждения в ДП содержат взаимно анормальные состояния.

ДИ (ДП) будем называть *нормальной*, если она не содержит анормальных изменений (состояний).

Утверждение 9.1.2. Правильная бесконфликтная ДИ (ДП) реализуема в антитонном базисе, если и только если она нормальная.

Очевидно, что нормальная ДИ является знакопеременной (сигналы каждого ее яруса имеют одинаковые знаки).

9. Формальные методы синтеза самосинхронных схем

Для того, чтобы привести ДИ к нормальному виду, нужно вводить дополнительные переменные таким образом, чтобы сделать ее знакопеременной. Таким образом анормальность не является критической некорректностью.

Следующая таблица дает классификацию типов некорректности.

Типы некорректности	Диаграммы изменений	Диаграммы переходов	Схемы
Критические	Некорректность по переключению	—	—
	Автопараллельность	Нарушение полумодулярности	Зависимость от задержек элементов
	Нарушение связности	Наличие фиктивных циклов	Функциональная независимость между группами элементов
Некритические	Конфликты изменений	конфликты состояний	—
	Ненормальность изменений	Ненормальность состояний	Нельзя использовать антитонный базис

9. Формальные методы синтеза самосинхронных схем

9.2. Реализация последовательных процессов

9.2.1. Граф конфликтов и его раскраска

Рассмотрим простейший пример диаграммы, содержащей конфликты.

ДИ: $\bullet \rightarrow +a1 \rightarrow -a1 \rightarrow +b \rightarrow +a2 \rightarrow -a2 \rightarrow -b$ (замкнуто)

ДП: $\bullet \rightarrow 0*0 \rightarrow 1*0 \rightarrow 00* \rightarrow 0*1 \rightarrow 1*1 \rightarrow 01*$ (замкнуто)

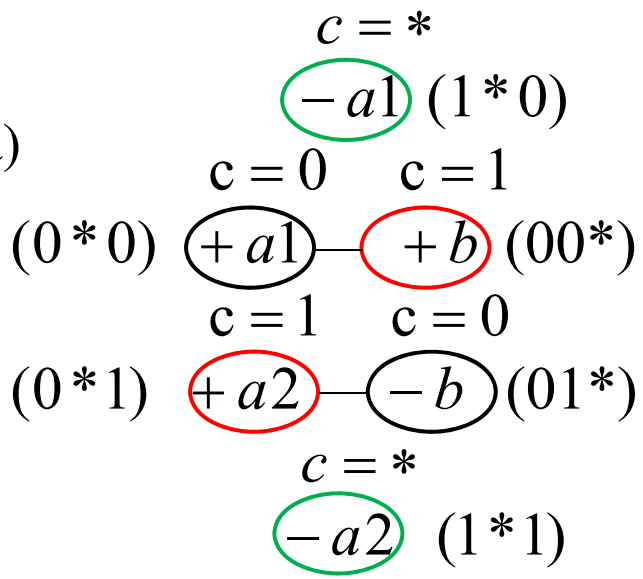
В ней присутствуют две пары конфликтных состояний (0*0, 00*) и (0*1, 01*).

Определение 9.2.1. Граф, в котором вершины являются вершинами ДИ (состояниями ДП), а дуги соединяют только пары конфликтующих вершин, назовем *графом конфликтов* (ГК).

Построим ГК для нашего примера и раскрасим его в два цвета в соответствии со значениями дополнительной переменной c (c =1 и c =0).

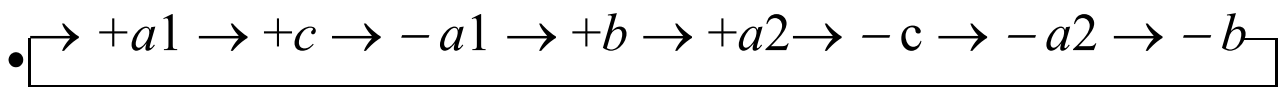
9. Формальные методы синтеза самосинхронных схем

ГК:
(первая раскраска)

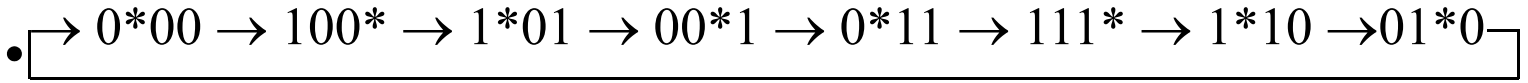


Согласно этой раскраске, между сигналами $+a1$ и $+b$ необходимо вставить сигнал $-c$, а между сигналами $+a2$ и $-b$ – сигнал $-c$. Тогда получим бесконфликтные диаграммы:

ДИ:



ДП:



9. Формальные методы синтеза самосинхронных схем

Можно предложить следующую процедуру построения бесконфликтной ДИ (ДП) по ДИ (ДП) содержащей конфликты:

- построение ГК на множестве вершин ДИ;
- раскраска ГК;
- кодирование цветов ГК дополнительными переменными;
- встраивание сигналов изменений дополнительных переменных в исходную ДИ в соответствии с кодированием.

Для данного примера эта процедура дала нужный результат. Однако она имеет существенные недостатки, связанные с многовариантностью мест расположения вершин дополнительных переменных и многовариантностью раскраски ГК.

Так, для рассмотренного примера вершину $+c$ нельзя вставить после вершины $-a1$, так как нетрудно видеть, что два смежных изменения одной и той же переменной всегда порождают конфликт (исчезает конфликт между $+a1$ и $+b$, но возникает новый между $+a1$ и $-c$). Кроме того, другая раскраска ГК (другая кодировка), устраняя прежние конфликты, порождает новые.

9. Формальные методы синтеза самосинхронных схем

Вновь возникающие конфликты после введения дополнительных переменных будем называть *вторичными*, в отличие от *первичных* (исходных).

Казалось бы, что появления вторичных конфликтов всегда можно избежать оптимизацией раскраски ГК, однако существуют примеры, это опровергающие.

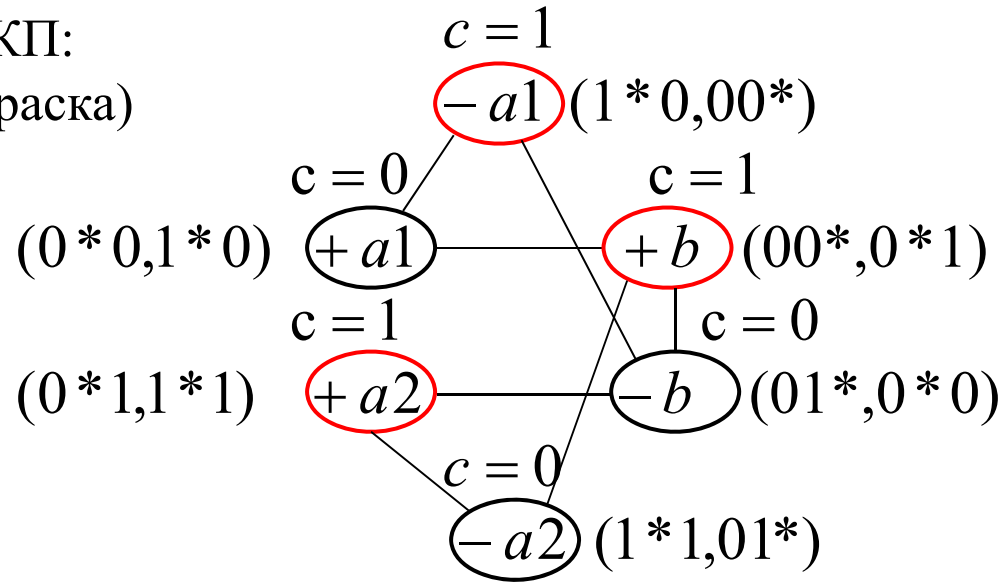
Обозначим через $post(x_i=a)$ множество состояний ДП, являющихся непосредственными последователями состояния, в котором переменная x_i переключилась после возбуждения в состояние a. Для последовательных процессов это множество состоит всего из одного состояния.

В ДИ последовательного процесса уберем все стрелки, оставив только вершины. Для каждой вершины в соответствующей ДП найдем ее состояние и состояние непосредственного последователя. Если пары состояний, соответствующие двум вершинам, содержат конфликтные состояния, то такие вершины соединим дугой. Построенный таким образом граф назовем *графом конфликтов с учетом последователей*, для краткости ГКП.

Построим ГКП для ДИ на слайде 18.

9. Формальные методы синтеза самосинхронных схем

ГКП:
(раскраска)



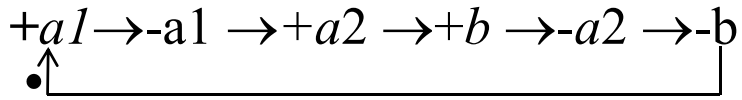
Хроматическое число для этого графа также равно двум. Если в начальном состоянии положить $c = 0$, то граф имеет единственную раскраску, которая гарантирует отсутствие вторичных конфликтов.

Рассмотрим другой пример, в котором для устранения конфликтных состояний требуется вводить как минимум две дополнительные переменные.

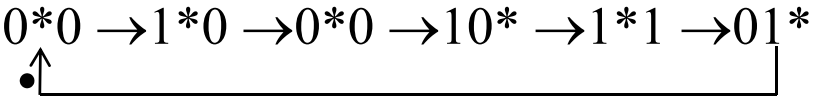
9. Формальные методы синтеза самосинхронных схем

Спецификация асимметричного Т-триггера

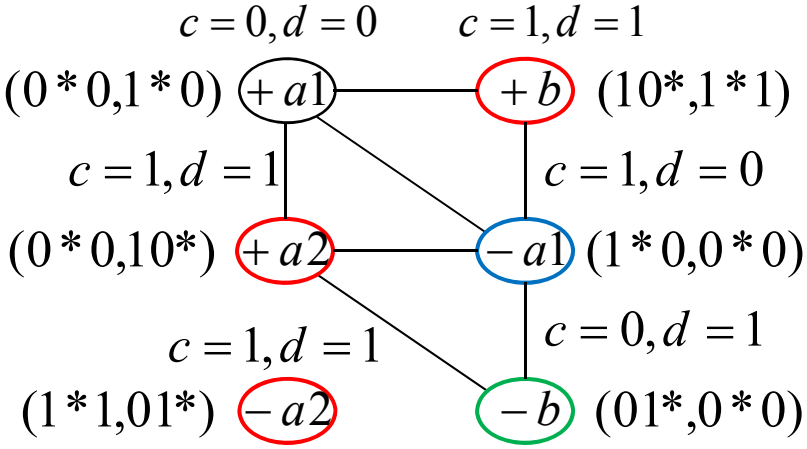
ДИ:



ДП:



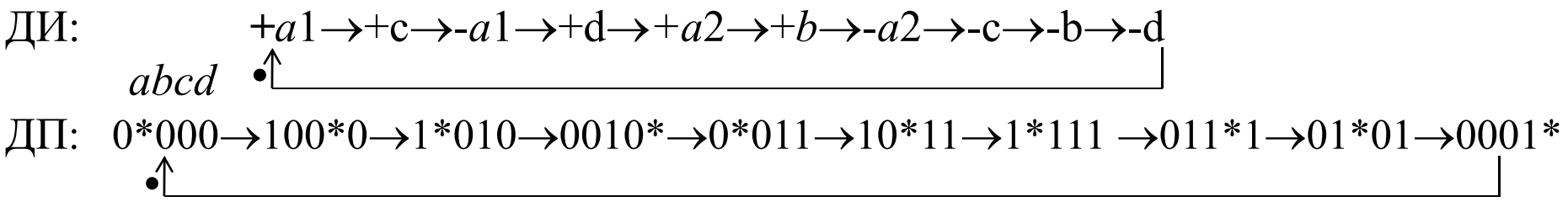
***Граф конфликтов с учетом последователей* (ГКП)**



Хроматическое число этого графа равно трем, следовательно необходимы две кодирующие переменные. Так как каждая переменная должна изменять свое значение четное число раз, то необходима раскраска в четыре цвета. При этом, любым двум конфликтным вершинам должны быть приписаны разные коды.

9. Формальные методы синтеза самосинхронных схем

Для варианта кодирования, указанного на графе, построим бесконфликтные диаграммы.



Легко видеть, что так как в ГКП нет избыточных дуг, решение проблемы его раскраски дает оптимальное разделение конфликтных вершин графа. Хроматическое число q графа ГКП определяет минимальное число кодирующих переменных $k =]\log_2 q[$.

9.2.2. Кодирование графа конфликтов

К сожалению, проблема кодирования раскраски графа отнюдь не проста и может вносить новые сложности. Например, в приведенном выше примере другое кодирование раскраски графа может привести ко вторичным конфликтам.

9. Формальные методы синтеза самосинхронных схем

Причина этого аналогична причине появления гонок при кодировании состояний асинхронного автомата.

Поставим в соответствие каждому цвету ГКП состояние асинхронного автомата, а дугам переходы. Для кодирования состояний можно использовать любой метод противогоночного кодирования. Число состояний автомата невелико и определяется хроматическим числом ГКП.

Противогоночное кодирование приводит к увеличению числа кодирующих переменных. Наиболее избыточным является кодирование соседними кодами. Такое кодирование по методу Хезелтайна для числа состояний q требует $(2\log_2 q\text{-}1)$ кодирующих переменных. Это верхняя оценка.

После того, как по исходной ДИ будет построена бесконфликтная ДИ, необходимо выполнить два заключительных шага:

- по бесконфликтной ДИ построить соответствующую ей ДП;
- на множестве состояний ДП построить таблицу истинности и по ней построить схему Маллера, проведя минимизацию соответствующих функций.

9. Формальные методы синтеза самосинхронных схем

Рассмотренный метод синтеза последовательных самосинхронных схем ориентирован на произвольный базис. Этот метод может быть обобщен на случай использования антитонного базиса.

Для того, чтобы привести бесконфликтную ДИ (ДП) к нормальному виду, необходимо обеспечить сравнимость состояний, соответствующих каждой паре смежных переходов.

Нетрудно показать, что в любой последовательной ДИ знакочередуемость изменений обеспечивает сравнимость каждой пары смежных состояний и несравнимость состояний через одно, что и требуется.

Поскольку для кодирования красок ГКП использовался автоматный подход, тип кодирования не меняется при инвертировании кодирующих переменных, так как при этом происходит только поворот куба кодирования. Поэтому для достижения знакочередуемости ДИ можно инвертировать изменения каждой кодирующей переменной.

9. Формальные методы синтеза самосинхронных схем

Если после этого остались места нарушений знакочередуемости, то в эти места ДИ необходимо вставить изменения дополнительных переменных.

Часто при составлении исходных ДИ нас интересуют не знаки изменений переменных, а сам факт их переключений. В этом случае можно избежать введения дополнительных переменных и обеспечивать полную знакочередуемость ДИ изменением знаков основных переменных.

Итак мы рассмотрели метод синтеза автономных последовательных схем. Для синтеза разомкнутых последовательных схем можно поступить следующим образом.

Построив описание функционирования схемы в виде разомкнутой ДИ, построить ее репозицию, например в виде протокола, которая описывает поведение среды. Замкнуть эти два описания и далее вести синтез только для внутренних переменных последовательного процесса, не обращая внимания на внешние переменные. При этом, естественно, в функции элементов построенной схемы будут входить выходные переменные среды.

9. Формальные методы синтеза самосинхронных схем

9.3. Параллельные процессы

9.3.1. Трудности, возникающие при реализации параллельных ДИ

1. В последовательных ДИ каждое изменение находится в одноднозначном соответствии с некоторым состоянием эквивалентной ДП. В произвольной ДИ одному изменению значения переменной соответствует в ДП зона ее возбуждения, содержащая много состояний. В результате устранение конфликтных состояний требует другой процедуры.

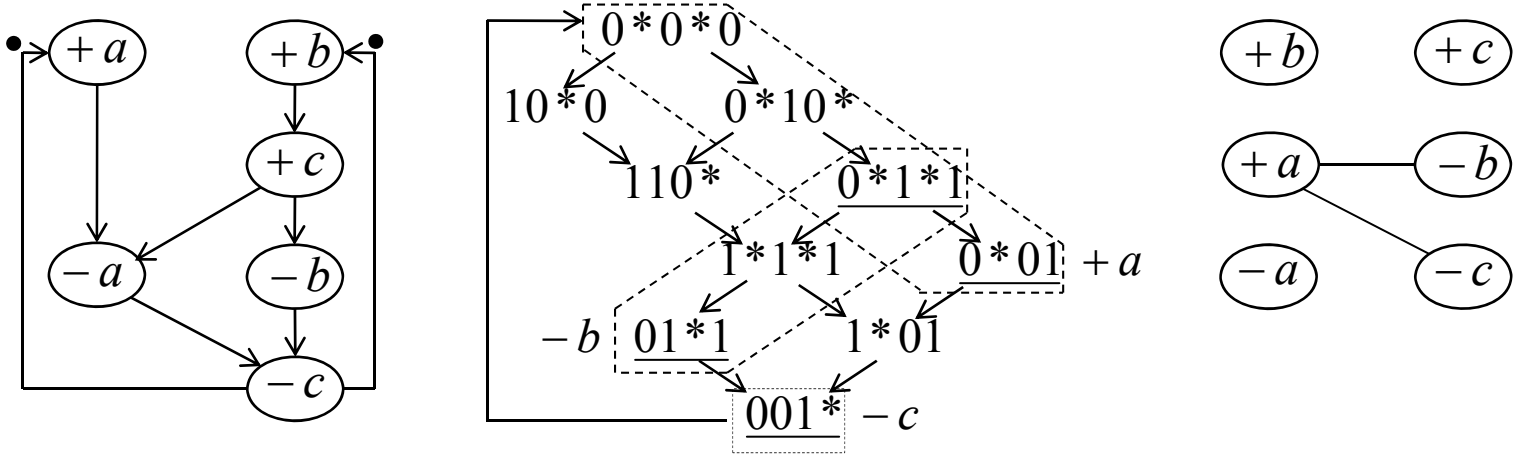
2. В произвольной ДИ нарушения реализуемости могут возникать не только между упорядоченными изменениями, но и между параллельными. Поэтому невозможно простым включением дополнительных переменных между конфликтующими изменениями устранить все конфликты, не нарушая исходного упорядочения изменений. Необходимы дополнительные исследования.

3. Будем рассматривать только дистрибутивные ДИ, не содержащие вершин типа ИЛИ, так как недистрибутивность требует преодоления дополнительных трудностей. Даже для дистрибутивных ДИ устранение конфликтов является сложной задачей.

9. Формальные методы синтеза самосинхронных схем

Возникающие проблемы проиллюстрируем на примере.

ДИ: ДП: ГК:



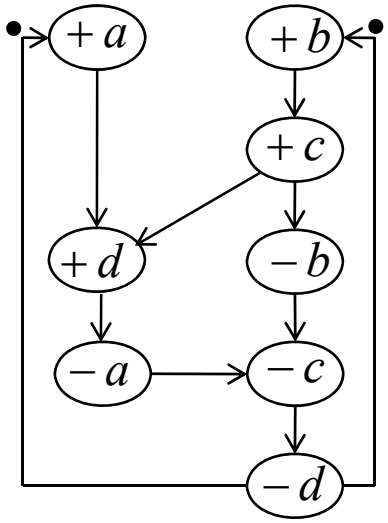
ДП имеет две пары конфликтных состояний (0*1*1, 01*1) и (0*01, 001*). Эти конфликты должны быть устранены.

Состояние 0*1*1 принадлежит зонам возбуждения двух сигналов $+a$ и $-b$, а состояние 01*1 – зоне возбуждения $-b$. Следовательно конфликт между изменениями $-b$ и $+a$. Аналогично, состояния 0*01 и 001* указывают на конфликт между изменениями $+a$ и $-c$. Этот факт отражен на графе конфликтов ГК.

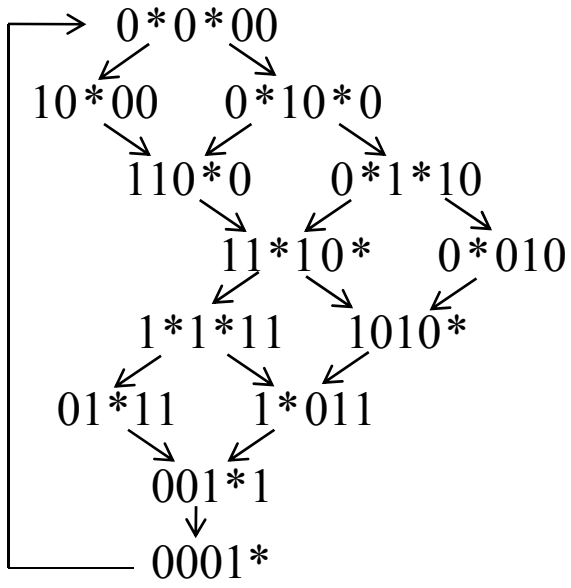
9. Формальные методы синтеза самосинхронных схем

Все конфликты можно исключить добавлением одной переменной d, как это показано на рисунке.

ДИ:



ДП:



Так как $+a \parallel -b$, то соответствующие им зоны возбуждения пересекаются и невозможно ввести промежуточную переменную так, чтобы в одной зоне она имела одно значение, а в другой другое.

Однако добавление изменений вспомогательной переменной между упорядоченными изменениями $+a \Rightarrow -c$ устраняет конфликт между ними.

Так как упорядоченные изменения $-b \Rightarrow -c$ не конфликтуют, то устранение конфликта между $+a$ и $-c$ привело к устранению конфликта между $+a$ и $-b$.

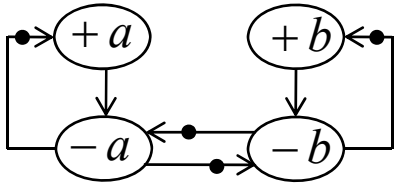
9. Формальные методы синтеза самосинхронных схем

9.3.2. ДИ, приводимые к бесконфликтной форме

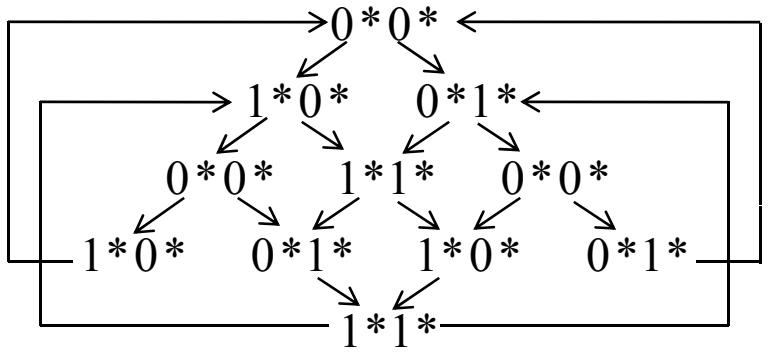
Из рассмотренного ранее примера можно сделать вывод, что для того, чтобы привести ДИ к бесконфликтной форме, достаточно удалить конфликты, возникающие между состояниями упорядоченных изменений, и что этим самым будут также удалены конфликты между неупорядоченными изменениями. Это действительно так, за редким исключением.

Рассмотри пример ДИ и соответствующей ей ДП.

ДИ:



ДП:



9. Формальные методы синтеза самосинхронных схем

Анализируя ДП можно придти к заключению, что в ней все пары конфликтных состояний не упорядочены (параллельны) и поэтому введением дополнительных переменных невозможно привести ее к бесконфликтной форме. Такая ДП является *неприводимой* к бесконфликтной форме.

Причина этого заключается в том, что ДИ, соответствующая этой ДП, не является безопасной, так как на дугах между сигналами $-a$ и $-b$ возможно накопление точек и, следовательно, необходима дополнительная память для их хранения. В дальнейшем будем иметь дело только с безопасными ДИ.

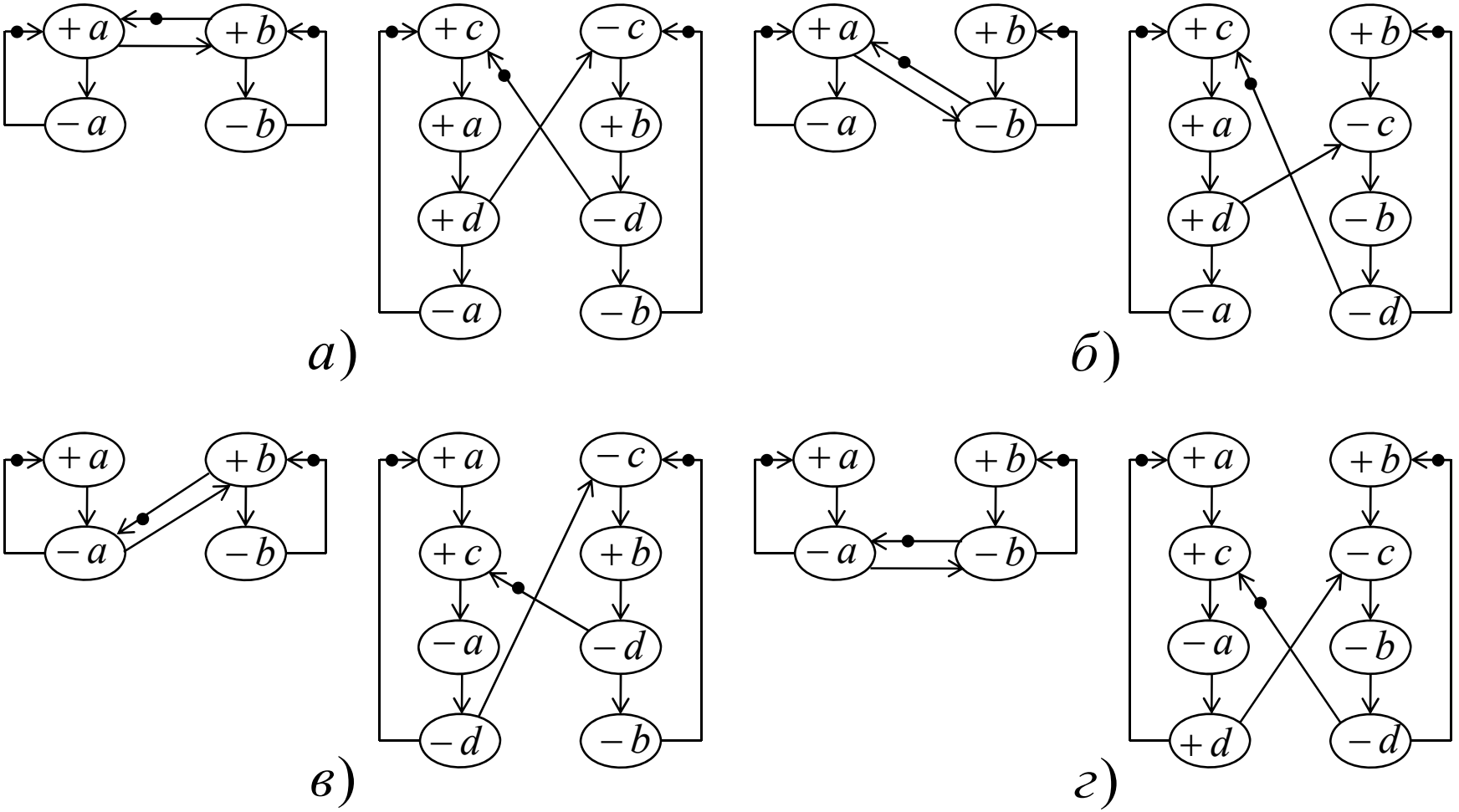
Доказано следующее утверждение.

Утверждение 9.3.1. Если в ДП, соответствующей некоторой ДИ, для двух или более упорядоченных конфликтных состояний v и w $(v \overset{s_1}{\Rightarrow} w \overset{s_2}{\Rightarrow} v)$ последовательности s_1 и s_2 изменений в ДИ содержат упорядоченные изменения c и d $(c \in s_1,\ d \in s_2)$, то ДП и соответствующая ей ДИ приводимы к бесконфликтной форме.

Поясним утверждение на примере ДИ, содержащей два активных цикла сигналов a и b.

9. Формальные методы синтеза самосинхронных схем

Четыре различных варианта введения дополнительных упорядоченных сигналов между циклами сигналов a и b



Логическое Проектирование Асинхронных Схем

Лекция 6

9. Формальные методы синтеза самосинхронных схем

9.3.3. Устранение конфликтов в приводимых ДИ

Для формализации процедуры приведения параллельной ДИ к бесконфликтной форме необходимо ввести следующие понятия.

Множество изменений $\{a_1,a_2,\ldots,a_p\}$ назовем *цепью*, если для любого i, $i = 1, 2,\ldots,p$-1 выполняется $a_i \Rightarrow a_{i+1}$ (отношение следования в отличие от отношения непосредственного следования $\rightarrow$).

Цепь l назовем *полной*, если не существует цепи l', множество вершин которой содержит подмножество вершин цепи l. (Если отсутствуют вершины типа ИЛИ, соседние изменения полной цепи должны всегда быть связаны отношением $\rightarrow$.) В дальнейшем будем рассматривать только полные цепи.

Если все конфликтные состояния в ДП различимы с помощью соответствующих изменений в ДИ, эквивалентной ДП, то может быть предложена следующая стратегия приведения ДИ к бесконфликтной форме.

9. Формальные методы синтеза самосинхронных схем

Алгоритм.

Шаг 1. Построить граф конфликтов (ГК) для ДИ, отражающий конфликты только среди упорядоченных изменений.

Шаг 2. Используя ГК, найти множество цепей L, удовлетворяющих условию полноты, т.е. каждая упорядоченная пара конфликтных изменений должна принадлежать одной и той же цепи из *L*.

Шаг 3. Решить проблему устранения конфликтов в каждой цепи с помощью ранее рассмотренного алгоритма устранения конфликтов в последовательных ДИ.

Может быть предложен другой подход к устранению конфликтов.

Итеративный алгоритм.

Этап 1. Выделим в ДИ, используя ее ГК, полную цепь l_1 и приведем ее к бесконфликтной форме. Однако некоторым вершинам этой цепи могут соответствовать состояния, конфликтующие с состояниями других цепей.

9. Формальные методы синтеза самосинхронных схем

С учетом введенных новых сигналов строим новую ДИ1, так как эти сигналы внесли изменения в порядок и, возможно, ранее неупорядоченные конфликты исчезли или стали упорядоченными.

Этап 2. Для полученной ДИ1 строим новый ГК1, с помощью которого выделяем полную цепь l_2, содержащую новые упорядоченные конфликты. После их устранения строим новую ДИ2 и так далее до тех пор, пока в очередной ДИ*k* все конфликты не будут устранены.

Доказана сходимость этого достаточно простого итеративного алгоритма.

С целью упрощения исходных описаний используется язык представления событий (ДИ). К сожалению, формальные методы синтеза опираются не только на ДИ, но и на ДП, поскольку конфликты между изменениями сигналов можно установить только в терминах состояний. По этой причине возникает проблема эффективности алгоритмов.

Сложность рассмотренных алгоритмов имеет квадратичную зависимость от размеров ДП.

9. Формальные методы синтеза самосинхронных схем

Для других универсальных методов поиска и устранения конфликтов, опирающихся на модели, использующие состояния, эта зависимость экспоненциальная.

Первый алгоритм использует обращение к ДП гораздо реже, чем итеративный. Поэтому он является более быстрым, но может дать избыточное число дополнительных переменных.

Итеративный алгоритм может дать лучшее решение, но за более длительное время.

Разработаны также алгоритмы синтеза недистрибутивных самосинхронных схем, базирующиеся на спецификациях ДИ с вершинами типа ИЛИ. Эти алгоритмы более сложные, так как такие схемы обладают более высокой степенью параллельности событий.

Разработанные формальные методы анализа и синтеза самосинхронных схем реализованы в рамках системы PETRIFY.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Одним из основных достоинств самосинхронных схем являются их *самодиагностические свойства*. Возрастание задержки любого элемента схемы сверх допустимого значения может интерпретироваться как дефект схемы, моделью которого являются *константные неисправности* (stack at fault).

Тип неисправностей определяется рядом факторов.

Во-первых, местом возникновения дефектов. Рассматриваемые здесь вопросы самодиагностики и саморемонта касаются лишь дефектов на выходах элементов или приводимых к выходам элементов.

Во-вторых, поведением дефектного элемента. Рассматриваются только константные неисправности, для которых собственная функция элемента $z_i = f_i(Z)$ заменяется на $z_i = 0$ или $z_i = 1$.

В третьих, кратностью неисправностей.

В четвертых, моментом возникновения дефекта. По этом признаку константные неисправности делятся на *консервативные* и *мутантные*.

10. Диагностические свойства самосинхронных схем и организация саморемонта

К *консервативным* относятся неисправности вида $z_i = \sigma,\ \sigma = \{0,1\}$, в момент возникновения которых элемент z_i уже имел на выходе значение σ.

К *мутантным* отнесем неисправности, связанные с непредвиденным переключением значения выхода элемента, которое может оказывать воздействие в момент своего возникновения и, следовательно, приводить к аномальному поведению схемы.

Схемы, в которых в процессе функционирования обнаруживаются неисправности некоторых классов, в технической диагностике называют *полностью самопроверяемыми*. Для таких схем обычно входные и выходные наборы разбиваются на классы. Реакция исправной и неисправной схемы на один и тот же входной класс должна приводить к различным классам выходных наборов.

Обычно самопроверяемые схемы снабжаются устройствами (тоже самопроверяемыми), распознающими принадлежность выходных наборов тому или иному классу. Неисправность схемы обнаруживается на выходах таких схем встроенного контроля.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Теория полностью самопроверяемых схем развита, по существу, лишь для класса синхронных схем. Однако есть некорректность в ее названии, так как она не учитывает возможные неисправности системы синхронизации.

Самосинхронные схемы не нуждаются в синхронизации и свободны от критических состязаний. Как будет показано далее, их тоже можно отнести к классу полностью самопроверяемых.

Самосинхронные схемы являются полностью самопроверяемыми относительно константных консервативных неисправностей. Именно на такие неисправности ориентированы методы самодиагностики и саморемонта.

10.1. Полностью самопроверяемые комбинационные схемы

Между понятиями *индицируемость* и *полная самопроверяемость* существует тесная взаимосвязь. Очевидно, что полная самопроверяемость возможна лишь при определенной дисциплине смены входных наборов, например, двухфазной дисциплине.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Будем считать, что неисправность p проявляется в том, что комбинационная схема реализует систему функций $F_p(X) \neq F(X)$.

Определение 10.1. Асинхронная комбинационная схема называется *полностью самопроверяемой* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого допустимого перехода a–b справедливо либо $F_p(b) = F(b)$, либо $F_p(b) \notin L$, где L – класс допустимых выходов;

б) для любого допустимого перехода b–a справедливо либо $F_p(a) = F(a)$, либо $F_p(a) \notin K$, где K – класс допустимых выходов;

в) существует хотя бы один допустимый переход a–b или b–a, для которого выполняется либо условие $F_p(b) \notin L$, либо условие $F_p(a) \notin K$ (самотестируемость).

Условия а) и б) – условия “защищенности от неисправностей”. Из определения видно, что если неисправность не проявилась в первой фазе (в момент возникновения), то она обязательно проявится во второй фазе.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Данное ранее определение индицируемости позволяет формально подойти к вопросам самопроверяемости комбинационных схем. Нетрудно доказать следующее утверждение.

Утверждение 10.1. *Индицируемые* (*самосинхронные*) *комбинационные схемы являются полностью самопроверяемыми для одиночных и кратных константных неисправностей их элементов*.

10.2. Полностью самопроверяемые автоматы

В общем случае автомат может быть задан системой уравнений $F(X,Z)$, где X – множество входных двоичных переменных, а Z – множество двоичных переменных, кодирующих внутренние состояния автомата.

Следует иметь в виду, что в случае неисправности $p \in P$ внутренний переход c–d состояний автомата не может завершиться, поэтому вместо символа d последним появится некоторый символ $\delta \in (c,d]$, а при переходе d–c – символ $\gamma \in (d,c]$,

10. Диагностические свойства самосинхронных схем и организация саморемонта

Определение 10.2. Асинхронный автомат Мура будем называть *полностью самопроверяемым* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого допустимого перехода a–b справедливо либо $F_p(b,\delta) = F(b,d)$, либо $F_p(b,\delta) \notin D$, где D – класс допустимых состояний;

б) для любого допустимого перехода b–a справедливо либо $F_p(a,\gamma) = F(a,c)$, либо $F_p(a,\gamma) \notin C$, где C – класс допустимых состояний;

в) существует хотябы один допустимый переход a–b или b–a, для которого выполняется либо условие $F_p(b,\delta) \notin D$, либо условие $F_p(a,\gamma) \notin C$ (самотестируемость).

Условия а) и б) – условия “защищенности от неисправностей”.

Автомат Мили будем считать композицией автомата Мура и комбинационной схемы. Заметим, что в автоматах, как и в комбинационных схемах, неисправность может выявляться не в момент возникновения, а в следующей фазе.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Утверждение 10.2. *Самосинхронные автоматы являются полностью самопроверяемыми для одиночных и кратных константных неисправностей их элементов.*

Механизм нахождения неисправностей связан со способами кодирования входных и выходных переменных саминхронизирующимися кодами. Исходя из расчетного быстродействия, можно установить некоторый интервал времени, в течение которого схема должна завершить переход. Отсутствие по истечении этого интервала выходного набора, принадлежащего классу D (для перехода a–b) или C (для обратного перехода b–a) интерпретируется как наличие неисправности в автомате.

Утверждения 10.1 и 10.2 устанавливают связь между проблемами технической диагностики и теории самосинхронных автоматов.

При традиционном подходе контролируемая схема проектируется без учета диагностических требований, а затем дополняется различными диагностическими схемами, распознающими принадлежность наборов тому или иному классу. При этом система синхронизации остается недиагностируемой.

10. Диагностические свойства самосинхронных схем и организация саморемонта

В самосинхронной схемотехнике контролируемость обеспечивается непосредственно при проектировании устройства, что значительно упрощает проектирование тестеров, обнаруживающих неисправности.

10.3. Диагностирование автономных схем

Рассмотрим вопрос о самопроверяемости автономных моделирующих схем управления, заданных с помощью модели Маллера.

Если в неавтономной схеме переходный процесс завершается некоторым тупиковым состоянием его диаграммы переходов, то исправная автономная схема не попадает в тупики.

Константная неисправность в автономной самосинхронной схеме сводится к фиксации значения переменной, соответствующей неисправному элементу. Естественным проявлением неисправности такой схемы представляется появление тупиковых состояний в ее диаграмме переходов.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Введем обозначения:

$R(\alpha)$ – множество рабочих состояний, содержащее состояние α (*рабочий цикл с состоянием* α);

R – *множество рабочих циклов схемы* (множество подмножеств состояний, образующих циклы);

T – *объединение множеств последовательностей состояний* неисправных схем , ведущих в тупики; такие последовательности находятся посредством преобразования диаграммы переходов исправной схемы по фиксированной неисправности.

Определение 10.3. Автономная схема с рабочими циклами из R называется *полностью самопроверяемой* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого рабочего цикла $R(\alpha) \in R$ выполняется либо $R_p(\alpha) = R(\alpha)$, либо $R_p(\alpha) \notin R$ и $R_p(\alpha) \in T$ (условие защищенности от неисправности);

б) существует хотя бы один рабочий цикл $R(\alpha) \in R$, для которого выполняется условие $R_p(\alpha) \notin R$ и $R_p(\alpha) \in T$ (самотестируемость).

10. Диагностические свойства самосинхронных схем и организация саморемонта

Выясним, относительно каких классов неисправностей полумодулярные схемы являются полностью самопроверяемыми.

Определение 10.4. Константную неисправность p будем называть: 1) *выносной* для множества R, если при ее возникновении схема оказывается в некотором состоянии ω таком, что $\omega \notin T$ и $\omega \notin R$; 2) *подменной* для множества R с рабочими циклами $R(\alpha)$ и $R(\beta)$, если найдется такое состояние $\gamma \in R_p(\alpha)$, что $\gamma \in R(\beta)$.

Выносная неисправность отличается от подменной тем, что выводит схему из той области, где она была задана. При подменной неисправности схема переходит в ранее неучтенный цикл диаграммы переходов. Схемы с выносными и подменными неисправностями не диагностируются. Как правило, в полумодулярных схемах управления такие неисправности являются экзотическими.

Утверждение 10.3. *Автономная схема с бесконечными рабочими циклами* $R(\alpha)$, $R(\beta)$, ... , $R(\gamma)$, *полумодулярная относительно* α, β, ... , γ, *является полностью самопроверяемой для невыносных и неподменных одиночных и кратных константных неисправностей элементов.*

10. Диагностические свойства самосинхронных схем и организация саморемонта

10.4. Саморемонт самосинхронных схем

Процесс саморемонта некоторого устройства связан с последовательной реализацией трех функций: *диагностирования* неисправности, ее *локализации* и *ремонта* методом замещения неисправного модуля резервным. Таким образом, саморемонт приводит к восстановлению работоспособной структуры устройства.

После саморемонта необходимо установить устройство в последнее контрольное состояние и реинициировать процесс.

Будем считать, что диагностирование неисправности осуществляется средствами самосинхронной схемотехники. Результатом диагностирования является выработка сигнала неисправности d =1.

Вторая функция – локализация неисправности – тоже реализуется достаточно просто. Если d = 1, то неисправность произошла в том модуле, в котором сигнал «ответ» b_i не совпадает с сигналом «запрос» a_i: $a_i \oplus b_i = 1$.

10. Диагностические свойства самосинхронных схем и организация саморемонта

Можно предложить два способа организации саморемонта устройств, имеющих регулярную модульную структуру.

1. Любой неисправный основной модуль структуры может быть замещен одним и тем же резервным модулем. Данный способ будем называть *скользящим резервированием с прямым замещением.*

2. Другой метод саморемонта основан на применении *скользящего резервирования с замещением посредством сдвига* по структуре.

При использовании любого из этих двух методов каждый i-й модуль состоит из функциональной части Φ_i, детектора D_i локализации неисправности модуля и коммутатора K_i на входах функциональной части Φ_i.

Детектор неисправности D_i модуля представляет собой триггер T_i фиксации неисправности, управляющий коммутацией, с функциями возбуждения R = «начальная установка» и $S = p(a_i \oplus b_i)$.

10. Диагностические свойства самосинхронных схем и организация саморемонта

При резервировании с прямым замещением каждый триггер неисправности T_i управляет коммутаторами тех модулей, на входы которых поступают выходы i-го модуля, и коммутатором резервного модуля.

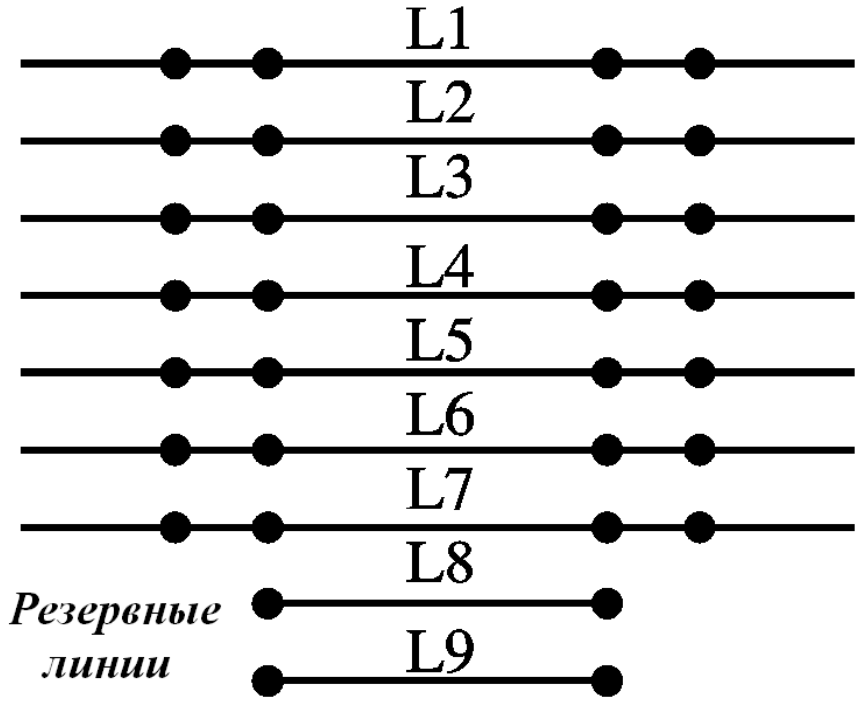
Таким образом, выходы всех основных модулей должны быть связаны с коммутатором резервного модуля и выходы резервного модуля должны быть связаны с коммутаторами всех основных модулей. Такая структура связей мало эффективна, так как сложность коммутатора резервного модуля становится очень высокой. Кроме того, значительно усложняется управление коммутацией в случае необходимости многократного резервирования.

Скользящее резервирование посредством сдвига более реалистично. В этом случае коммутаторы основных и резервных модулей одинаковы по сложности (для однородных структур), значительно упрощается управление коммутацией особенно в случае многократного резервирования.

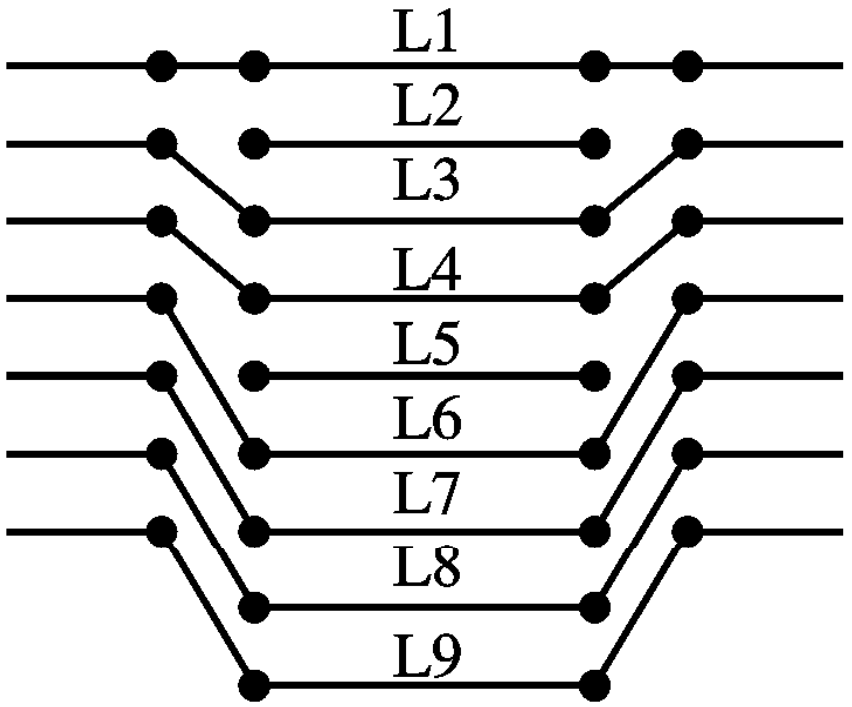
Ниже приведен пример замещения неисправных линий связи самосинхронного интерфейса при использовании резервирования линий посредством сдвига.

10. Диагностические свойства самосинхронных схем и организация саморемонта

а) Коммутация в случае исправных линий

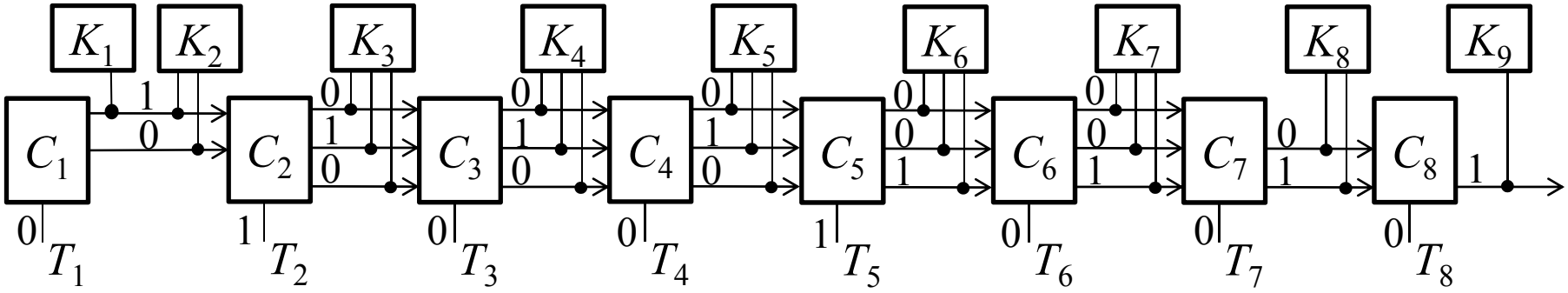


б) Коммутация в случае неисправных линий L2 и L5



Структурная схема организации управления коммутацией линий связи для рассматриваемого примера приведена на следующем рисунке.

10. Диагностические свойства самосинхронных схем и организация саморемонта



Выбор конкретного метода реконфигурации устройства при обнаружении неисправностей его блоков (метод саморемонта) зависит от структуры устройства и семантики протекающих в нем процессов. Организация саморемонта нерегулярных структур связана с большими трудностями.

11. Проблема построения электронных арбитров и синхронизаторов

При решении задач временного согласования обмена информацией между блоками системы, разделения общего ресурса, организации прерываний, а также любых других задач, связанных с использованием элементов взаимного исключения (неделимых операций), сталкиваются с проблемой аномального поведения элементарных элементов памяти, или *электронного арбитража*.

Устройства, обеспечивающие доступ от многих пользователей к единому ресурсу, называются *арбитрами* (или неограниченными арбитрами).

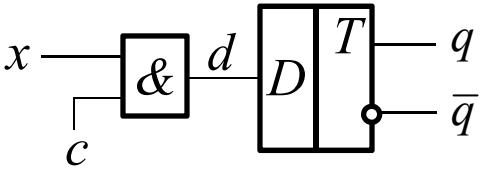
Устройства, осуществляющие временную привязку асинхронного сигнала к тактовой частоте синхронного устройства, называют *синхронизаторами* (или ограниченными арбитрами).

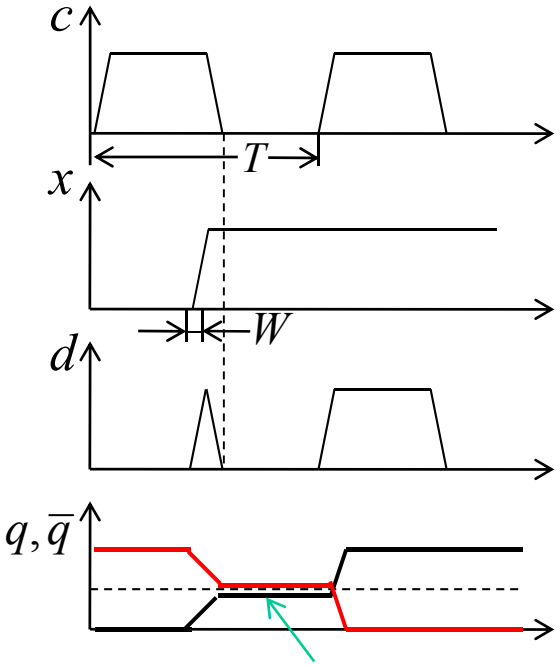
В 1981г. Леонардо Марино формально доказал теорему, утверждающую что в такого типа устройствах всегда существует вероятность возникновения аномального поведения.

Аномальное поведение синхронизаторов и арбитров является источником возникновения сбоев в вычислительных и управляющих системах.

11. Проблема построения электронных арбитров и синхронизаторов

Поведение D-триггера при приеме асинхронного сигнала по синхротакту



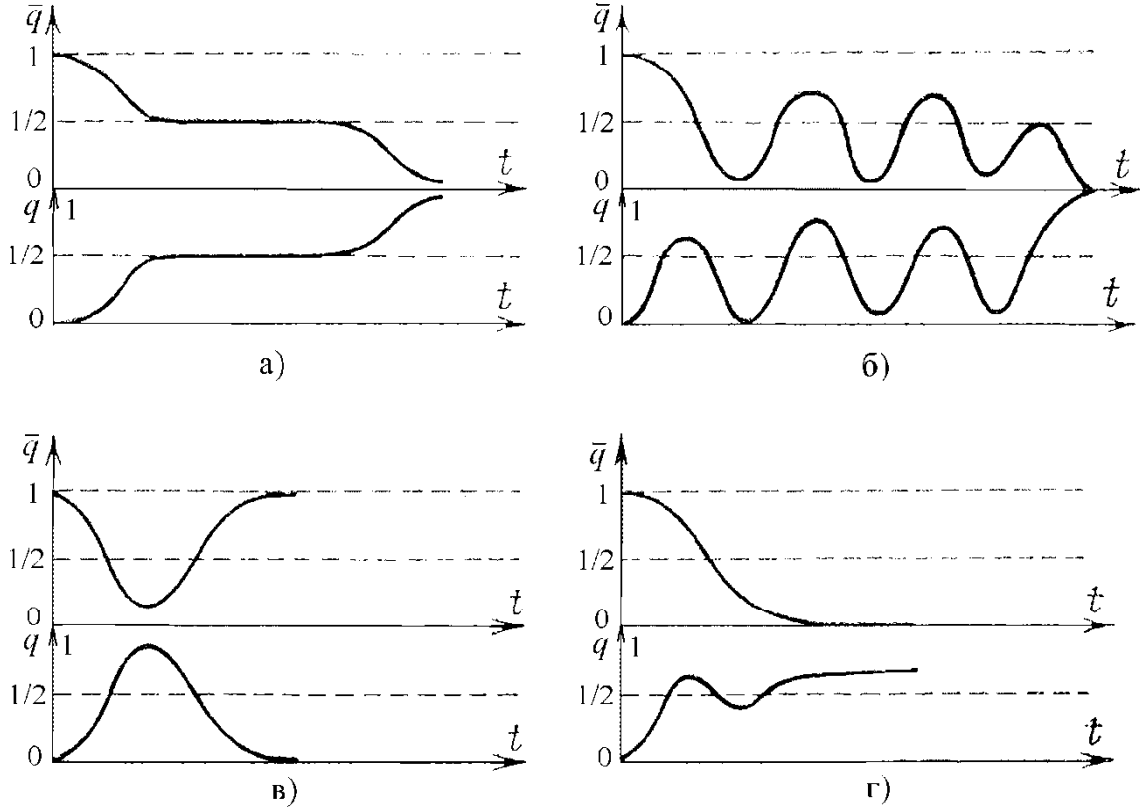


c – синхротакты с периодом T;
x – асинхронный сигнал;
d – сигнал на выходе элемента И;
$q, \bar{q}$ – выходные сигналы D-триггера;
W – *окно арбитража* – интервал времени, при попадании в который фронта асинхронного сигнала со 100% вероятностью поведение схемы становится аномальным.

Очевидно, что вероятность попадания схемы в аномальное состояние $p = W / T$.
Величина W измеряется долями наносекунды.

11. Проблема построения электронных арбитров и синхронизаторов

Возможные виды аномального поведения триггера



а) *метастабильная* аномалия; б) колебательная аномалия; в) нестабильный переходной процесс; г) затягивание фронта.

11. Проблема построения электронных арбитров и синхронизаторов

При появлении короткого импульса на входе *D*-триггера его энергии может оказаться недостаточно для полного переключения триггера, но достаточно для инициации переходного процесса, что приводит к аномальному поведению.

Возможность возникновения аномалий в) и г) определятся технологией изготовления устройства и с ними можно бороться простыми средствами (правильно проектировать). Аномалии типов а) и б) носят фундаментальный характер.

11.1. Метастабильность

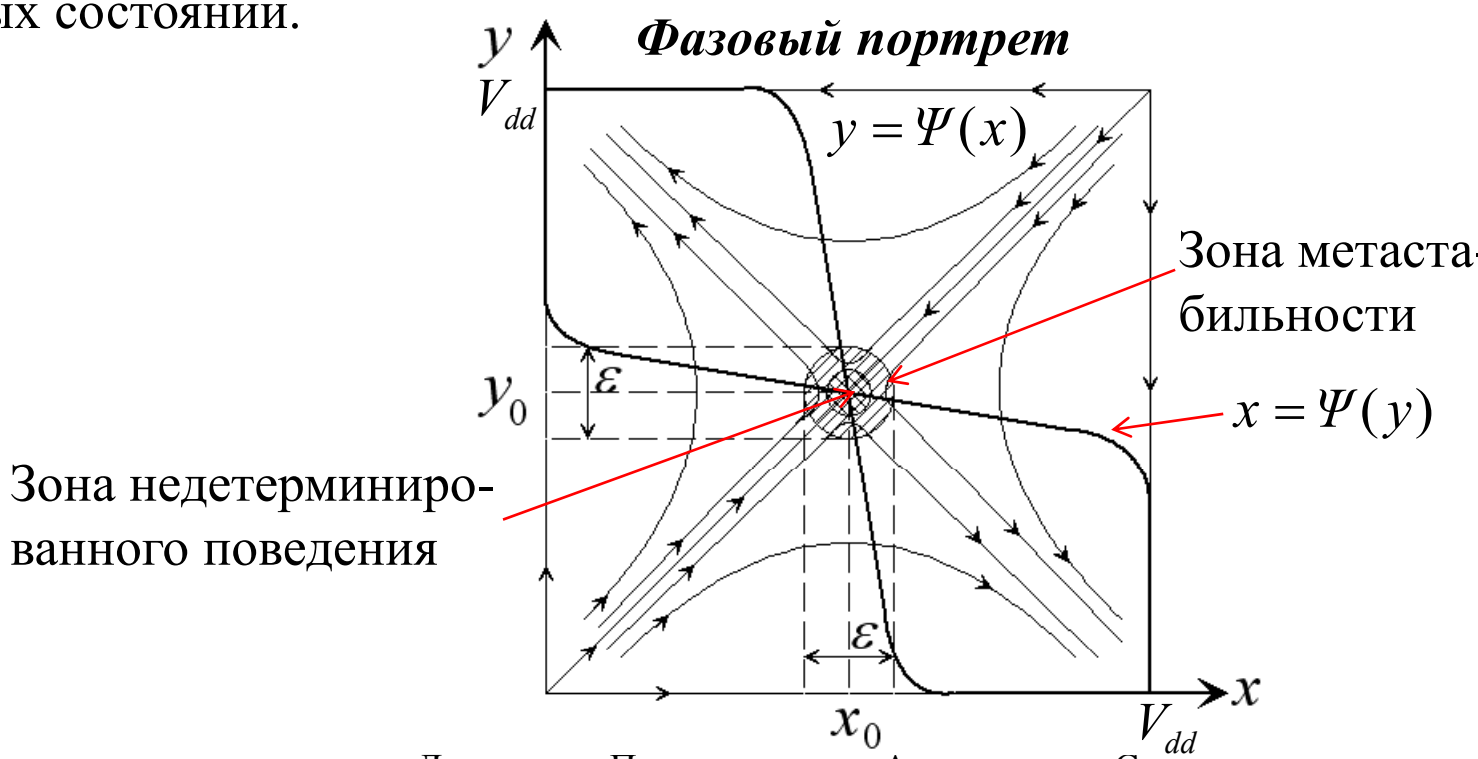
Рассмотрим поведение простейшего $\overline{R}\overline{S}$-триггера, построенного из двух элементов И-НЕ, на входы которого поступают два асинхронных сигнала. При отсутствии входных сигналов оба входа равны 0 и оба выхода равны 1. Если оба входа станут равными 1 строго одновременно, то оба плеча могут начать переключаться одновременно, что может привести к синфазным колебаниям.

Существует временной интервал W (начальное критическое окно) такой, что при рассогласовании фронтов асинхронных сигналов меньше W аномальное поведение неизбежно.

11. Проблема построения электронных арбитров и синхронизаторов

Поведение триггера можно описать системой уравнений: $\begin{cases} \dot{x} = F(y, x), \\ \dot{y} = F(x, y), \end{cases}$

где $x, y \in [0, V_{dd}]$ – напряжения на выходах элементов триггера, а F – некоторая нелинейная функция, определяющая временные и электрические параметры инвертора. Переходный процесс в такой системе удобно рассматривать на фазовой плоскости, в которой изображающая точка движется к одному из устойчивых состояний.



11. Проблема построения электронных арбитров и синхронизаторов

Формы фазовых траекторий описываются уравнением

$$\frac{dy}{dx} = \frac{F(x,y)}{F(y,x)}.$$

Совокупность фазовых траекторий образует фазовый портрет системы. Этот портрет имеет особую точку $]x_0,y_0[$ типа “седло”, которая расположена на пересечении статических передаточных характеристик инверторов. Стрелки на траекториях указывают направление роста времени. Время достижения устойчивых состояний (время переходного процесса) тем больше, чем ближе фазовая траектория подходит к точке $]x_0,y_0[$. Нормальное время переключения триггера определяется траекториями на границах фазового портрета.

Вблизи точки $]x_0,y_0[$ существует зона (зона метастабильности), при прохождении через которую время достижения устойчивого состояния может многократно возрасти. Внутри этой зоны существует зона недетерминированного поведения, в которой изображающая точка осуществляет хаотические блуждания (броуновское движение). Время выхода из нее определяется случайным процессом блуждания.

11. Проблема построения электронных арбитров и синхронизаторов

Анализ системы диф. уравнений первого порядка, описывающих поведения триггера показывает, что в ней не могут возникнуть колебания в области точки x_0,y_0, если аргументы функции F не содержат запаздывания. При объяснении факта существования колебательной аномалии необходимо либо ввести в уравнения запаздывание сигналов, либо рассматривать уравнения более высокого порядка. Остановимся лишь на метастабильной аномалии.

В одной из работ 1975г. было показано, что среднее время выхода из метастабильной зоны в отсутствие шумов зависит от постоянных времени элементов триггера. При этом относительная частота $\nu(t)$ переходных процессов длительностью больше t при вероятности попадания в метастабильную зону, равной 1, имеет вид:

$$\nu(t) = e^{-t/\tau},$$

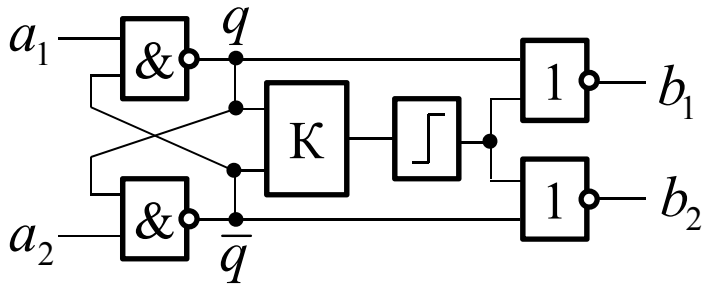
где τ - суммарная постоянная времени всех цепей обратных связей триггера.

Несмотря на то, что с уменьшением τ, эта относительная частота уменьшается, было замечено, что с увеличением быстродействия элементов триггера, увеличивается относительная средняя длительность переходного процесса.

11. Проблема построения электронных арбитров и синхронизаторов

11.2. Неограниченные арбитры

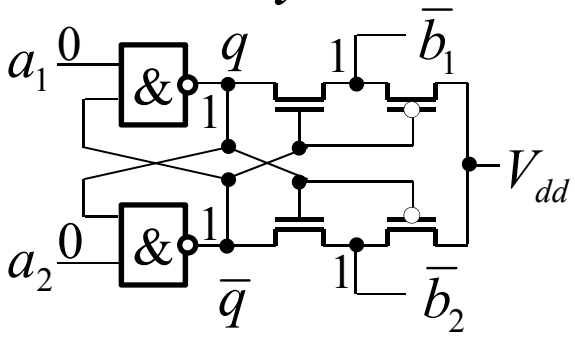
Коль скоро, согласно теореме Марино, не существует схем триггеров, в которых не возникала бы метастабильность, были предприняты попытки зафиксировать момент выхода схемы из метастабильного состояния. Идея такой фиксации иллюстрируется следующей схемой.



В этой схеме K – компаратор с пороговым элементом на выходе. Когда плечи триггера $(q, \overline{q})$ разойдутся на некоторую величину, можно считать, что триггер вышел из метастабильного состояния. Заметим, что эта же идея работает и в случае колебательной аномалии, когда напряжения на обоих плечах триггера изменяется синфазно. Реализация компаратора с порогом показана на следующем рисунке.

11. Проблема построения электронных арбитров и синхронизаторов

Схема мутекса



Назовем эту схему *мутексом* (от англ. MUTEX – MUTually EXclusive element).

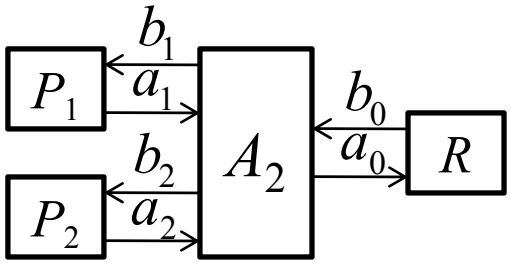
Выходные сигналы схемы сохраняют значение 11 до тех пор, пока плечи триггера не разойдутся на величину порога транзистора (т.е.выйдут из метастабильной зоны). После этого одно из плеч (в чью пользу разрешился арбитраж) начнет переключаться в состояние 0.

Таким образом, мутекс маскирует метастабильную и синфазную колебательную аномалии и не пропускает на выход неопределенных сигналов.

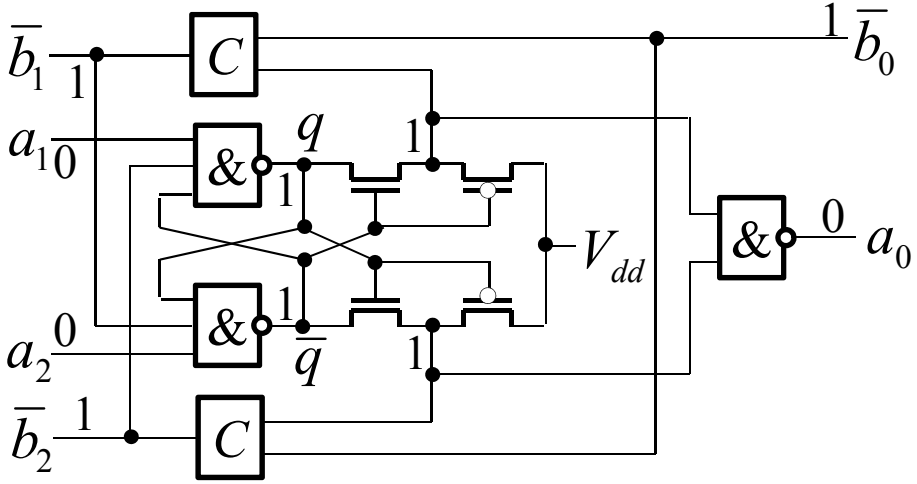
На основе мутакса могут быть построены асинхронные арбитры, разрешающие доступ к распределяемому ресурсу. Рассмотрим двухвходовый арбитр.

11. Проблема построения электронных арбитров и синхронизаторов

Двухвходовый арбитр



а) Обозначение



б) Реализация

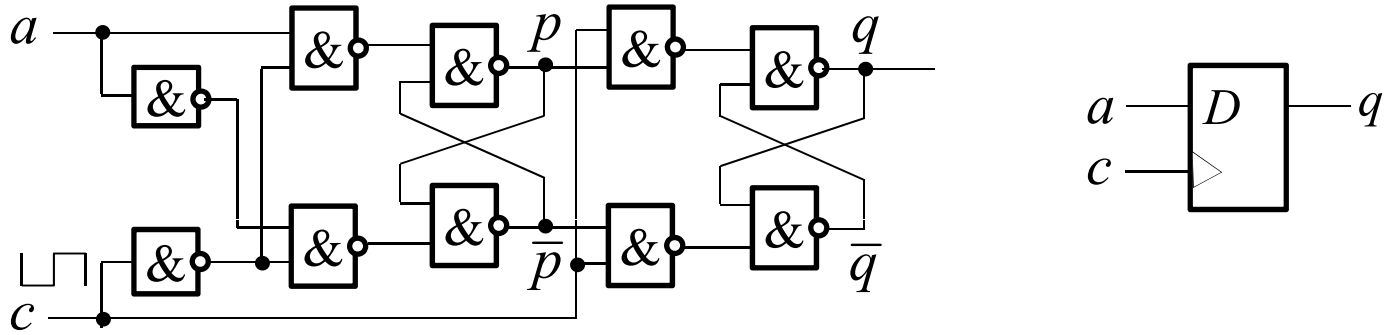
Многовходовый арбитр может быть построен из двухвходовых в виде пирамидальной схемы в случае одинаковых приоритетов запросов от процессов на некоторый ресурс. При наличии статических приоритетов может быть построена схема из последовательных и пирамидальных участков. Динамические приоритеты требуют реконфигурации структуры многовходового арбитра, что трудно реализуемо.

11. Проблема построения электронных арбитров и синхронизаторов

11.3. Синхронизаторы

В отличие от неограниченных арбитров синхронизаторы не могут быть “чистыми”, т.е. они всегда будут источниками ложных срабатываний.

Используем в качестве синхронизатора обычный D-триггер, схема которого имеет вид:



Во время паузы сигнала c он принимает значение асинхронного сигнала a в триггер $(p, \overline{p})$. Если фронт асинхронного сигнала попадет в окно арбитража W, то этот триггер установится в метастабильное состояние и в нем начнется процесс арбитража.

11. Проблема построения электронных арбитров и синхронизаторов

При с =1 метастабильное состояние переписывается в триггер $(q, \overline{q})$ и, если в течение этого полутакта в первом триггере не закончится арбитраж, по при при следующем с =0 с выхода q будет снят дефектный сигнал, который может привести к сбою в устройстве.

Таким образом, при использовании D-триггера в качестве синхронизатора на арбитраж отводится полтакта ($t = T/2$).

Экспериментальные исследования показывают, что вероятность возникновения в схеме триггера аномального состояния любого типа длительностью, превышающей временной интервал t, приблизительно описывается экспоненциальной зависимостью

$$P_t = P_0 \cdot e^{-t/\tau},$$

где τ - суммарная постоянная времени всех цепей обратной связи арбитрирующего триггера (находится экспериментально), P_0 – вероятность возникновения арбитража:

$$P_0 = \frac{W}{T_c} \cdot F_a = W \cdot F_c \cdot F_a.$$

11. Проблема построения электронных арбитров и синхронизаторов

Здесь W – ширина окна арбитража, T – период синхронизации, F_c – частота синхронизации, F_a – частота асинхронного сигнала.

Из вышеприведенных зависимостей нетрудно посчитать средний интервал времени $\bar{t}$ между сбоями синхронизатора:

$$\bar{t} = \frac{1}{P_t} = \frac{e^{t/\tau}}{W \cdot F_c \cdot F_a}.$$

Пример расчета среднего времени между сбоями для D-триггера.

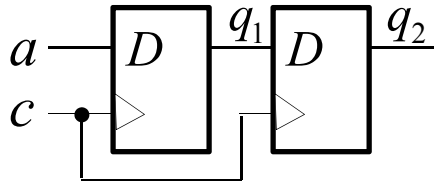
Пусть D-триггер выполнен в КМОП технологии 0.13мкм. Для этой технологии: задержка вентиля (инвертора) – $\tau_g \approx 33$ пс, $\tau \approx 2\tau_g = 66$ пс, $W \approx 4\tau_g = 132$ пс. Если F_c =500 Мгц (T = 2нс), F_a =50 Мгц, то

$$\bar{t} = \frac{e^{0.5T/\tau}}{W \cdot F_c \cdot F_a} = \frac{e^{1000/66}}{132 \cdot 10^{-12} \cdot 0.5 \cdot 10^9 \cdot 0.5 \cdot 10^8} \approx \frac{e^{15.15}}{3.3 \cdot 10^6} \approx 1.15\,\text{с}.$$

Задержка синхронизации равна одному такту.

11. Проблема построения электронных арбитров и синхронизаторов

Для увеличения среднего времени между сбоями обычно синхронизатор строят из двух последовательно включенных D-триггеров:



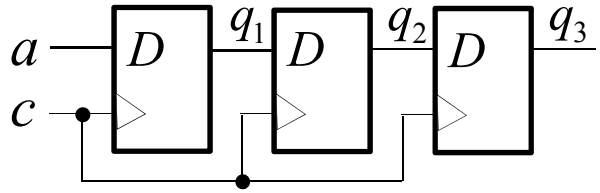
В такой схеме, если метастабильная аномалия не закончится в первом *D*-триггере за время, равное $0.5T$, то она передается в первый триггер второго D-триггера, а затем в его выходной триггер. Это равносильно увеличению времени арбитража на величину T. Тогда при расчете среднего времени между сбоями следует положить $t = 1.5T = 3$ *нс* и для нашего примера

$$\bar{t} = \frac{e^{1.5T/\tau}}{W \cdot F_c \cdot F_a} \approx \frac{e^{45.45}}{3.3 \cdot 10^5} \approx 1.66 \cdot 10^{11}\,\text{с} \approx 2250 \text{ лет.}$$

При этом задержка синхронизации составит два такта.

Для дальнейшего увеличения интервала между сбоями можно включить третий *D*-триггер и увеличить задержку синхронизации до трех тактов.

11. Проблема построения электронных арбитров и синхронизаторов



Рассмотренный расчетный пример не требует введения в синхронизатор третьего триггера. Обычно достаточно синхронизатора из двух *D*-триггеров.

Пример реального расчета. Пусть при условии использования той же технологии F_c = 200 Мгц, F_a = 20 Мгц, а в качестве синхронизатора используется один *D*-триггер, тогда t = *T*/2 = 2.5 нс и

$$\bar{t} = \frac{e^{t/\tau}}{W \cdot F_c \cdot F_a} = \frac{e^{2500/66}}{132 \cdot 10^{-12} \cdot 0.2 \cdot 10^9 \cdot 0.2 \cdot 10^7} \approx \frac{e^{37.8}}{5.28 \cdot 10^6} \approx 5.35 \cdot 10^9 \text{ с} \approx 170 \text{лет}.$$

Конечно эта цифра неправдоподобна, но удовлетворительна. Поэтому такого типа расчетам следует доверять в том случае, когда сбои оказываются опасными. Считается допустимым, когда частота сбоев от синхронизации не превышает одного сбоя за четыре года.

Логическое Проектирование Асинхронных Схем

Лекция 7

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Существующие подходы к проектированию асинхронных аппаратных средств, которые не используют общих часов в процессе координации событий, в том числе и подход самосинхронизации, приводят к решениям высокой сложности и требуют перепроектирования имеющихся аппаратных средств.

Существуют четыре подхода к организации поведения аппаратных средств во времени:
ПС – полностью синхронные (синхронизация от общих часов);
ПА – полностью асинхронные (например, самосинхронные);
ЛАГС – локально асинхронные, глобально синхронные;
ГАЛС – глобально асинхронные, локально синхронные.

ПС и ЛАГС требуют системы доставки синхронизирующих сигналов от общих часов. Недостатки таких систем уже обсуждались. ПА системы обладают рядом важных преимуществ, но они очень сложны и разрывают непрерывность эволюционного развития аппаратуры. Наиболее привлекательными являются ГАЛС системы, так как они *объединяют преимущества как синхронного, так и асинхронного подходов*.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Будем рассматривать **ГАЛП** системы (глобально асинхронные, локально произвольные), которые являются расширением ГАЛС систем. В таких системах сосуществуют одновременно два типа времени: физическое время, в котором функционируют физические блоки системы, и логическое время, представляющее причинно-следственную семантику взаимодействия блоков системы.

В физическом времени асинхронность трактуется как непредсказуемые вариации длительностей физических переходных процессов. В логическом времени асинхронность трактуется как вариации количества дискретных шагов в процессах.

Будем говорить о *глобальном* логическом времени, если функционирование структурных блоков системы координируется в логическом времени. *Локальное* логическое время протекает внутри структурных блоков.

Любая асинхронная система может быть декомпозирована на синхронизирующую подсистему (***синхро-стратум***) и синхронизируемую подсистему (например, ***процессорный стратум***), состоящий из синхронизируемых блоков системы.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Такой подход позволяет при построении асинхронной системы использовать синхронные прототипы блоков, что обеспечивает непрерывность в эволюционном развитии аппаратных средств.

Синхро-стратум функционирует как распределенные асинхронные часы, с помощью которых осуществляется глобальная синхронизация блоков системы, и может быть построен в соответствии с прототипом системы синхронизации от общих часов.

Асинхронное взаимодействие синхростратума с блоками системы может быть организовано по принципу «запрос-ответ» (хендшейка), которое требует от системных блоков формирования сигналов о завершении в них переходных процессов.

Такая задача может быть решена многими способами, например, сигнал завершения переходного процесса формируется с помощью задержки, включенной параллельно блоку, использованием самосинхронных блоков, уже имеющих такой сигнал, или применением старт-стопного генератора.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Возможно также использование синхростратумов второго и более высокого уровней иерархии.

Взаимодействие блоков системы с синхростратумом изучается на уровне сигналов инициации и завершения переходных процессов. Основное внимание уделяется способам трансформации системы синхронизации прототипа в спецификацию синхростратума, с помощью которого осуществляется глобальная синхронизация системных блоков.

12.1. Логическое и физическое время

Когда говорят о синхронных или асинхронных устройствах, в действительности имеют в виду способ *синхронизации* поведения устройства. Проблема заключается в организации *временного* (*темпорального*) поведения устройства, или инкорпорировании времени в модель устройства. Очевидно, что необходимо ясно определить, что понимается под словом *время*.

Современная наука говорит, что время имеет двойственную природу (как и материя).

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

С одной стороны, начиная с античных греков, время трактуется как число (Платон и Аристотель) или как момент (Corpus Hermeticum). С другой, оно трактуется как отражение причинно-следственного отношения меду событиями, определяющего их упорядочение.

Известно высказывание Аристотеля: «Если ничего не происходит, то нет и времени». Традиция греков была продолжена Лейбницем, который трактовал время как отношение частичного порядка на событиях, устанавливающее причинно-следственное связи между ними.

Энциклопедия McGraw-Hill Encyclopedia of Science & Technology определяет синхронизацию как «процесс поддержания одной операции в шагах другой». Здесь время рассматривается как дискретная величина; далее будем называть ее *логическим,* или *дискретным* временем.

Концепция аналогового времени связана с именем Исаака Ньютона, который продолжил и развил античную индоевропейскую концептуализацию времени как непрерывной величины.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Время трактуется как непрерывная независимая физическая переменная, которая называется *физическим* временем. Заметим, что все существующие определения физического времени связаны с процедурой его измерения, которая может быть выполнена только с определенной степенью точности.

Процедура измерения обеспечивает связь между физическим и логическим временем. Основным измерительным прибором являются часы; в процессе измерения они сравнивают временной интервал между двумя дискретными событиями с числом также дискретных событий, с помощью которых осуществляется измерение.

Корреляция двух событий тесно связана с концепцией *одновременности*, которая может трактоваться только как логическая абстракция, достигаемая также с определенной степенью точности. Неопределенность концепции одновременности является источником электронного арбитража.

Все шаги проектирования системы в той или иной степени касаются организации временного поведения ее блоков и поэтому проектирование ее системы синхронизации можно интерпретировать как создание системного времени.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Специалисты, проектирующие систему, по-разному трактуют системное время, также как и связанные с ним термины *синхронный* и *асинхронный*.

Специалисты в области программирования и архитектуры имеют дело с логическим временем. Они трактуют время как число шагов в процессе с дискретными состояниями. Асинхронизм трактуется как вариации (например, в зависимости от данных) числа шагов в процессе (алгоритме) до получения результата.

Специалисты в области микроэлектроники и вычислительной техники имеют дело с физическими процессами и, следовательно, с аналоговым физическим временем. Для них асинхронность связана с неуправляемыми вариациями времени перехода из одного дискретного состояния в другое. Основные факторы таких вариаций могут быть постоянными (дисперсия технологических параметров), переменными (изменение температуры, питающих напряжений и др.), а также зависящими от данных (время завершения переносов в сумматоре).

Проблемы синхронизации сосредоточены вокруг интерфейса между физическим и логическим временем. Простейшим способом обеспечения такого интерфейса в синхронных системах является использование системных часов.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Системные часы дают возможность полностью исключить из рассмотрения физическое время и модель устройства функционирует только в логическом времени. Это является основным преимуществом синхронного подхода.

В асинхронном подходе к проектированию системное время определяется частичным порядком, накладываемым на дискретные события. Время вводится с помощью спецификации причинно-следственных отношений между событиями; интерфейс между физическим и логическим временем обеспечивается механизмами фиксации моментов окончания переходных процессов в системных компонентах.

Основные методологические различиях между этими двумя подходами.

В синхронном подходе механизм, обеспечивающий системное время, полностью отделен от модели поведения системы.

В асинхронном подходе механизм, обеспечивающий системное время, встроен в модель поведения системы и его следует разрабатывать вместе с исходной поведенческой спецификации.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Асинхронное проектирование считают значительно более сложным, чем синхронное. Основная причина заключается в том, что поведенческие алгоритмы обычно формулируются как синхронные. Перевод синхронных алгоритмов в асинхронные – сложная задача, требующая большой изобретательности.

В очень редких случаев такая задача не может быть решена, если она принципиально связана с синхронностью. Например, трудно представить асинхронную реализацию бытовых часов.

Как правило, асинхронные реализации превосходят по сложности синхронные в 2.5 – 4 раза и, обладая весьма положительными свойствами, являются по этой причине мало эффективными. Поэтому в последние 15 лет бурно развивается ГАЛС подход, сочетающий в себе преимущества синхронного и асинхронного подходов и свободный от многих их недостатков.

Декомпозиция системы на синхростратум и процессорный стратум, вообще говоря, не обеспечивает достижения новых возможностей. Однако она позволяет абстрагироваться от функционального назначения блоков системы и сосредоточить основное внимание на проблеме их глобальной синхронизации.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Кроме того, выделение синхростратума позволяет строить глобально асинхронные системы по их синхронным прототипам. Далее будут рассмотрены методы построения спецификаций синхростратумов, соответствующих синхронным прототипам систем.

12.2. Спецификации и реализации синхростратумов

12.2.1. Стратегии синхронизации

Синхронизации посредством сигналов от часов, так или иначе, связана с организацией взаимодействия схемных блоков по принципу «ведущий – ведомый». В качестве модели рассмотрим клеточные массивы из автоматов Мура, построенных по двухрегистровой схеме в соответствии с принципом «основной – вспомогательный» (см. рис.).

При одном значении синхросигнала T автомат изменяет свое состояние посредством записи нового состояния в основной регистр. При другом значении T новое текущее состояние автомата копируется во вспомогательный регистр.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Такая двухфазная схема называется схемой с разнополярным управлением.

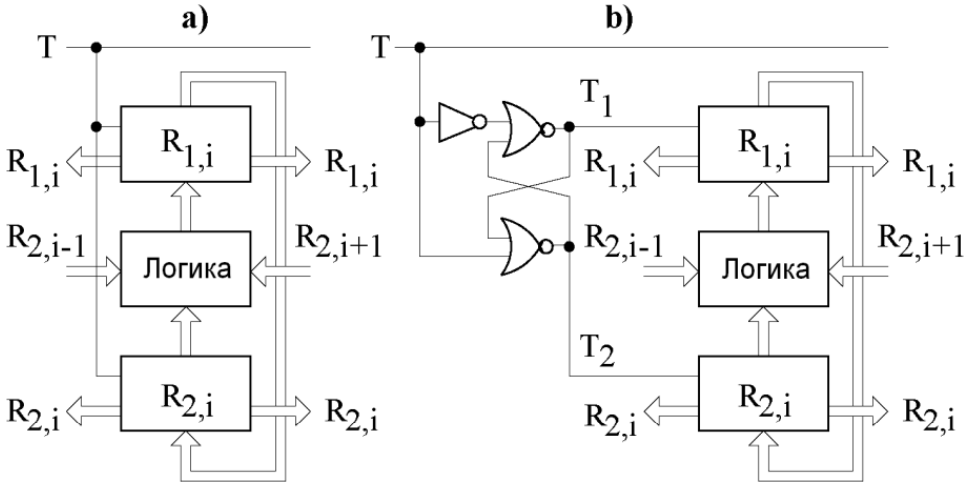


Рис.12.1. Двухрегистровая структура автомата с двухфазной синхронизацией: a) с разнополярным управлением регистрами; b) с однополярным управлением.

Массивы автоматов могут синхронизироваться с помощью различных систем синхронизации. В простейшем случае используется однопроводная двухфазная синхронизация с разнополярным управлением регистрами (рис. a). Состояние автомата определяется функцией переходов

$$S_i(t) = F_i[S_{i-1}(t-1), S_i(t-1), S_{i+1}(t-1)].$$

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Синхро-поведение автомата в этом случае имеет вид:

$$\text{если } T = 1, \text{то } R_{1,i} := F_i(R_{2,\text{i-1}}, R_{2,\text{i}}, R_{2,\text{i+1}}),$$

$$\text{если } T = 0, \text{то } R_{2,i} := R_{1,i}.$$

Схема автомата с двухфазным разнополярным управлением регистрами работает правильно только в случае, когда разбросом задержек в проводах синхросигнала T можно пренебречь.

Для того, чтобы исключить влияние задержек в проводах, следует использовать двухфазную двухпроводную синхронизацию с однополярным управлением регистрами. Регистры синхронизируются различными сигналами одной полярности и дисциплина следования синхросигналов имеет вид: $\ldots \to +T_1 \to -T_1 \to +T_2 \to -T_2 \to +T_1 \to \ldots$. В этом случае синхро-поведение автомата определяется как:

$$\text{если } T_1 = 1, \text{то } R_{1,i} := F_i(R_{2,\text{i-1}}, R_{2,\text{i}}, R_{2,\text{i+1}}),$$

$$\text{если } T_2 = 1, \text{то } R_{2,i} := R_{1,i}.$$

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Синхро-последовательность T может быть преобразована в пару синхро-последовательностей T_1 и T_2 с помощью специальной схемы, реализующей сигнальный граф ... $\rightarrow +T \rightarrow -T_1 \rightarrow +T_2 \rightarrow -T \rightarrow -T_2 \rightarrow +T_1 \rightarrow$ ..., например, как это показано на рис.12.1,b.

В дальнейшем будем полагать, без потери общности, что в цепи из клеточных автоматов, каждый автомат синхронизируется с помощью только одного синхросигнала. В то же время, различные автоматы цепи могут синхронизироваться разными синхросигналами.

В зависимости от структуры взаимосоединений между автоматами и принятого протокола взаимодействия цепочки автоматов могут синхронизироваться системой синхронизации из нескольких синхро-последовательностей. Будем называть синхро-последовательности *треками*.

Временная диаграмма наиболее часто применяемой двухтрековой двухфазной системы синхронизации приведена на рис.12.2,a. Существуют также трехтрековые (рис.12.2,b) и четырехтрековые (рис.12.2,c) схемы синхронизации.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

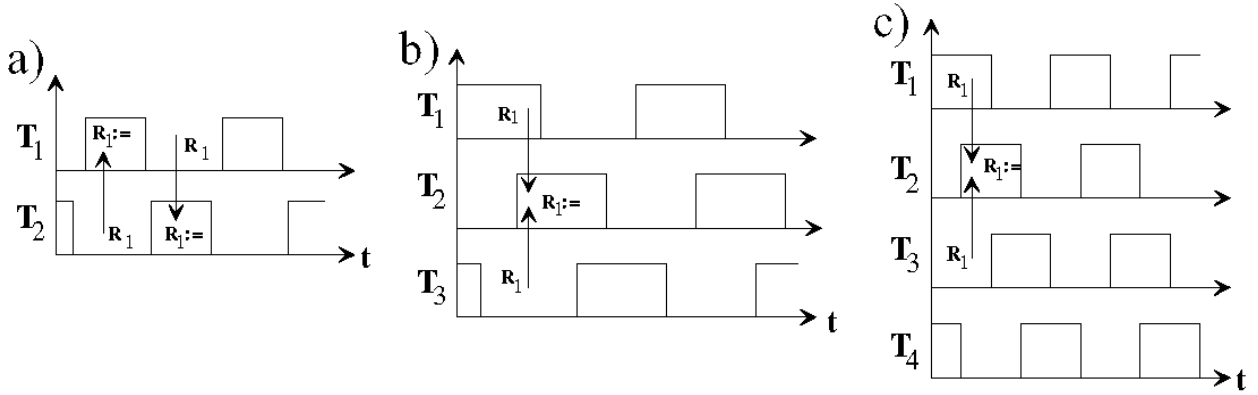


Рис.12.2. Временные диаграммы многотрековых синхросигналов: a) двухтрековая, b) трехтрековая и c) четырехтрековая.

Пример двухтрековой двухфазной синхронизации для одномерного массива автоматов приведен на рис.12.3,а. Сигналы T_1 синхронизируют нечетные автоматы, а сигналы T_2 – четные. При принятой двухфазной дисциплине функционирования автоматов они имеют следующее синхроповедение

$$\text{если } T = 1, \text{то } R_{1,i} := F_i(R_{1,\text{i-1}}, R_{2,\text{i}}, R_{1,\text{i+1}}),$$
$$\text{если } T = 0, \text{то } R_{2,i} := R_{1,i},$$

и выходным сигналом каждого автомата является состояние его регистра R_1.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

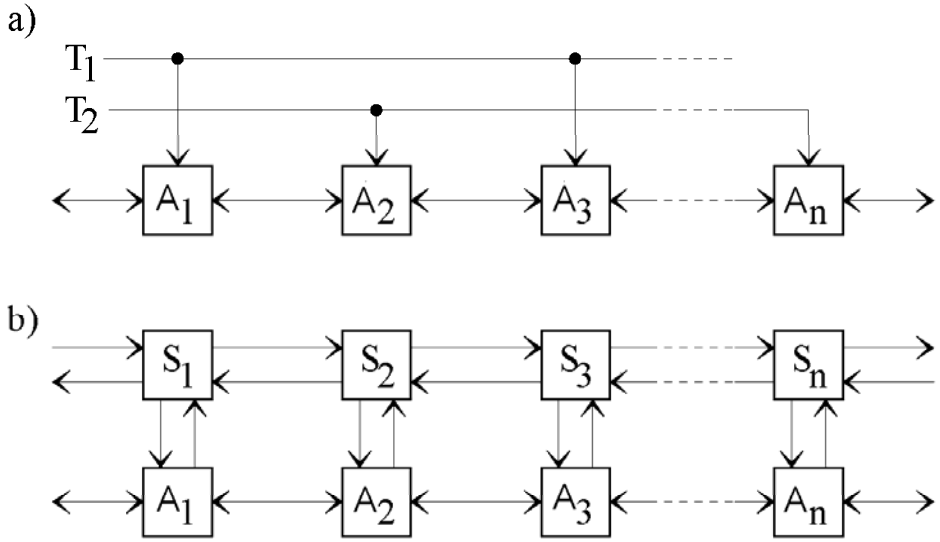


Рис.12.3. Одномерный массив автоматов: a) синхронная двухпроводная двухфазная реализация, b) асинхронная реализация с выделенным синхростратумом.

В таком массиве информационные потоки могут передаваться в обоих направлениях и процессы могут быть инициированы через свободные порты обоих оконечных автоматов. Другими словами, может быть реализован любой волновой алгоритм.

Пример асинхронной реализации одномерного массива с выделенным синхростратумом приведен на рис.12.3,b.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Структура взаимоседининений элементов синхростратума изоморфна структуре взаимосоединений автоматов в массиве. Элементы синхростратума взаимодействуют с автоматами по принципу «запрос – ответ».

Выходной сигнал «запрос» элемента синхростратума является сигналом локальной синхронизации автомата. Каждый автомат получает этот сигнал лишь при условии завершения переходных процессов в его соседях, а не после завершения переходных процессов во всех автоматах, как в прототипе.

12.2.2. Двухтрековая синхронизация одномерных массивов

Пусть одномерный массив на рис.12.3,а, состоит из автоматов Мили и выполняет алгоритм, представленный последовательностью шагов, т.е. в логическом времени. В каждый момент t этого времени все автоматы должны выполнять t-й шаг алгоритма. При этом внутренне состояние $S_i(t)$ каждого i-го автомата и состояния его правого $X_i(t)$ и левого $Y_i(t)$ выходов в момент времени t определяются как:

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

$$
\begin{aligned}
S_i(t) &= F_i[X_{i-1}(t), S_i(t-1), Y_{i+1}(t)],\\
X_i(t) &= f_{i,1}[X_{i-1}(t), S_i(t-1), Y_{i+1}(t)],\\
Y_i(t) &= f_{i,2}[X_{i-1}(t), S_i(t-1), Y_{i+1}(t)].
\end{aligned}
\qquad (1)
$$

Нас не интересуют ни структура, ни содержание этих уравнений. Интерес представляют только временные интервалы поведения, поэтому достаточно рассмотреть проблему синхронизации массива на модельном уровне.

В соответствии со структура рис.12.3,а автоматы взаимодействуют по принципу «ведущий – ведомый», используя две последовательности синхросигналов T_1 и T_2. Пусть эти сигналы изменяют свои значения следующим образом:

$$\ldots\rightarrow\{0,0\}\rightarrow\{1,0\}\rightarrow\{0,0\}\rightarrow\{0,1\}\rightarrow\{0,0\}\rightarrow\ldots$$

При T_1=0 (T_2=0) нечетные (четные) автоматы передают информацию своим соседям; при T_1=1 (T_2=1) нечетные (четные) автоматы получают информацию от своих соседей.

В такой структуре синхронизации каждый шаг k в логическом времени состоит из двух последовательных шагов в физическом времени. Рассмотрим следующий граф.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

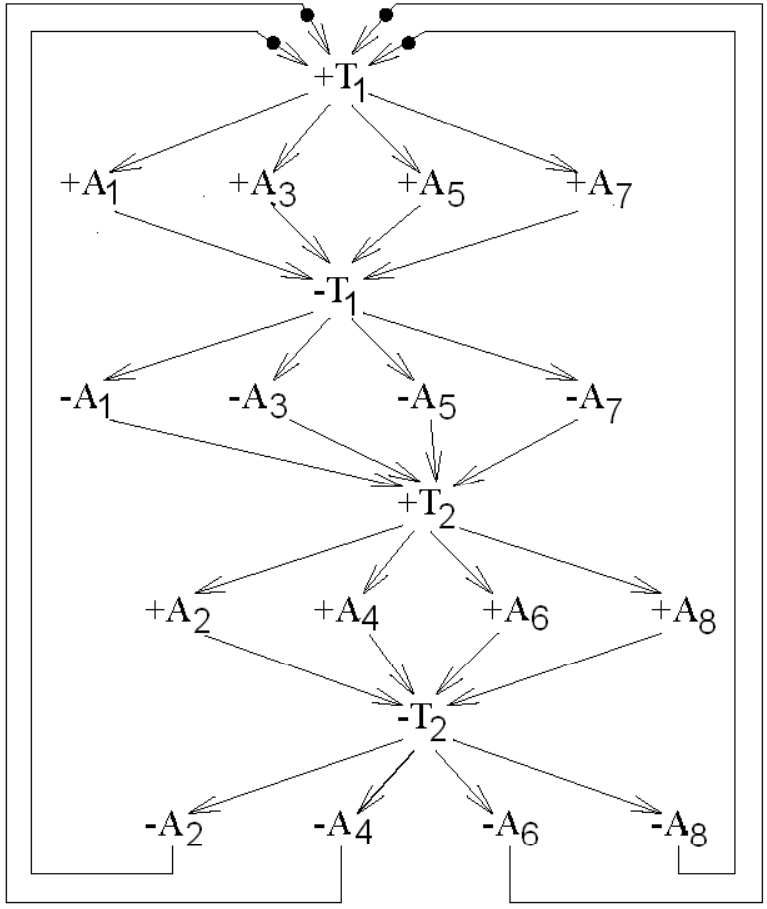


Рис.12.4. Сигнальный граф параллельной двухтрековой двухфазной синхронизации одномерного массива из 8 автоматов.

В этом графе сигналы $\pm T_1$ и $\pm T_2$ представляют переходы сигналов синхронизации T_1 и T_2, а события $+A_i$ и $-A_i$ имеют смысл переходных процессов в автоматах A_i.

Синхронизация в узловых вершинах графа требует дополнительного времени (вершины $-T_1$, $+T_2$ ожидают сигналов окончания переходных процессов от нечетных автоматов, а $-T_2$, $+T_1$ – от четных).

Граф показывает, что на шаге k таким образом структурированного логического времени поведение четных автоматов описывается системой уравнений:

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

$$
\begin{aligned}
S_i(k) &= F_i[X_{i-1}(k), S_i(k-1), Y_{i+1}(k)],\\
X_i(k) &= f_{i,1}[X_{i-1}(k), S_i(k-1), Y_{i+1}(k)],\\
Y_i(k) &= f_{i,2}[X_{i-1}(k), S_i(k-1), Y_{i+1}(k)].
\end{aligned}
\tag{2}
$$

Для нечетных автоматов уравнения выглядят следующим образом:

$$
\begin{aligned}
S_i(k) &= F_i[X_{i-1}(k-1), S_i(k-1), Y_{i+1}(k-1)],\\
X_i(k) &= f_{i,1}[X_{i-1}(k-1), S_i(k-1), Y_{i+1}(k-1)],\\
Y_i(k) &= f_{i,2}[X_{i-1}(k-1), S_i(k-1), Y_{i+1}(k-1)].
\end{aligned}
\tag{3}
$$

Переход от системы (1) к системам (2) и (3) является чисто формальным и может быть выполнен для любой системы (1).

12.2.3. Глобально распределенная волновая синхронизация

Теперь рассмотрим проблему проектирования распределенного синхронизатора для массива из согласованных асинхронных автоматов. На рис.12.5 показан фрагмент развертки сигнального графа для структуры, представленной на рис.12.3,b и содержащей 8 автоматов.

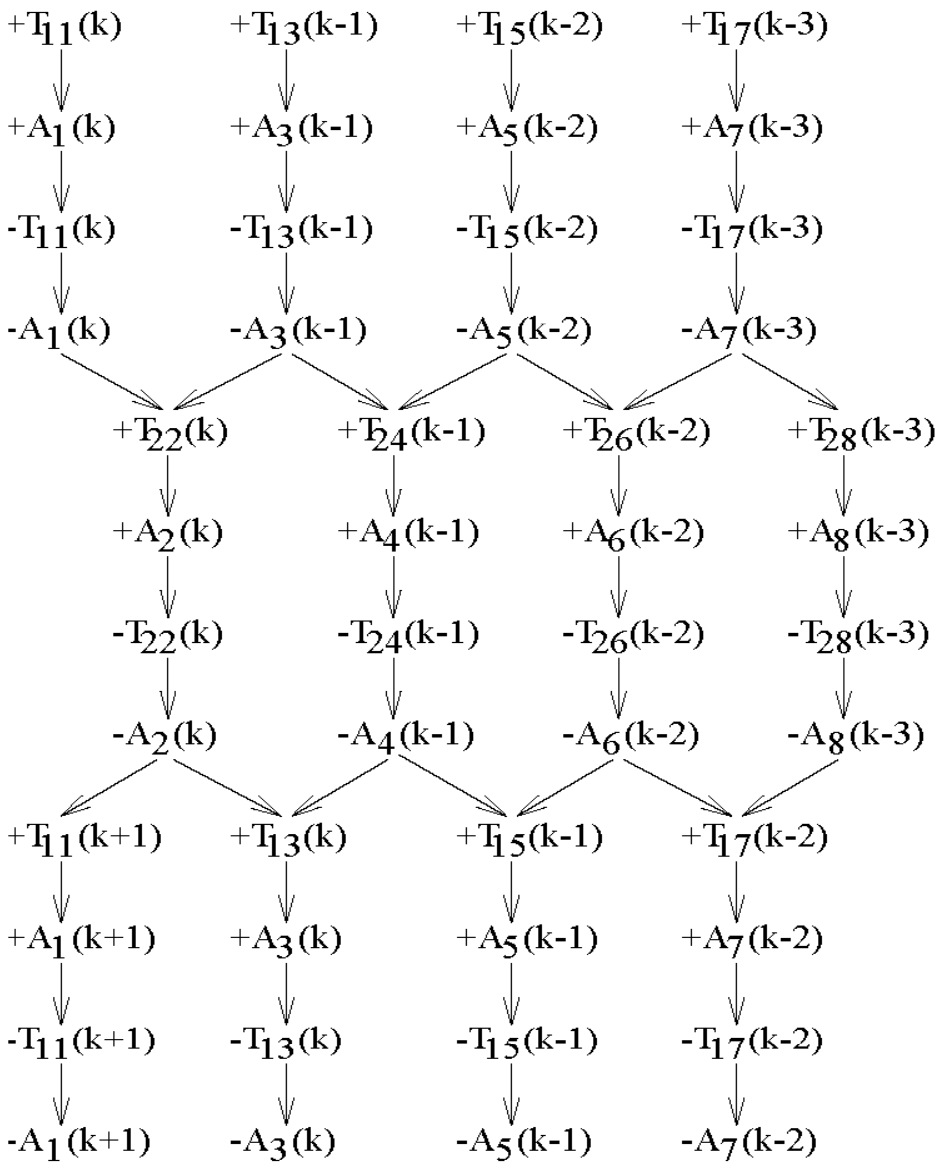


Рис.12.5. Фрагмент развертки сигнального графа, описывающего волновую логическую синхронизацию в одномерном массиве.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

В этом графе $\pm T_{ij}(k)$ означает событие, заключающееся в k-м переходе синхросигнала $T_i, i \in \{1,2\}$, (шаг k логического времени), которое наступает в j-м автомате массива. Система уравнений типа (1) приобретает вид:

$$\begin{aligned} S_i(k) &= F_i[X_{i-1}(k), S_i(k-1), Y_{i+1}(k-1)], \\ X_i(k) &= f_{i,1}[X_{i-1}(k), S_i(k-1), Y_{i+1}(k-1)], \\ Y_i(k) &= f_{i,2}[X_{i-1}(k), S_i(k-1), Y_{i+1}(k-1)]. \end{aligned} \qquad (4)$$

Нетрудно видеть, что на рис.12.5 один и то же момент логического времени существует в различных автоматах массива в различные моменты физического времени и в один и тот же момент физического времени представлены различные моменты логического времени.

Граф на рис.12.5 является спецификацией для проектирования синхростратума. Реализация синхростратума требует введения в спецификацию дополнительных переменных и специального проектирования. Одно из возможных решений выглядит следующим образом.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Пусть a_j – событие, соответствующее полному циклу синхронизации j-го автомата, т.е. $a_j = +T_{ij} \rightarrow +A_j \rightarrow -T_{ij} \rightarrow -A_j$. Тогда, следуя графу на рис.12.5, необходимая координация событий описывается маркированной сетью Петри:

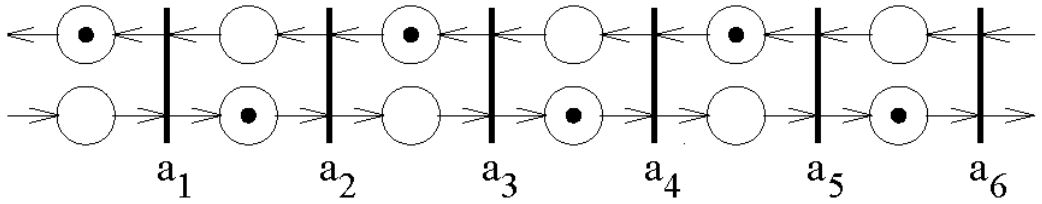


Рис.12.6 Маркированная сеть Петри, описывающая пайплайновое взаимодействие автоматов в одномерном массиве.

Это спецификация простого пайплайна. Прямой переход от спецификации к реализации, использующей ячейки Давида, дает простую схему синхростратума (см. рис.12.7).

Такой переход возможен, так как, в отличие от пайплайна на С-элеметах, в пайплайне на ячейках Давида выходной сигнал каждой j-й ячейки совершает полный цикл события a_j и только после этого начинает работать следующая (j+1)-я ячейка.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

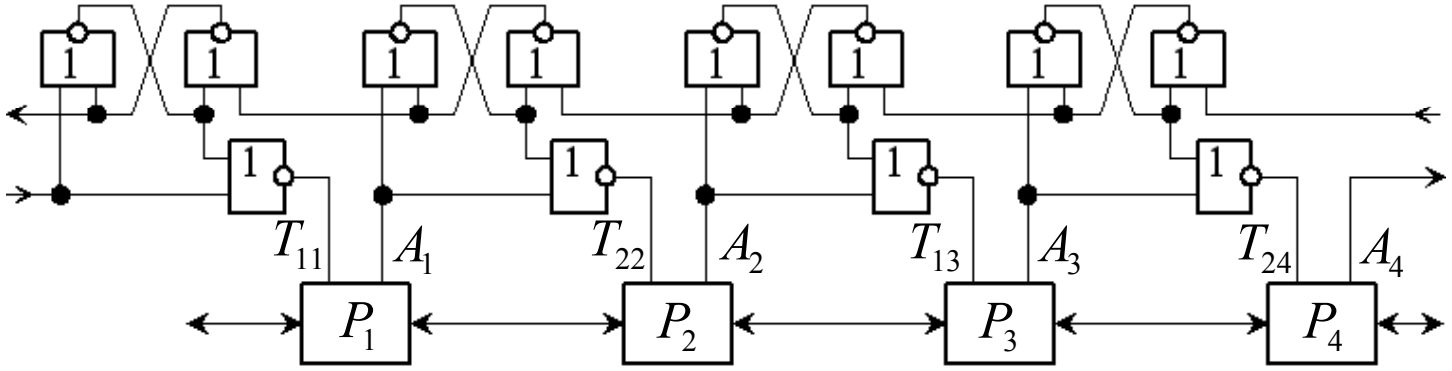


Рис.12.7. Синхростратум на распределительных ячейках.

Однако недостатком такого решения является низкая пропускная способность пайплайна. Наибольшая пропускная способность достигается при его заполнении на 1/4, т.е. если размерность массива n, то количество синхроволн логического времени в массиве равно $n/4$.

Развертка сигнального графа на рис.12.5, позволяет получить сеть Петри иного вида, которая представлена на рис.12.8. Снова методом прямой трансляции сети Петри в схему можно построить синхростратум, в котором используются три ячейки Давида на два автомата (вспомним, что цикл не может содержать меньше трех ячеек).

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

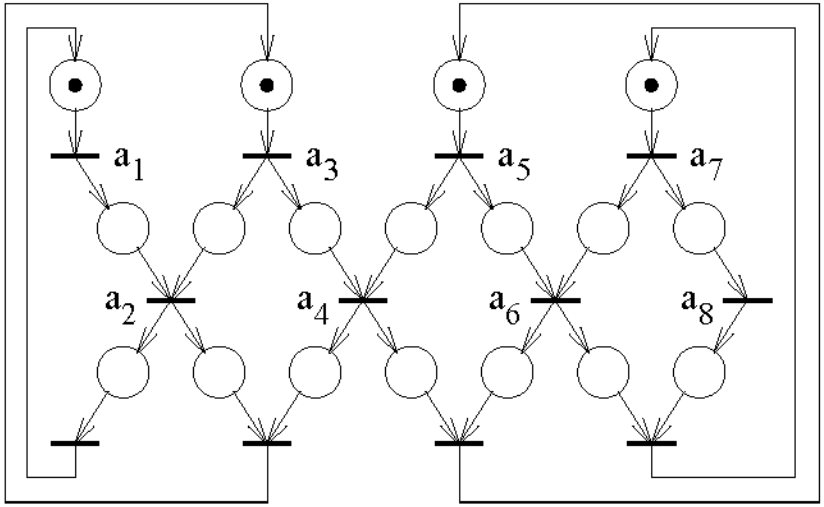


Рис.12.8. Другой вариант спецификации взаимодействия автоматов в одномерном массиве.

12.2.4. Глобально распределенная параллельная синхронизация

На рис.12.9 приведен фрагмент развертки сигнального графа, который описывает параллельную синхронизацию в отличие от синхронизации, основанной на распространении волн. Этот фрагмент отличается от фрагмента на рис.12.5 тем, что все параллельные сигналы (на одном и том же ярусе развертки) соответствуют одному и тому же моменту логического времени.

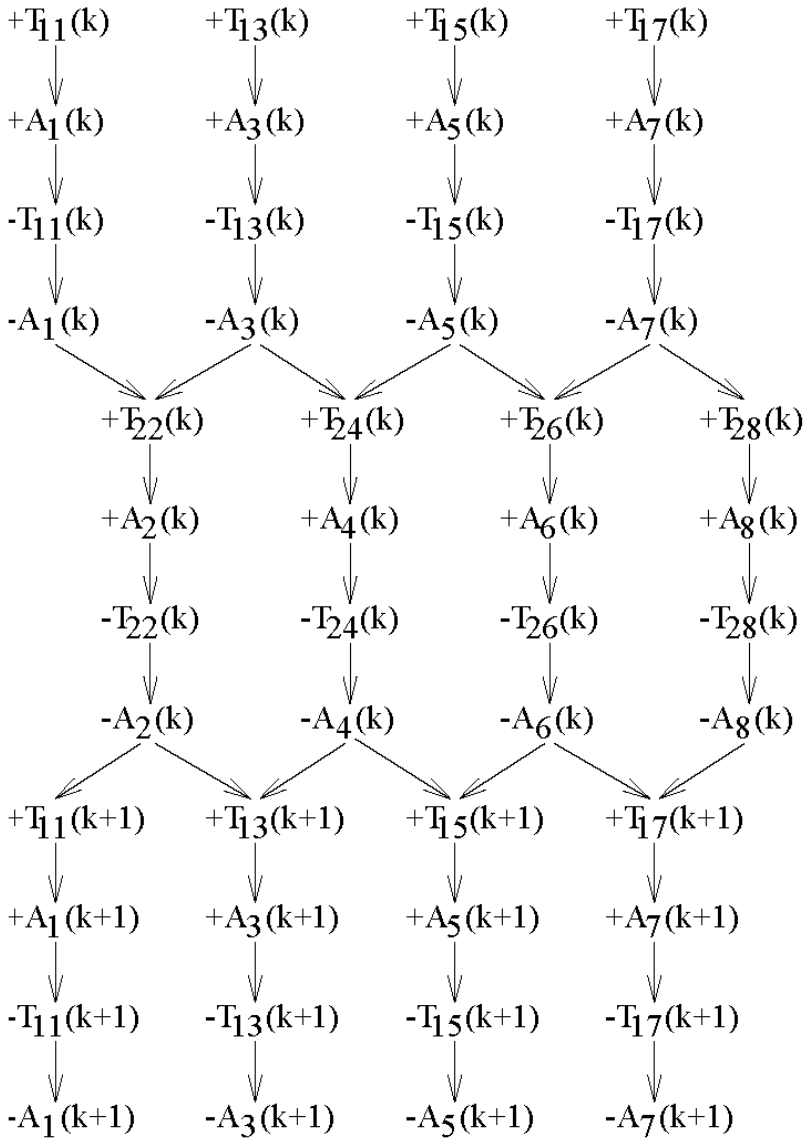


Рис.12.9. Фрагмент развертки сигнального графа для параллельной синхронизации одномерного массива.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Сравнивая рис.12.5 и рис.12.9, можно сделать заключение, что *эти графы отличаются только маркировкой моментов логического времени.*

Отсюда следует парадоксальный вывод: *одна и та же схема синхростратума в зависимости от принятого соглашения о маркировке моментов логического времени* (*т.е. в зависимости от автоматных уравнений, определяющих алгоритм*) *может обеспечить либо параллельную, либо волновую систему глобальной синхронизации.*

При параллельной синхронизации поведение автоматов тоже описывается системами уравнений (3) и (4). Заметим, что при достаточной ширине сигнального графа различным моментам логического времени может соответствовать один и тот же момент физического времени времени.

При синтезе схемы синхростратума сигнальный граф должен быть приведен к корректному виду введением в него дополнительных переменных, с помощью которых устраняются конфликты кодирования состояний схемы. Было найдено удивительно простое решение, представленное на следующем рисунке.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

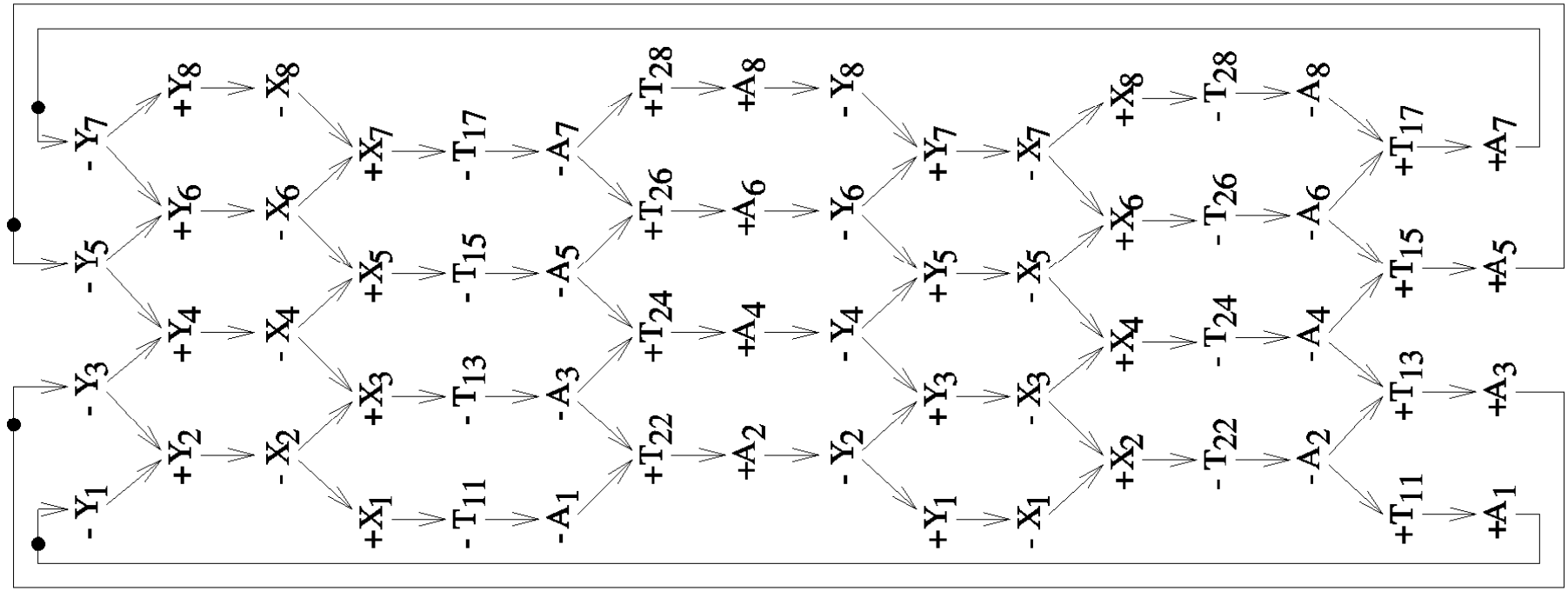


Рис.12.10. Правильный сигнальный граф для параллельной синхронизации одномерного массива.

В этом графе сигналы $\pm X_i$ и $\pm Y_i$ являются дополнительными переменными. Полученный сигнальный граф реализуется очень простой схемой, представленной на рис.12.11.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

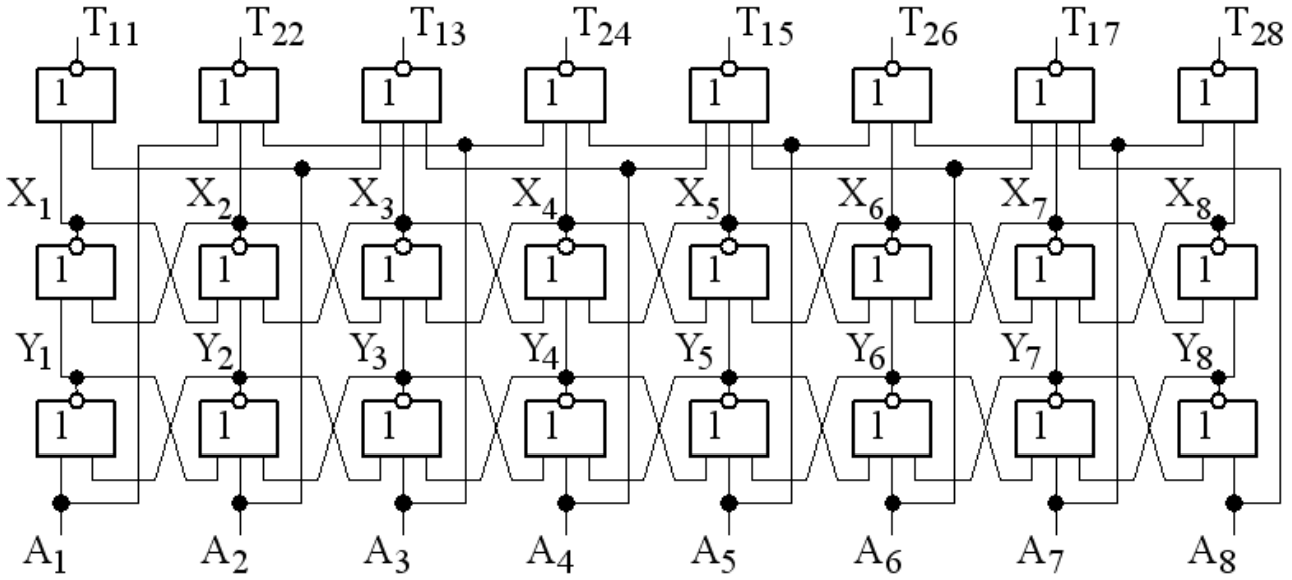


Рис.12.11. Синхростратум для параллельной синхронизации одномерного массива из 8 автоматов.

Рассмотрим синхростратум для системы из автоматов, помещенных в вершины произвольного графа. Дуги графа соответствуют связям между автоматами. Согласно принятой дисциплине синхронизации прототипа (двухтрековая, двухфазная), граф связи должен быть графом Кёнига (т.е. двудольным).

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Произвольный граф связей всегда может быть приведен к графу Кёнига, например, вставкой буферных регистров во все связи. Вставка буферных регистров может оказаться полезной независимо от типа графа взаимосоединений, особенно если цикл работы автомата значительно длиннее цикла записи в буферные регистры. Тогда сигналы $T_{1j} = 1$ будут инициировать активность автоматов, а сигналы $T_{2j} = 1$ – циклы записи в буферные регистры.

Возвратимся к рис.12.11 и рассмотрим локальные свойства схемы синхростратума, гарантирующие правильность ее поведения.

Во-первых, в слое вентилей, с выходов которых снимаются сигналы T_{ij}, связи между соседними вентилями (через автоматы A_j) обеспечивают переключение вентиля из состояния 0 в состояние 1, если и только если выходные сигналы всех соседних вентилей равны 0.

Во-вторых, переключение вентиля с выходом T_{ij} из состояния 0 в состояние 1 должно быть детерминированным. Следовательно, необходима память для сохранения его предыдущего состояния; такая память организована на двух слоях вентилей с выходами X_j и Y_j.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Каждый из этих выходов может переключиться в состояние 1 не раньше, чем два его боковых соседа переключатся в 0. Поэтому связи вентилей X_j и Y_j со своими боковыми соседями аналогичны связям между вентилями T_{ij} .

Соблюдая эти требования к локальному поведению вентилей, с учетом числа соседей каждой вершины графа связей можно построить следующую схему элемента синхростратума.

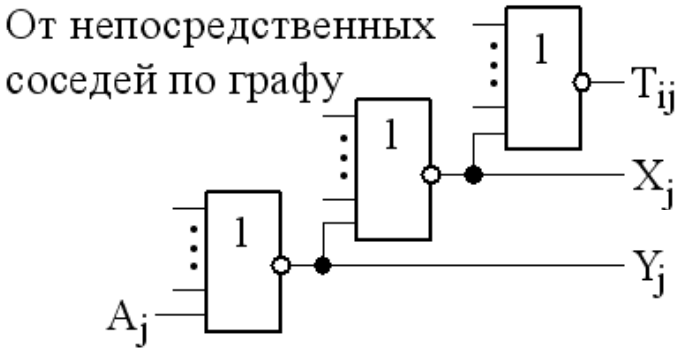


Рис.12.12. Схема элемента синхростратума для произвольного графа (Кёнига) взаимосоединений.

Логическое Проектирование Асинхронных Схем

Лекция 8

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

12.2.5. Уточнение определения глобальной синхронизации

Попытаемся дать более четкое определение концепция глобальной синхронизации. Возьмем в качестве прототипа систему синхронизации массива автоматов с двухтрековой двухфазной дисциплиной типа «ведущий-ведомый».

Такая система синхронизации разбивает все множество автоматов на два подмножества: автоматы $\{A_i\}$, тактируемые сигналами T_1, и $\{B_j\}$, тактируемые T_2. Для параллельной синхронизации развертка сигнального графа (подобного рис. 12.4 из лекции 7), может быть записана в виде:

$$\ldots \to +T_1(k) \to \{+a_i(k)\} \to -T_1(k) \to \{-a_i(k)\} \to +T_2(k) \to \{+b_j(k)\} \to -T_2(k) \to$$
$$\{-b_j(k)\} \to +T_1(k{+}1) \to \{+a_i(k{+}1)\} \to -T_1(k{+}1) \to \{-a_i(k{+}1)\} \to +T_2(k{+}1) \to \ldots .$$

Сигналы $\pm a_i(k)$ и $\pm b_j(k)$ являются сигналами завершения переходных процессов в физическом времени в соответствующих автоматах.

Введем следующие обозначения:
T_{1i} и T_{2j} – сигналы “запроса” автоматов A_i и B_j соответственно;

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

$\{T_{2j}[A_i]\}$ – сигналы запроса для подмножества автоматов B_j, являющихся непосредственными соседями автомата A_i в графе взаимосоединений;

$\{T_{1i}[B_j]\}$ – сигналы запроса для подмножества автоматов A_i, являющихся непосредственными соседями автомата B_j в графе взаимосоединений;

a_i и b_j – сигналы "ответа" автоматов A_i и B_j соответственно.

Будем говорить, что сигнальный граф поведения синхростратума специфицирует глобальную параллельную синхронизацию асинхронной системы, соответствующей синхронному прототипу, если удовлетворяются следующие условия:

1) *сигнальный граф синхронного прототипа гомоморфен сигнальному графу синхростратума относительно отображения* $\{\pm T_{lj}(k)\} \to \pm T_l(k), l \in \{1,2\}$, *и*

2) *сигнальный граф синхростратума соответствует отношению следования между событиями в каждом автомате и в его непосредственных соседях, заданному системами автоматных уравнений* ((2) *и* (3) из лекции 7):

$$+T_{1i}(k) \to +a_i(k) \to -T_{1i}(k) \to -a_i(k) \to \{+T_{2j}[A_i](k)\},$$

$$+T_{2j}(k) \to +b_j(k) \to -T_{2j}(k) \to -b_j(k) \to \{+T_{1i}[B_j](k+1)\}.$$

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Для волновой синхронизации, определяемой системой (4) (лекция 7), направление распространения синхроволн должно быть задано в каждой точке синхростратума. Поэтому для каждого автомата (A_i и B_j) множество его ближайших соседей должно быть разбито на два подмножества – подмножество источников фронта синхроволны (Bs_j (A_i) и As_i (B_j)) и подмножество ее приемников (Br_j (A_i) и Ar_i (B_j)).

Кроме того, в синхростратуме некоторые вершины должны быть назначены водителями ритма. Все ближайшие соседи таких вершин в графе взаимосоединений являются получателями фронта синхроволны. (В качестве источника синхроволн может быть использован любой цикл.)

Такое разбиение вершин графа приводит к четырехцветному графу и может быть выполнено различными способами, зависящими от выбора вершины, задающей ритм. В однородных массивах это сделать нетрудно. В произвольном графе взаимосоединений могут возникать конфликты и такое разбиение автоматов на подмножества может оказаться сложной задачей.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

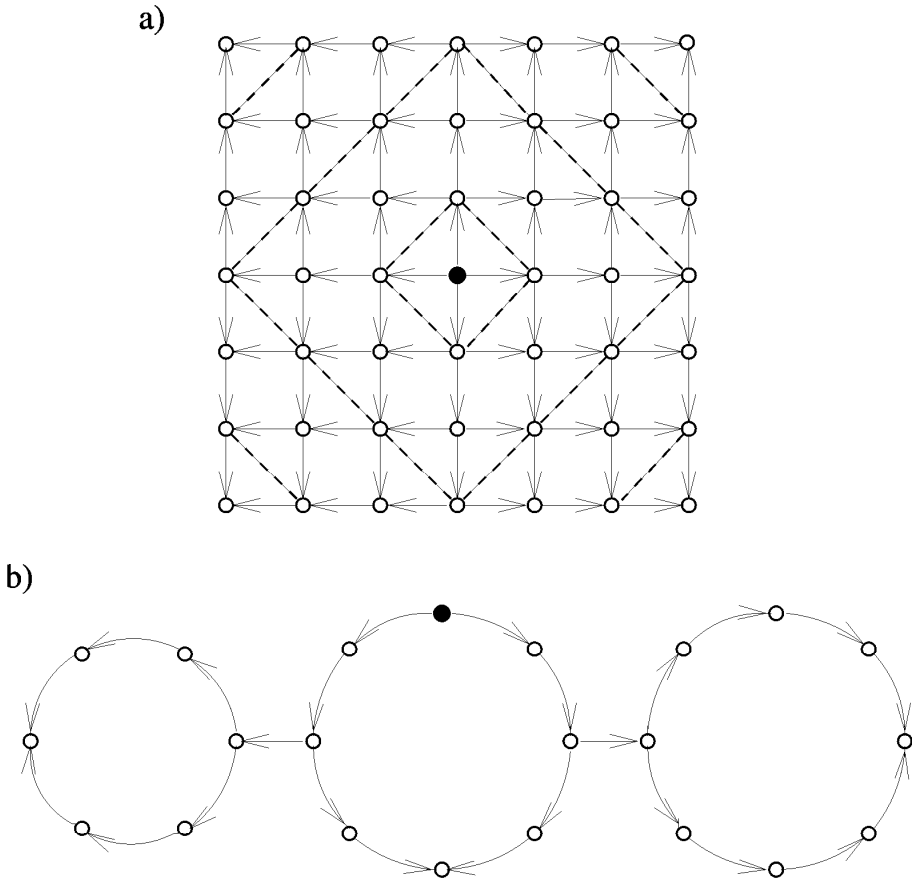


Рис.13.13. Примеры организации распространения синхроволн: a) в двумерном массиве, b) в неоднородном графе.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

В любом случае синхростратум должен обеспечивать следующий порядок событий:

$$+T_{1i}(k) \to +a_i(k) \to -T_{1i}(k) \to -a_i(k) \to \{+T_{2j}[Ar_i](k)\},$$

$$+T_{2j}(k) \to +b_j(k) \to -T_{2j}(k) \to -b_j(k) \to \{+T_{1i}[Br_j](k+1)\},$$

а для водителя ритма:

$$+T_{1i}(k) \to +a_i(k) \to -T_{1i}(k) \to -a_i(k) \to +T_{1i}(k+1).$$

Выбор между волновой и параллельной синхронизацией строго зависит от решаемой задачи, хотя параллельная синхронизация в общем случае более предпочтительна.

12.2.6. Спецификации и реализации синхростратумов

Для случая двухтрековой двухфазной синхронизации одномерного массива (рис.12.3,а, лекция 7) можно предложить другие достаточно простые реализации синхростратума.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

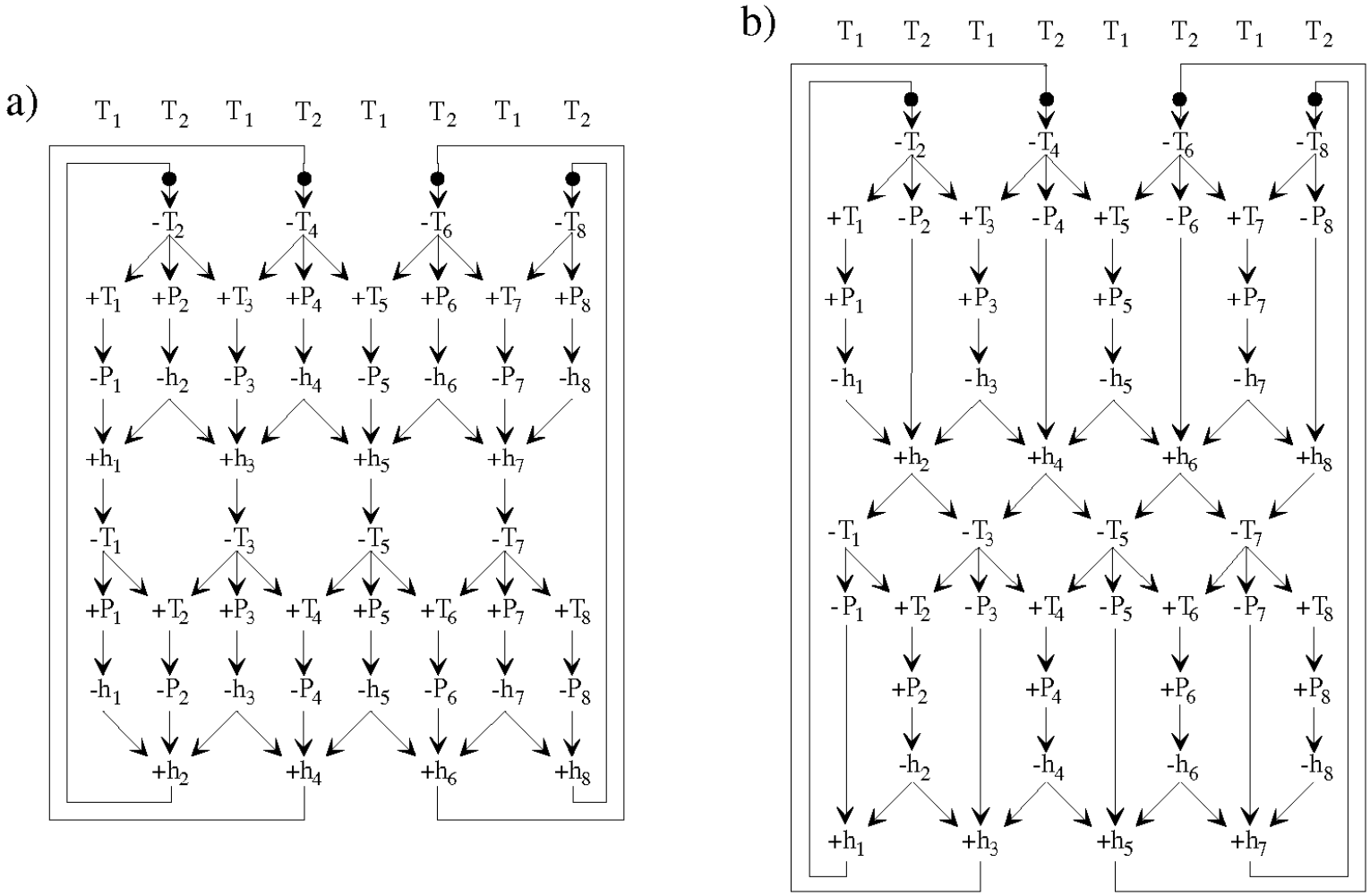


Рис.13.14. Сигнальные графы для двухтрековой двухфазной синхронизации массива из восьми автоматов: a) версия первая, b) версия вторая.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Строка над графом рис.12.14,a указывает на соответствие локальных сигналов синхростратума сигналам синхронизации T_1 и T_2 прототипа. В самом графе сигналы $\pm T_j$ – изменения выходных сигналов синхростратума, инициирующих фазы работы j-го автомата; $\pm P_j$ – изменения сигналов ответа j-го автомата; $\pm h_j$ – изменения локальных промежуточных переменных .

Построение по этому сигнальному графу модуля синхростратума дает следующие уравнения вентилей:

$$T_j = \overline{T_{j-1} \vee h_j \vee T_{j+1}}, \quad h_j = \overline{h_{j-1} \vee P_j (T_j \vee T_{j-1} T_{j+1}) \vee h_{j+1}}.$$

Модуль синхростратума, построенный по графу рис.12.14,b, проще и функции его вентилей имеют вид:

$$T_j = \overline{T_{j-1} \vee h_{j-1} h_{j+1} \vee T_{j+1}}, \quad h_j = \overline{h_{j-1} \vee P_j \vee h_{j+1}}.$$

Оба графа на рис.12.14 допускают параллельность работы нечетных и четных автоматов в противоположных фазах изменения синхронизирующих сигналов, что значительно увеличивает производительность автоматного стратума. По графам можно убедиться, что оба синхростратума вносят задержку $D_{ss} = 6\tau_g$.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

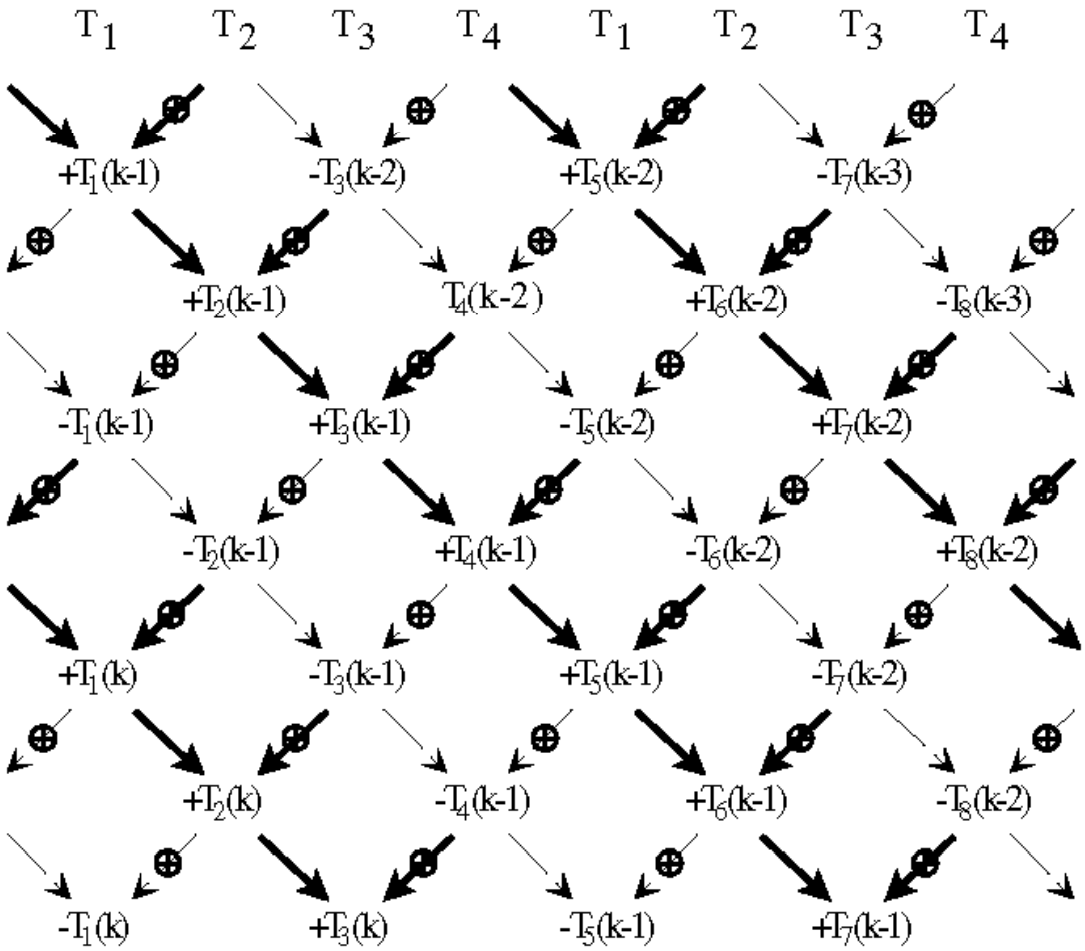
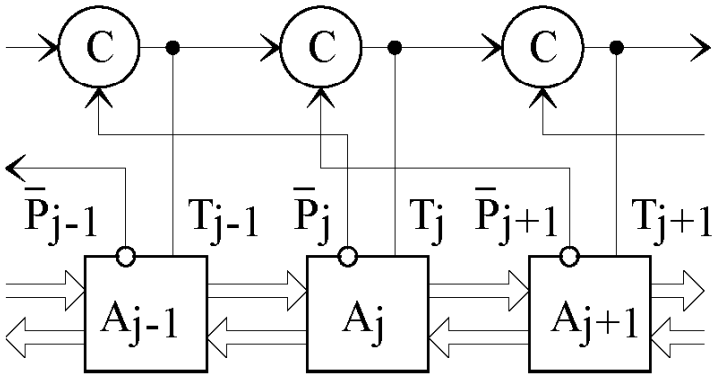


Рис.12.15. Фрагмент развертки сигнального графа для 4-трековой волновой синхронизации одномерного массива автоматов.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Этот граф является спецификацией поведения хорошо известной схемы одномерного пайплайна. Сигналы $\pm T_j(k)$ изменения T_j в моменты k логического времени. Символом $\otimes$ маркированы связи, в разрывы которых вставлены автоматы A_j с помощью сигналов запрос-ответ. Жирные стрелки, определяют условия изменения состояний j-х автоматов (состояний их основных регистров R_{1j}); штрих-стрелки, определяют условия для запоминания измененных состояний в j-х автоматах (переписи состояний в регистры R_{2j}). Легко видеть, что модуль синхростратума реализуется С-элементом: $T_j = T_{j-1}\overline{P}_{j+1} \vee T_j(T_{j-1} \vee \overline{P}_{j+1})$.



$$D_{ss} = 4\tau_c = 8\tau_{gate}$$

Рис.12.16. Одномерный асинхронный массив автоматов для синхронного прототипа с 4-трековой 2-фазной синхронизацией.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Следующий рисунок демонстрирует две возможные структуры для распространения синхроволн в двумерном массиве.

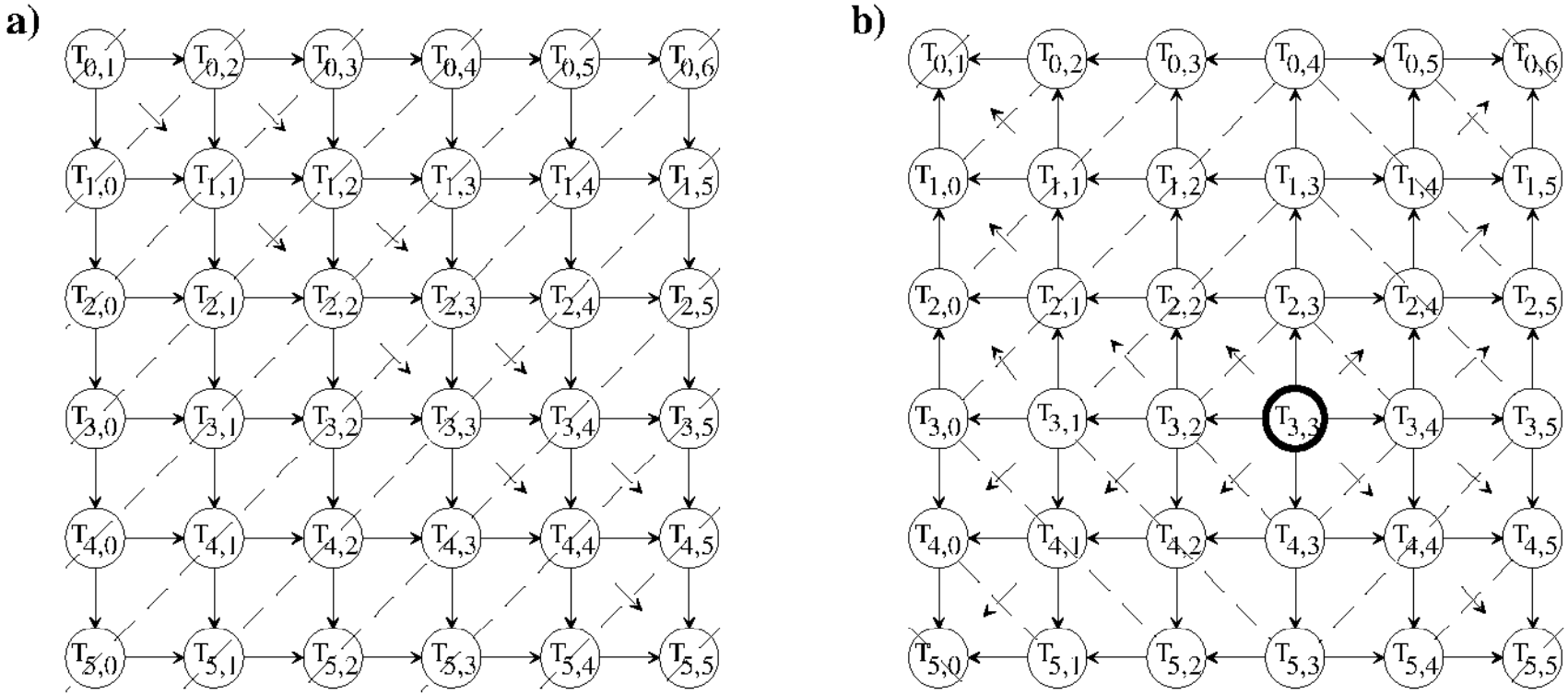


Рис.12.17. Двумерный массив: a) с линейным фронтом синхроволн; b) с одним источником волн (водителем ритма).

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Из графа на рис.12.17,а непосредственно следует автоматное уравнение для элемента массива, формирующего сигналы $T_{i,j}$:

$$T_{i,j} = T_{i-1,j}T_{i,j-1}\overline{T}_{i+1,j}\overline{T}_{i,j+1} \vee T_{i,j}(T_{i-1,j} \vee T_{i,j-1} \vee \overline{T}_{i+1,j} \vee \overline{T}_{i,j+1}).$$

Это уравнение четырехвходового С-элемента. Если в нем заменить сигналы $\overline{T}_{i+1,j}$ и $\overline{T}_{i,j+1}$ соответственно сигналами $\overline{P}_{i+1,j}$ и $\overline{P}_{i,j+1}$, что эквивалентно вставлению автоматов $A_{i+1,j}$ и $A_{i,j+1}$ их хендшейк-сигналами в разрывы проводов соответствующих сигналов, то получим уравнение модуля синхростратума.

Для структуры распространения волн от водителя ритма (рис.12.17,b) тоже нетрудно построить синхростратум, используя 4-входовые С-элементы. При получении уравнений С-элементов, определяющих межэлементные связи, следует пользоваться следующим правилом: *переменная входит в уравнение без инверсии, если соответствующая ей стрелка в графе является входящей, и с инверсией, если эта стрелка исходящая. Например, все переменные в уравнении С-элемента, которые относятся к водителю ритма, должны быть инвертированы.*

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

В разделе 12.2.4 был продемонстрирован способ преобразования волновой синхронизации в параллельную изменением начальной маркировки моментов логического времени в сигнальном графе спецификации. Однако такое преобразование может привести к необходимости изменения уравнений клеточных автоматов.

Фрагмент развертки сигнального графа синхростратума, соответствующего синхронному прототипу одномерного массива с четырехтрековой двухфазной параллельной синхронизацией, приведен на следующем слайде.

Вершины этого графа, в которых изменяются моменты логического времени, вызывают переходы в автоматах (изменяются состояния основных регистров). Таким образом, сигнальный граф устанавливает следующие правила переходов состояний автоматов и обмена данными:

- для автоматов с номерами $j = 4k+1$ и $j = 4k+2$, $k = 0,1,2,\ldots$:
  если $T_j = 1$, то $R_{1,j} := F_j(R_{2,j-1}, R_{2,j}, R_{2,j+1})$; если $T_j = 0$, то $R_{2,j} := R_{1,j}$;
- для автоматов с номерами $j = 4k+3$ и $j = 4k+4$, $k = 0,1,2,\ldots$:
  если $T_j = 0$, то $R_{1,j} := F_j(R_{2,j-1}, R_{2,j}, R_{2,j+1})$; если $T_j = 1$, то $R_{2,j} := R_{1,j}$.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

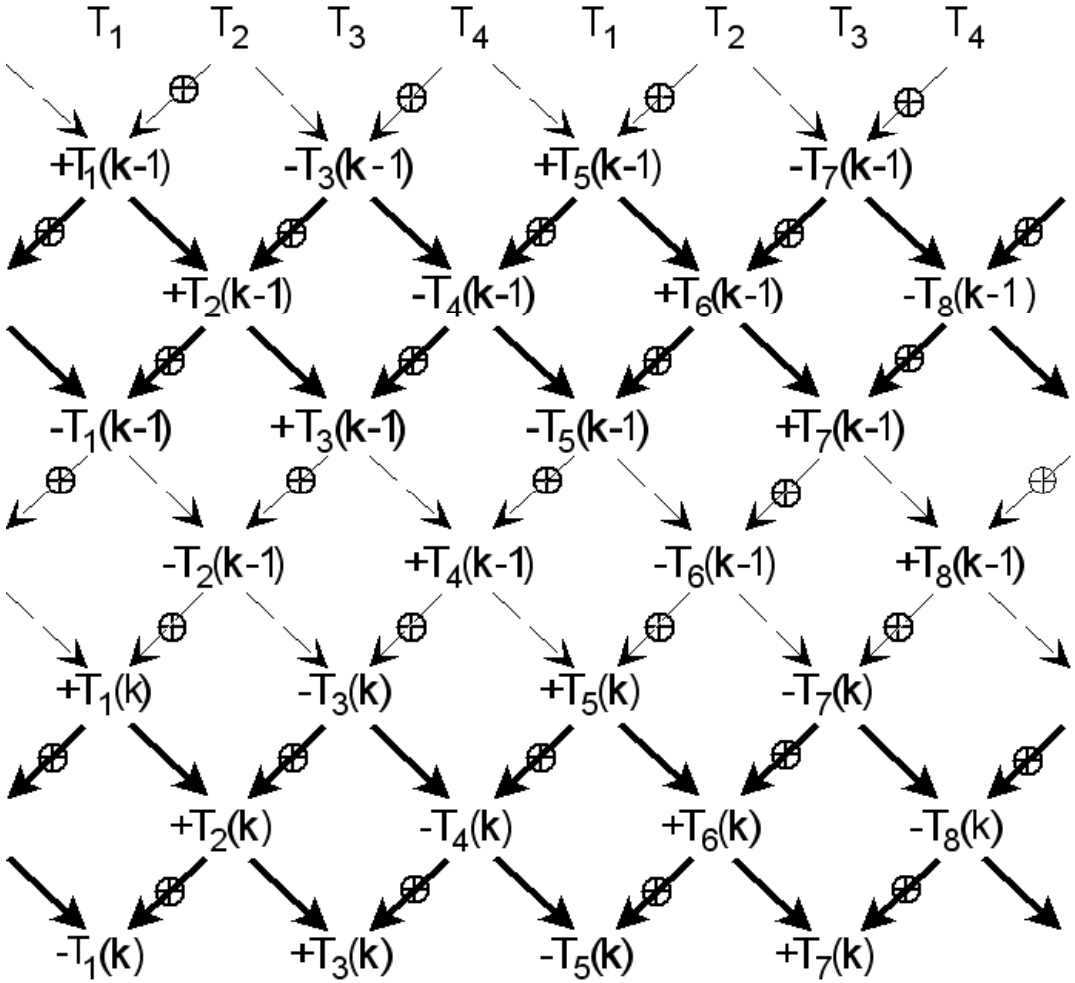


Рис.12.18. Развертка сигнального графа для 4-трековой двухфазной параллельной синхронизации одномерного массива.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

12.3. Пример применения подхода ГАЛП

Рассмотрим применение ГАЛП подхода к построению логической схемы очень быстрого асинхронного буферного запоминающего устройства типа FIFO (первым пришел – первым вышел). Это FIFO является регистровым пайплайном с управляющей логикой, обеспечивающей максимальную пропускную способность. Оно проектировалось для интерфейсного модуля, реализующего асинхронный транспортный механизм передачи данных по широкому оптоволоконному каналу связи (ATM-over-fiber).

12.3.1 Протокол передачи данных

Входные данные поступают в виде 8-байтных слов (64 бита) с частотой 300 Мгц. Каждое слово сопровождается изменением значения сигнала *A*. Каждое изменение этого сигнала сопровождает изменение значения входного слова, как показано на временной диаграмме рис.12.19. Легко видеть, что сигнал A изменяется с частотой 150 мегагерц (период равен 6.6 нсек).

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

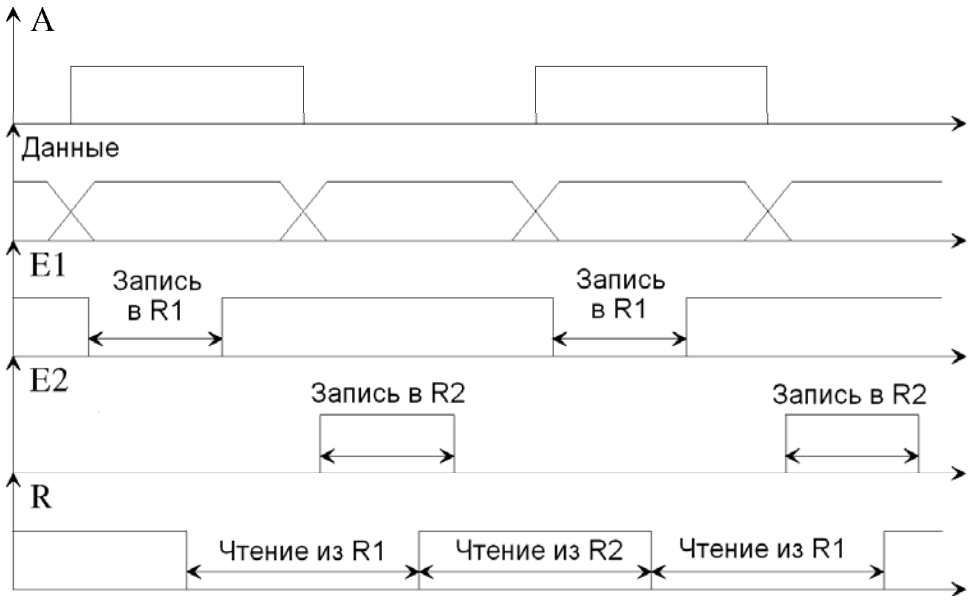


Рис.12.19. Временная диаграмма.

Используются два приемных регистра *R*1 и *R*2. На фронте сигнала *A* данные записываются в регистр *R*1, а на спаде – в *R*2. Запись в *R*1 (сигналам *E*1) совпадает во времени с чтением из *R*2 и наоборот, запись в *R*2 (сигналом *E*2) совпадает во времени с чтением из *R*1. Таким образом, предлагается структура FIFO с расщеплением пути (трека) прохождения данных типа “один-два-один”.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Выходная головка FIFO должна обеспечивать альтернативное чтение данных из двух треков, запрашивая данные из них с помощью сигнала запроса *R*, собирать данные в один трек и вырабатывать для них сигнал сопровождения.

12.3.2 Основные схемы

Поведение FIFO в основном зависит от типов триггеров (защелок) его регистров и от способа формирования разрешающего сигнала *En*. В последние годы стало популярным использовать при проектировании асинхронных схем защелки со слабыми транзисторами (например, защелка Свенссона). В данном случае их использование дает возможность сократить емкостную нагрузку на сигнал разрешения *En* и передавать бит данных только по одному проводу. В дальнейшем будем использовать защелку, представленную на рис.12.20,a.

Для сбора данных из двух треков в один потребуется также мультиплексор, который может быть построен в соответствии с рис.12.20,b.

Обе эти схемы требуют для своей работы два разрешающих сигнала: сигнал *En* и его инверсии $\overline{En}$.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

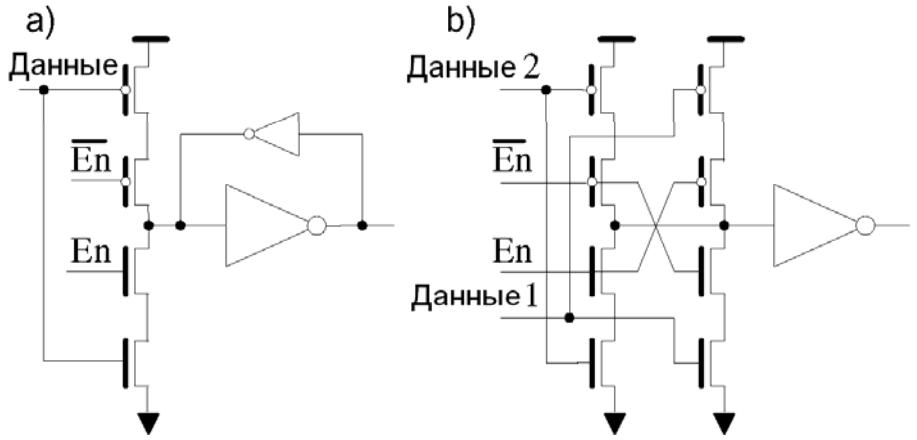


Рис.12.20. Схемы: a) защелки и b) мультиплексора.

Время переключения сигнала *En* определяется двумя факторами.
Во-первых, поведением провода этого сигнала как длинной линии с распределенными *RC*-параметрами. Задержка распространения сигнала по линии определяется как $\tau_l = RCl^2 / 2$, где R и C – распределенные параметры на единицу длины и l – длина линии.

Во-вторых, уменьшение задержки линии ограничено допустимой плотностью тока в проводе. Для того, чтобы зарядить емкость C до напряжения V током I, необходимо время $\tau_l = CV / I$.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

Задержка линии, вызванная обоими факторами, может быть сокращена за счет сегментирования линии и введения промежуточных инверторов.

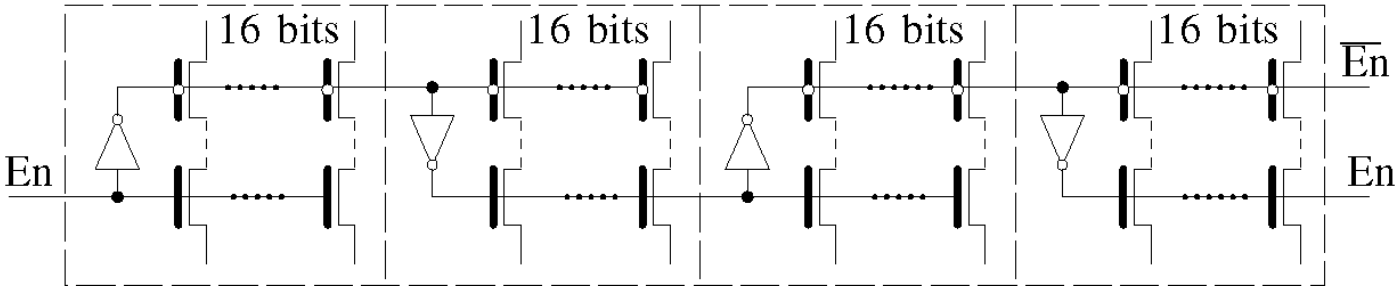


Рис.12.21. Реализация разрешающего сигнала *En*.

Существует еще одна причина для сегментации провода разрешающего сигнала. Асинхронное FIFO предполагает формирование сигнала момента завершения процесса записи информации в регистр, что требует дополнительного оборудования и внесения дополнительной задержки. Время, требуемое на распространение сигнала *En*, значительно больше времени переключения защелок регистра. Сегментация провода этого сигнала обеспечивает последовательный порядок его прохождения на защелки и задержка сигнала *En* может рассматриваться как встроенная задержка.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

12.3.3. Стратегия синхронизации и проектирование синхростратума

Синхронным прототипом для FIFO может служить синхронный сдвиговый регистр, в котором сигнал *A* является синхросигналом. Как известно, структура синхростратума зависит от способа синхронизации прототипа. Выберем в качестве прототипа двухтрековый сдвиговый регистр с 6-фазной синхронизацией.

Каждый трек представляет собой сдвиговый регистр с трехфазной синхронизацией. Четные синхросигналы управляют сдвигом данных в одном регистре, а нечетные – в другом. На рис.12.22 стрелки показывают условия формирования фронтов и спадов сигналов в асинхронной реализации. Штриховые стрелки указывают направление движения данных в обоих треках.

По временным диаграммам, представленным на рис.12.19 и рис.12.22, можно построить сигнальный граф, моделирующий поведение всего устройства. На рис.12.23 приведен сигнальный граф, специфицирующий поведение FIFO из 8 регистров.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

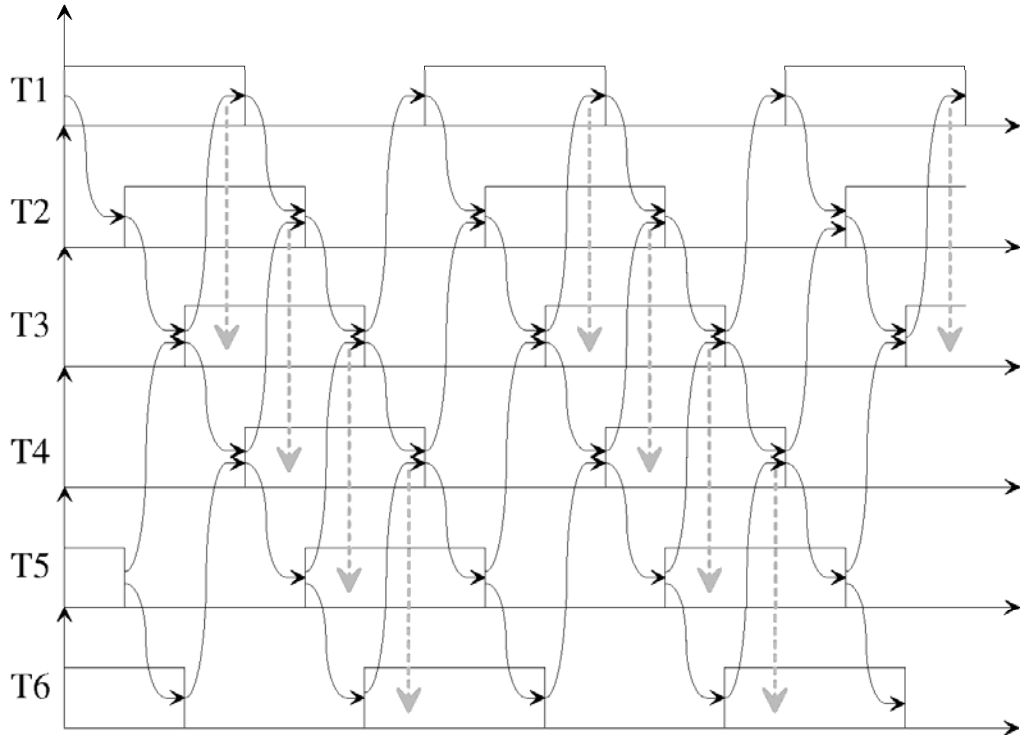


Рис.22 Шестифазная синхронизация двухтрекового сдвигового регистра.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

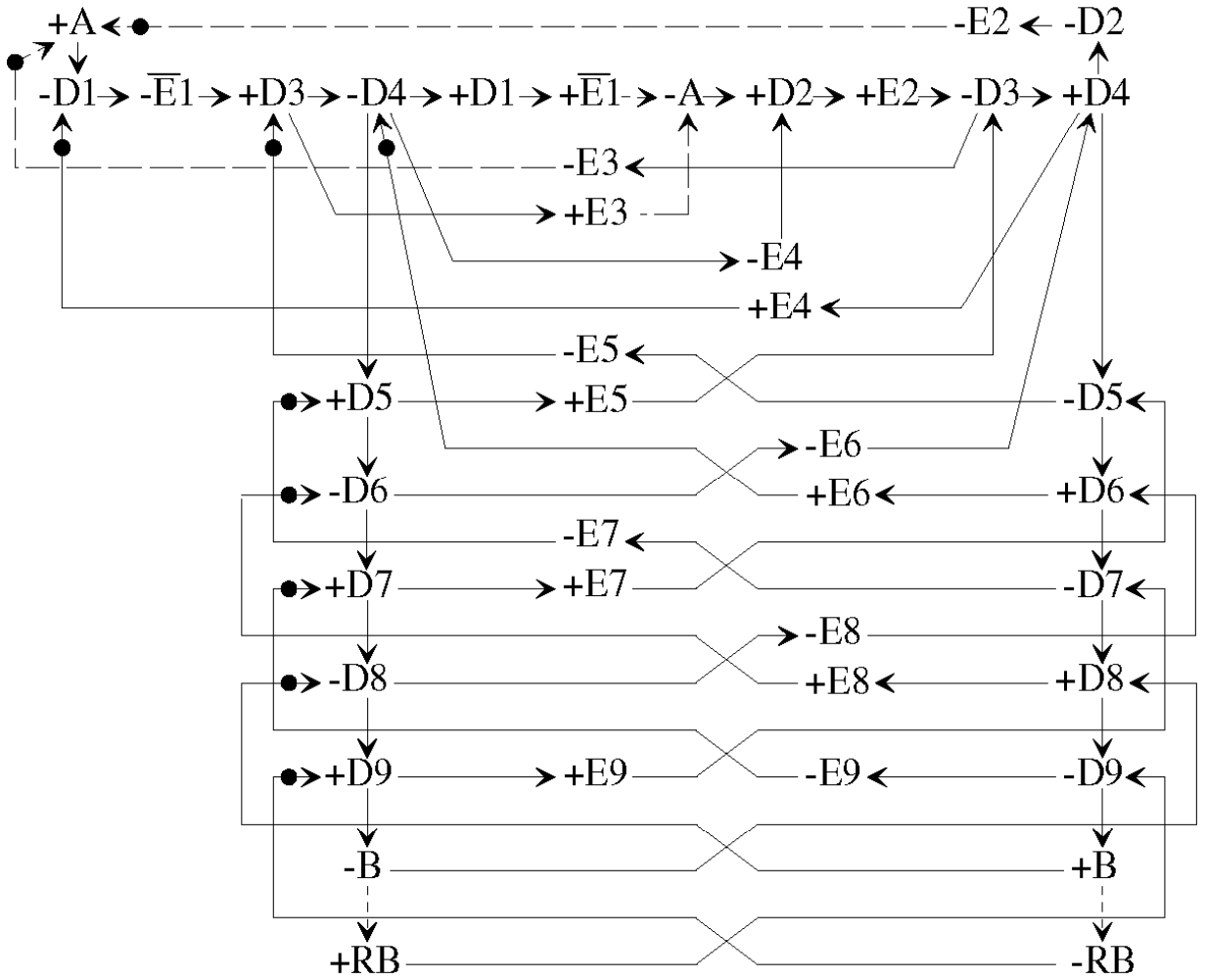


Рис.12.23. Сигнальный граф поведения FIFO из 8 регистров.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

В этом графе сигналы D_i соответствуют сигналам T_j на рис.12.22 ($j = i_{\mathrm{mod}\,6}$), а сигналы E_i являются сигналами ответа на запросы D_i. Заметим, что вход FIFO не связан хендшейком с источником данных. Поэтому предполагается, что за время между двумя соседними изменениями сигнала A, все переходные процессы во входных схемах FIFO завершиться.

Внутри FIFO данные продвигаются к выходу в силу внутренней динамики и само FIFO функционирует как схема, не зависящая от задержек элементов. Дуги, показанные в сигнальном графе штриховыми линиями, введены лишь для обеспечения корректности графа, что необходимо при использовании процедуры формального синтеза схем управления.

По спецификации на рис.12.23 можно построить логические функции элементов устройства, схема которого приведена на рис.12.24:

$$A = \overline{E_2 \vee \overline{E}_1 \cdot E_3}; \quad \overline{E}_1 = D_1; \quad D_1 = \overline{A \cdot E_4 \cdot D_4};$$

$$D_2 = \overline{A \vee E_4 \vee D_4}; \quad D_3 = \overline{E_2 \cdot E_5 \vee D_4 \cdot (E_2 \vee E_5)};$$

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

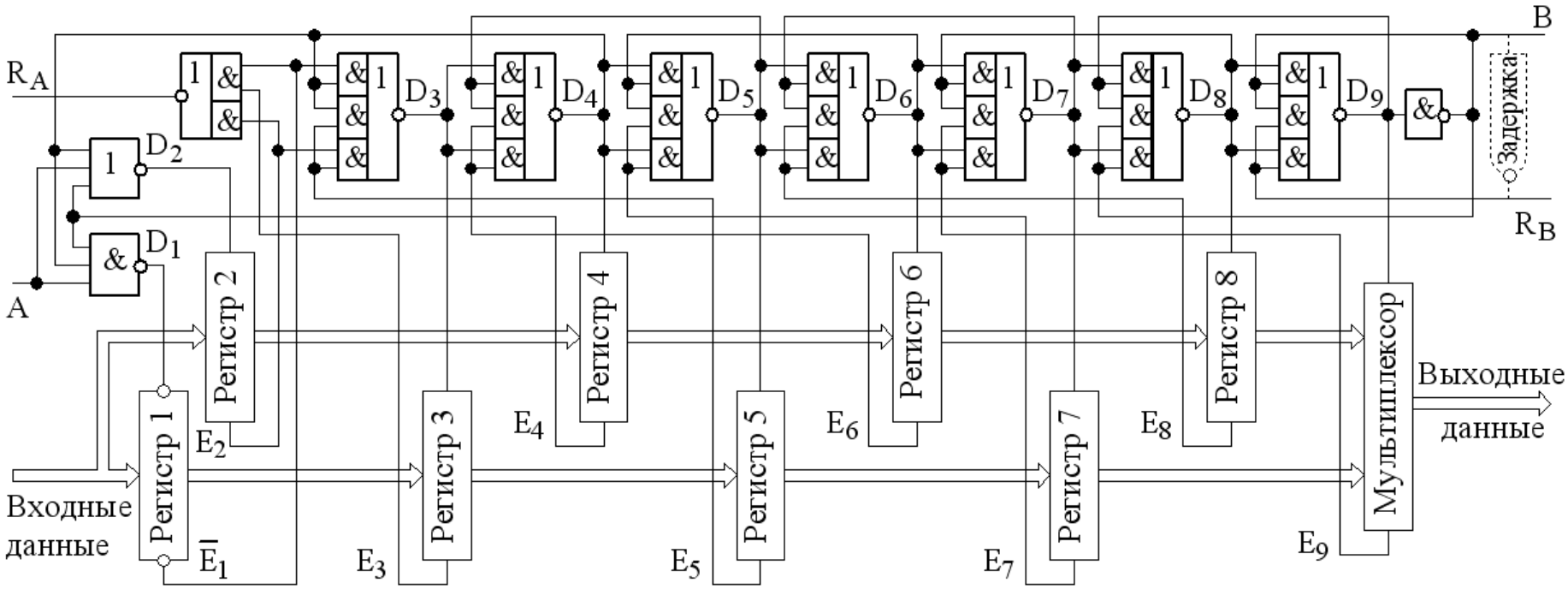


Рис.12.24. Асинхронное двухтрековое FIFO из 8 регистров.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

$$D_j = \overline{D_{j-1} \cdot E_{j+2} \vee D_{j+1} \cdot (D_{j-1} \vee E_{j+2})}, \quad 3 < j < N-1;$$

$$D_{N-1} = \overline{D_{N-2} \cdot B \vee D_N \cdot (D_{N-2} \vee B)};$$

$$D_N = \overline{D_{N-1} \cdot R_D \vee B \cdot (D_{N-1} \vee R_B)};$$

$$E_j = D_j, \quad j = 2,3,...,N; \quad B = \overline{D}_N; \quad R_B = \overline{B}.$$

В этой схеме сигнал A является внешним сигналом, а его функция, полученная по сигнальному графу, присвоена выходному сигналу R_A входной головки (запрос следующего слова). Сигнал R_A может быть использован для проверки правильности приема данных. В выходную головку схемы включена “задержка” для ограничения скорости чтения выходных данных. Эта задержка может быть заменена сигналами от среды, потребляющей данные, если среда взаимодействует с FIFO по принципу хендшейка.

Анализируя граф на рис.12.23, можно определить длительность полного цикла для сигнала A (два изменения значения), которая равна $8\tau_g + 4\tau_l$, где τ_l – задержка прохождения сигнала E_j по регистру R_j . Время цикла приема нового слова в любой регистр FIFO равно $6\tau_g + 2\tau_l$.

12. Проектирование глобально асинхронных систем с произвольной локальной синхронизацией

12.4. Заключение

На основании вышеизложенного можно сделать заключение, что практически для *любой синхронной системы, представляющей собой массив блоков* (*реализующих процессы*) *любой размерности с произвольным графом Кёнига межблочных соединений, может быть спроектирована система асинхронной глобальной синхронизации* (*координации*). Для этого достаточно снабдить блоки системы сигналами окончания в них переходных процессов и построить синхростратум из модулей, например, типа рис.12.

При изложении ГАЛП методологии основное внимание было сфокусировано на “Глобальной Асинхронности”. “Локальная Произвольность” подразумевает широкие возможности по организации взаимодействия по принципу хендшейка между синхростратумом и процессорным стратумом. С этой целью могут быть использованы самосинхронные устройства, токовые сенсоры, старт-стопные локальные генераторы (часы), параллельные инкорпорированные задержки и др.

Теория Логического Проектирования

Часть 2:

Применение многозначной логики при проектировании фаззи-контроллеров

Лекция 9

Литература:

[1]. *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, by Ronald R. Yager, Lotfi A. Zadeh (Editor), Kluwer International Series in Engineering and Computer Science, 165, Jan. 1992, 356 p.

[2]. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*, by George J. Klir (Editor), Bo Yuan (Editor), Selected Papers by Lotfi A. Zadeh (Advances in Fuzzy Systems – Applications and Theory), World Scientific Pub Co.; Vol. 6, June 1996, 826 p.

[3]. Varshavsky, V.; Marakhovsky, V.; Levin, I., Saito, H., "Multi-Valued Logic Approach to Fussy Controllers Implementation", *WSEAS Transaction on Electronics*, Issue 6, Vol.4, June 2007, pp. 109 – 133.

[4]. *Fuzzy Logic Technology and Applications*, by Robert J. Marks II (Editor), IEEE Technology Update Series, Selected Conference Papers, 1994, 575 p.

1. *Введение*

Устройства управления, построенные на основе *фаззи логики* (*фаззи контроллеры*), нашли очень широкое распространение в различных областях применения. *Фаззи* (Fuzzy) – это калька с английского, означает *нечеткий*, *размытый*.

Фаззи логика является методологией для выражения законов функционирования систем в лингвистических терминах вместо дифференциальных уравнений.

Она применяется в тех случаях, когда не представляется возможным построить адекватную математическую модель управления, и часто применяется в тех случаях, когда не требуется высокая точность управления. Тем не менее, иногда она обеспечивает даже большую точность, чем традиционный подход, рекомендуемый теорией автоматического управления.

Примеры продуктов, содержащих фаззи контроллеры

(из http://www.makinhole.com/thdt-1.htm)

Продукт	Компания
Кондиционеры	Hitachi, Matsushita, Mitsubishi, Sharp
Самолетные системы	Rockwell Corp.
Анти-блок. тормозов	Nissan
Автомоб. двигатели	Nissan
Автомоб. трансмиссия	Honda, Mitsubishi, Nissan, Saturn, Subaru
Управ. цемент. печами	Mitsubishi Chemical
Химические смесители	Fuji Electric
Копир. машины	Canon
Круиз-контроль	Isuzu, Nissan, Mitsubishi
Посудомоеч. машины	Matsushita
Сушилки	Matsushita

продукт	компания
Управление экскалатором	Fujitec, Mitsubishi Electric, Toshiba
Управление производством	Omron
Система диагностики (гольф)	Maruman Golf
Распознавание рукопис. текста	Sony
Система упр-я здоровьем	Omron
Увлажнитель воздуха	Casio
Управ-е прокатным станом	Nippon Steel
Керосиновые нагреватели	Matsushita
Микроволновые печи	Hitachi, Matsushita, Sanyo, Sharp, Toshiba
Плазменная гравировка	Mitsubishi Electric
Холодильники	Matsushita, Sharp
Рисоварки	Matsushita, Sanyo
Душевые системы	Matsushita

Продукт	Компания
Фото-камеры	Canon, Minolta
Упр. причалив. шатлов	NASA
Торговля акциями	Yamaichi
Упр-е системы метро	Hitachi
Телевидение	Goldstar, Hitachi, Samsung, Sony
Переводчики	Epson
Тостеры	Sony
Система упр-я движением	Matsushita
Пылесосы	Hitachi, Matsushita, Toshiba
Видеокамеры	Canon, Matsushita, Sanyo
Стиральные машины	Goldstar, Hitachi, Matsushita, Samsung, Sanyo, Sharp

1. *Введение*

Методология построения схем управления на основе нечеткой логики подразумевает последовательное выполнение трех этапов: фаззификации, нечеткого вывода и дефаззификации.

1. ***Фаззификация*** – преобразование аналоговых (непрерывных) входных переменных в лингвистические переменные. Например, преобразование температуры в лингвистические термины (переменные) *холодно, тепло, горячо* или преобразование скорости в термины *отрицательная большая* (*NB*), *отрицательная маленькая* (*NS*), *нулевая* (*Z*), *положительная маленькая* (*PS*), *положительная большая* (*PB*).

Такое преобразование осуществляется с использованием ***функций принадлежности*** (*membership functions)*, которые определяют диапазон значений каждой лингвистической переменной и степень ее принадлежности этим значениям (ее вес). Лингвистические переменные могут иметь взвешенное участие в нескольких функциях принадлежности одновременно.

Пример функций принадлежности приведен на рис.1.

1. *Введение*

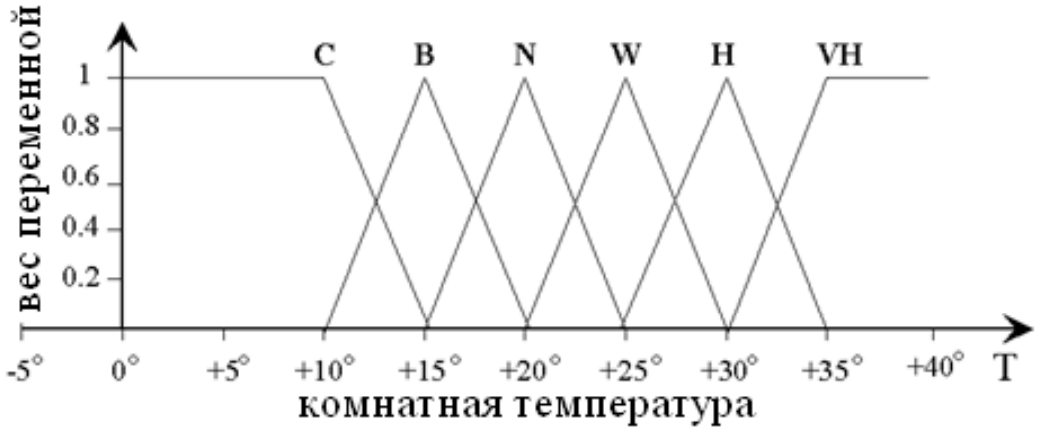


Рис.1 Функции принадлежности лингвистических переменных:
C – холодно, **B** – свежо, **N** – норма, **W** – тепло, **H** – жарко, **VH** – горячо.

2. ***Фаззи вывод*** (нечеткий вывод) отображает входные лингвистические переменные в выходные лингвистические переменные на основе системы нечетких правил типа "***ЕСЛИ*** *A* ***ТО*** *B*". Например, "ЕСЛИ температура характеризуется как «тепло» ТО скорость должна быть «положительной маленькой»" или " ЕСЛИ скорость «положительная большая» ТО сила равна «нулю»". Так как входные лингвистические переменные взвешены, то выходные лингвистические переменные также должны быть взвешенными. Существуют два традиционных подхода к фаззи выводу: подход Мамдани и подход Сугено.

1. *Введение*

Подход Момдани является более интуитивным и полагает, что выходные переменные образуют множество нечетких (лингвистических) переменных (fuzzy set). При использовании этого подхода все правила вывода содержат условную часть, или причину, (после ЕСЛИ) и последующую часть, которая является следствием (после ТО).

В подходе Сугено выходные переменные рассматриваются как множества, состоящие из одного элемента, и вторая часть правил вывода, как правило, представлена уравнениями. Этот подход более удобен для математического анализа, моделирования нелинейных систем и интерполяции.

3. На этапе ***дефаззификации*** взвешенные значения выходных лингвистических переменных, полученные при фаззи выводе, должны быть преобразованы в аналоговые (непрерывные) переменные. Эта процедура также основана на функциях принадлежности.

Известны два основных метода дефаззификации:
– метод *максимума*, в котором выходные значения определяются лингвистическими переменными с максимальным весом;

1. *Введение*

– метод *центроида*, в котором аналоговое значение вычисляется как взвешенная сумма всех активных выходных функций принадлежности.

Практически во всех применениях фаззи контроллер преобразует аналоговые входные сигналы в аналоговые выходные. Лингвистическая переменная являются субъективной характеристикой входной аналоговой переменной. Значения аналоговой переменной преобразуются на основе заданных функций принадлежности во множество взвешенных значений соответствующих лингвистических переменных. Эта процедура называется фаззификацией.

Очевидно, что при фаззификации должен использоваться аналого-цифровой преобразователь. Процедура дефаззификации должна включать обратное цифро-аналоговое преобразование.

Устройства управления, использующие фаззи логику, как правило, строятся на основе программируемых контроллеров, широко представленных на рынке вместе с системами их программирования. К недостатком таких устройств относятся достаточно высокая стоимость и низкая скорость реакции.

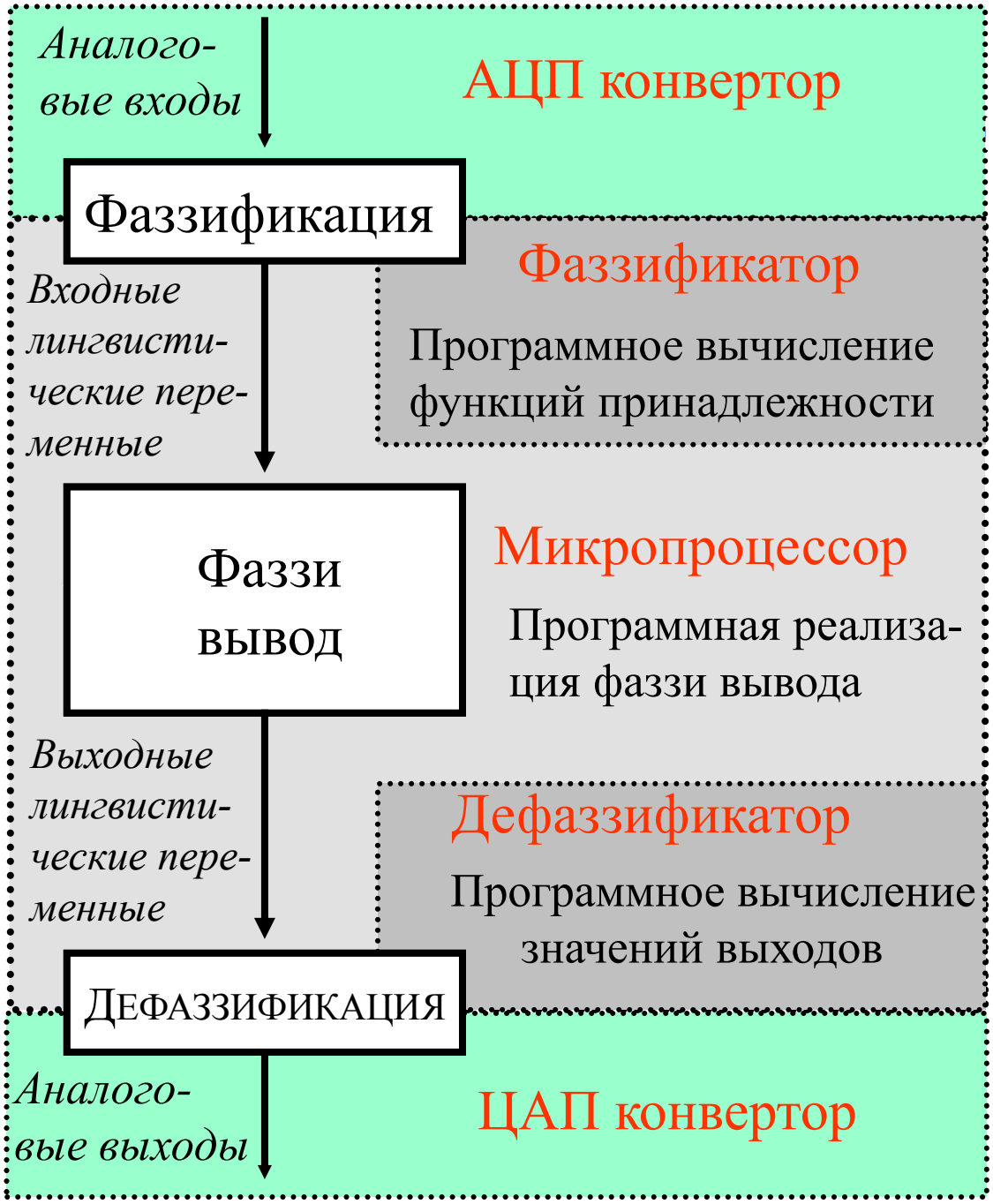


Рис.2 Стандартная реализация фаззи контроллеров.

1. *Введение*

Для большого число применений фаззи контроллеры могут быть реализованы в виде достаточно простых аналоговых устройств, не использующих микропроцессоры. Рассмотрим этот альтернативный подход.

Фаззи контроллер является детерминированным устройством, несмотря на слово «фаззи». Это означает, что оно может быть построено в виде аналоговых устройств, реализующего аналоговые функции $Y = f(x_1, x_2, ..., x_n)$.

В этой связи возникают два важных вопроса:

1. Как перейти от стандартной спецификации функции фаззи логики к спецификации соответствующей аналоговой функции?

2. Как построить реализацию на основе спецификации аналоговой функции или функции фаззи логики?

Прежде всего обратимся к функциям принадлежности. В большинстве публикаций они имеют треугольную или трапецеидальную форму (см. рис. 3).

1. *Введение*

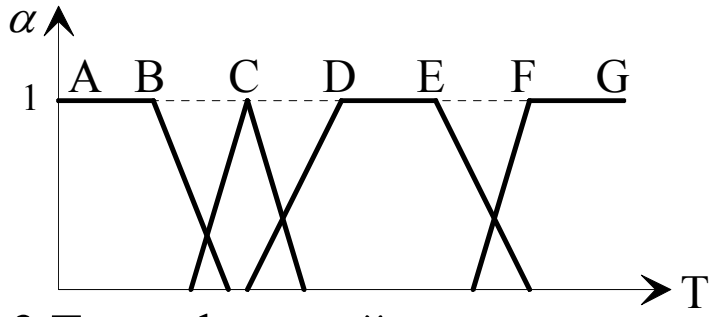


Рис.3 Типы функций принадлежности.

Лингвистические точки (переменные) A и B означают «холодно», C – свежо, D и E – «тепло», F и G – «горячо» . Эти точки определяют связь лингвистических переменных со значениями аналоговой переменной T (T – температура).

Относительно лингвистических точек двух (или более) входных аналоговых переменных строят таблицу фаззи правил, связывающую комбинации входных лингвистических переменных с выходными.

На основе функций принадлежности поставим в соответствие множествам входным и выходным лингвистическим переменным множества целых чисел, разбивающих аналогичным образом диапазон изменений соответствующих аналоговых переменных.

1. *Введение*

Тогда таблица фаззи правил преобразуется в таблицу функции многозначной логики. В клетках такой таблицы записано цифровое представление значения многозначной логической функции на наборах многозначных входных переменных.

Часто лингвистические точки функций принадлежности делят диапазон изменения соответствующей аналоговой переменной на равные отрезки, т.е. равномерно. В этом случае таблица многозначной функции будет впрямую соответствовать таблице фаззи правил.

В противном случае при построении таблицы функции многозначной логики может потребоваться либо увеличение числа градаций для равномерного разбиения диапазона на логические уровни, либо введение специальной процедуры выравнивания логических уровней (такая процедура будет обсуждаться позднее). Следовательно, таблицы фаззи правил могут быть заменены таблицами функций многозначной логики.

Как далее будет показано, аппаратная реализация таких функции обладает многими замечательными свойствами.

2. Аппаратная реализация фаззи контроллеров

2.1 Суммирующий усилитель как многозначный логический элемент

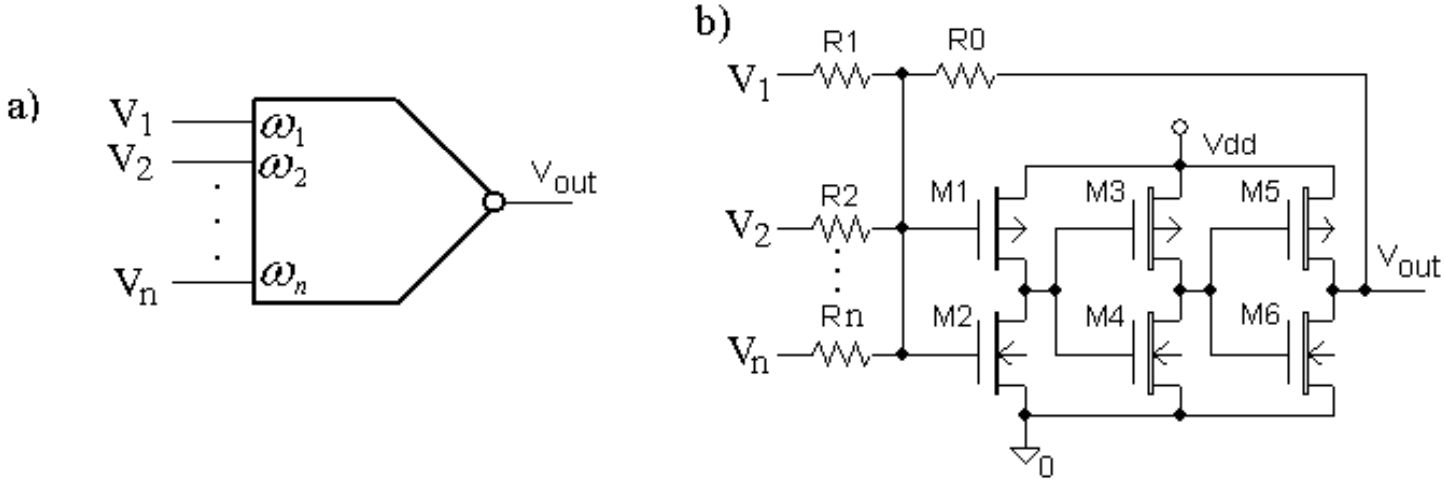


Рис.2 Суммирующий усилитель: a) обозначение,
b) КМОП реализация из симметричных инверторов.

Поведение суммирующего усилителя с точностью до членов второго порядка малости, значения которых обратно пропорциональны коэффициенту усиления усилителя при разомкнутой обратной связи (см. рис.2), определяется следующей системой уравнений:

2. Аппаратная реализация фаззи контроллеров

$$V_{out} = \begin{cases} V_{dd}, & \text{если} \quad \sum_{j=1}^{n} \frac{R_0}{R_j}(V_j - \frac{V_{dd}}{2}) \le -\frac{V_{dd}}{2} \\ \frac{V_{dd}}{2} - \sum_{j=1}^{n} \frac{R_0}{R_j}(V_j - \frac{V_{dd}}{2}), & \text{если} \quad -\frac{V_{dd}}{2} < \sum_{j=1}^{n} \frac{R_0}{R_j}(V_j - \frac{V_{dd}}{2}) < \frac{V_{dd}}{2} \\ 0, & \text{если} \quad \frac{V_{dd}}{2} \le \sum_{j=1}^{n} \frac{R_0}{R_j}(V_j - \frac{V_{dd}}{2}) \end{cases} \qquad (1)$$

Здесь V_{dd} – напряжение питания, V_j – напряжение на j-м входе, R_j – сопротивление j-го входа, R_0 – сопротивление обратной связи, и $V_{dd}/2$ – средняя точка питающего напряжения.

Зависимость V_{out} от $\sum_{j=1}^{n} \frac{R_0}{R_j} \cdot (V_j - \frac{V_{dd}}{2})$ показана на рис.3,а. Разобьем равномерно напряжение питания V_{dd} на $m = 2k + 1$ уровней. Заменим входные напряжения V_j m-значными логическими переменными $x_j = (2 \cdot V_j - V_{dd})k / V_{dd}$ и выходное напряжение V_{out} m-значной переменной y, введя обозначение $R_0/R_j = \omega_j$. Тогда система (1) может быть представлена в виде (2). Графическое изображение (2) показано на рис.3,b.

2. Аппаратная реализация фаззи контроллеров

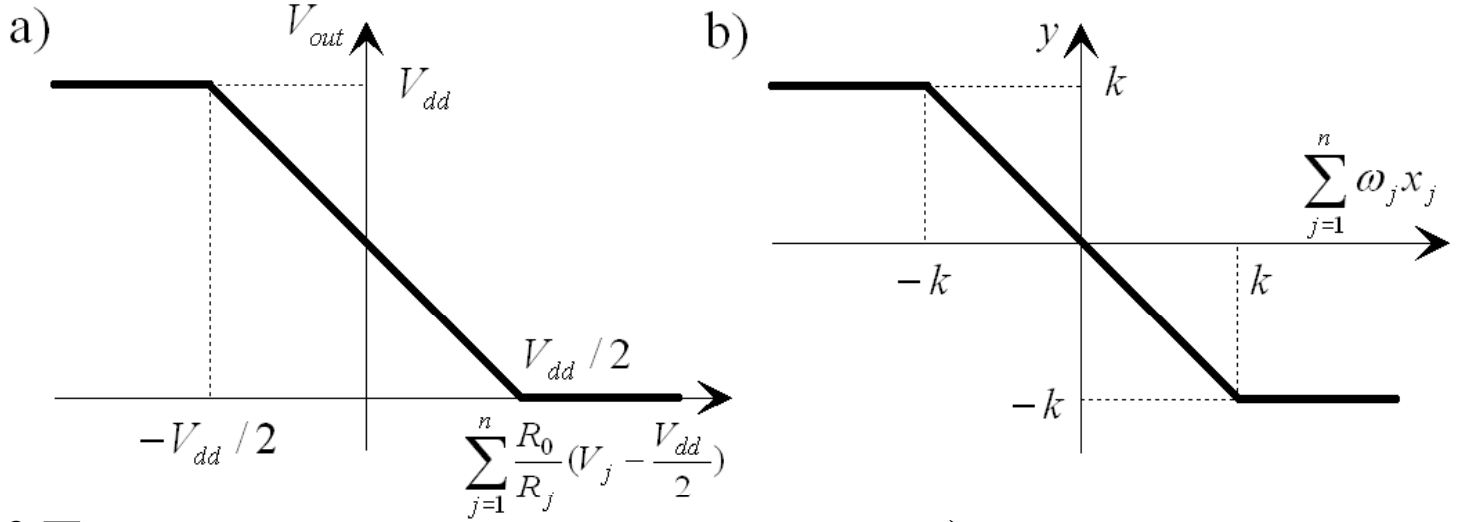


Рис.3 Поведение суммирующего усилителя: a) в координатах напряжений, b) в координатах переменных многозначной логики.

$$y(X)=S(\sum_{j=1}^{n}\omega_j\cdot x_j)=\begin{cases}+k, & \text{если} \quad \sum_{j=1}^{n}\omega_j\cdot x_j\le -k\\ -\sum_{j=1}^{n}\omega_j\cdot x_j, & \text{если } k>\sum_{j=1}^{n}\omega_j\cdot x_j>-k\\ -k, & \text{если} \quad \sum_{j=1}^{n}\omega_j\cdot x_j\ge +k\end{cases} \qquad (2)$$

2. Аппаратная реализация фаззи контроллеров

Далее будем называть функциональный элемент, поведение которого описывается системой (2), многозначным пороговым элементом. В случае, когда $\omega_j = 1$, $j = 1,2,3$, будем называть его мажоритарным элементом и обозначать как $maj(x_1, x_2, x_3)$.

2.2 Функциональная полнота порогового элемента

Основная операция (*или множество основных операций*) *называется функционально полной в логике произвольной значности*, *если произвольная функция этой логики может быть представлена в виде суперпозиции основных операций.*

Существует несколько известных функционально полных множеств базовых функций. Для обеспечения функциональной полноты некоторой новой функции достаточно показать, что каждая функция известного функционально полного множества может быть представлена с помощью этой функции. Одной из функционально полных функций в m-значной логике является функция Вебба:

$$w(x, y) = [\max(x, y) + 1]_{\mathrm{mod}\, m}.$$

2. Аппаратная реализация фаззи контроллеров

Следовательно, достаточно показать, что функция Вебба может быть представлена с помощью новой операции.

Сначала представим функцию $\max(x_1, x_2)$ с помощью пороговых функций. Для этого предварительно рассмотрим следующую функцию

$$f_a(x) = \max(x, a) = \begin{cases} a, & \text{если } a \geq x \\ x, & \text{если } x > a \end{cases}, \quad |x| \leq k, |a| \leq k,$$

график которой показан на рис.4,а.

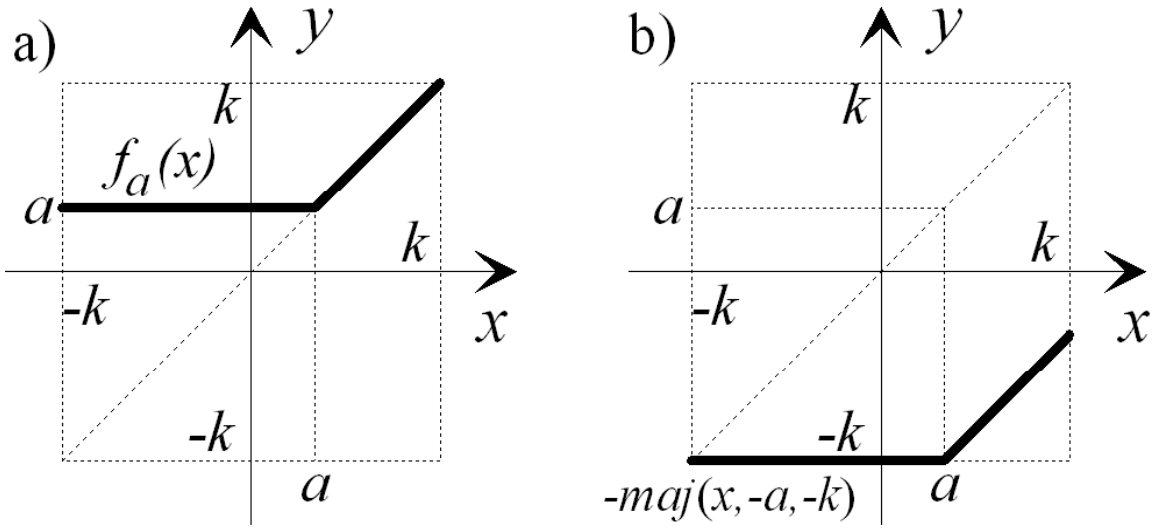


Рис. 4 Графики функций: a) $f_a(x)$ и b) $-maj(x,-a,-k)$.

2. Аппаратная реализация фаззи контроллеров

На рис.4,b показан график функции $-maj(x,-a,-k)$. Так как $x < a$, $x - a - k < -k$ и $-maj(x,-a,-k) = -k$. Для всех значений x, как это следует из рис.4,

$$f_a(x) = -maj(x,-a,-k) + a + k.$$

Следовательно, $f_a(x) = -maj(-maj(x,-a,-k),a,k)$.

Принимая во внимание, что $-maj(a,b,c) = maj(-a,-b,-c)$, заменив x на x_1 и a на x_2, получим

$$\max(x_1,x_2) = maj(maj(x_1,-x_2,-k),-x_2,-k).$$

Теперь рассмотрим, как можно представить функцию $y = (x+1)_{\mathrm{mod}m}$, $x \geq 0$, $0 \leq y \leq m$-1, через пороговые функции.

Введем обозначение $m = 2k+1$ и изменим начало координат таким образом, чтобы функция имела вид $y = (x+k+1)_{\mathrm{mod}(2k+1)} - k$, $x \geq -k$, $-k \leq y \leq +k$.

Реализацию этой функции на пороговых элементах проиллюстрируем последовательностью графиков на рис.5.

2. Аппаратная реализация фаззи контроллеров

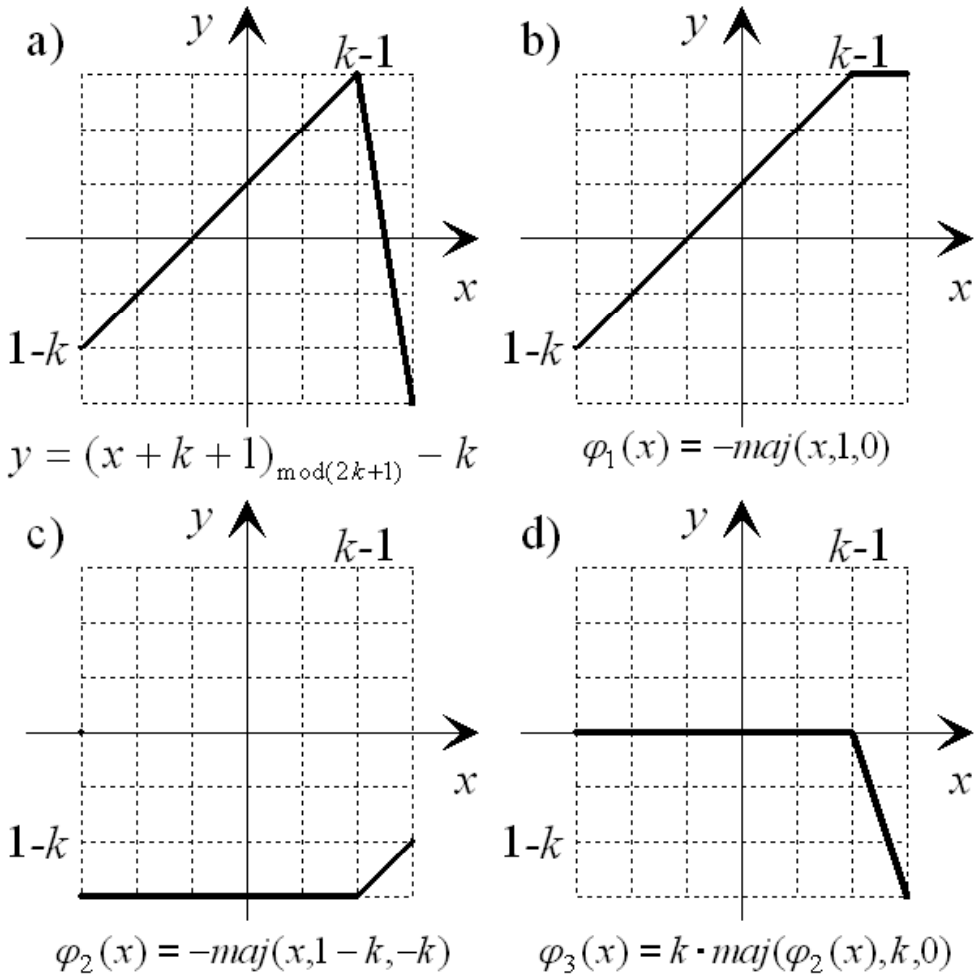


Рис. 5 Реализация функции $y = (x + k + 1)_{\mathrm{mod}(2k+1)} - k$ на пороговых элементах.

2. Аппаратная реализация фаззи контроллеров

Легко видеть, что

$$(x+k+1)_{\mathrm{mod}(2k+1)} - k = \varphi_1(x) + 2\varphi_3(x),$$

и функция может быть реализована на пороговых элементах как

$$y = maj(maj(x,1,0), k \cdot maj(maj(x,1-k,-k),-k,0),$$
$$k \cdot maj(maj(x,1-k,-k),-k,0)).$$

Следовательно, доказана функциональная полнота суммирующего усилителя в произвольно значной логике. Однако процедура доказательства функциональной полноты не дает информации об эффективных методах синтеза.

Далее будут рассмотрены некоторые методы проектирования схем, реализующих функции многозначной логики.

2. Аппаратная реализация фаззи контроллеров

2.3 Представление фаззи устройств в виде аналоговых схем на основе многозначной логики

Стандартная реализация фаззи устройств имеет структуру, представленную на рис.6. Фаззификатор преобразует множество аналоговых входных переменных $X = \{x_1, x_2, ..., x_n\}$ в множество взвешенных лингвистических (цифровых) переменных $A = \{a_1, a_2, ..., a_n\}$.

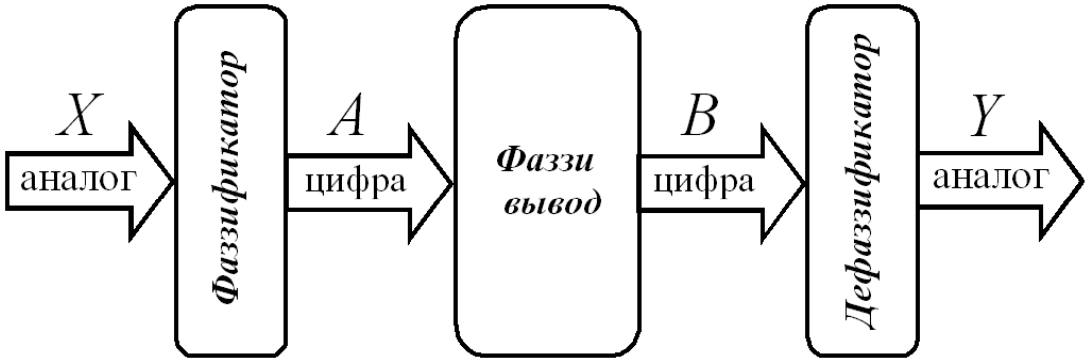


Fig.6 Стандартная структура фаззи устройства.

Блок фаззи вывода на основе фаззи правил вырабатывает множество значений взвешенных лингвистических переменных $B = \{b_1, b_2, ..., b_k\}$.

2. Аппаратная реализация фаззи контроллеров

Дефаззификатор преобразует множество взвешенных лингвистических переменных (цифровых) $B = \{b_1, b_2, ..., b_k\}$ в множество выходных аналоговых переменных $Y = \{y_1, y_2, ..., y_k\}$.

Как правило, фаззификатор и дефаззификатор включают в себя АЦП и ЦАП конверторы и реализуются на аппаратно-программном уровне.

Фаззи вывод обычно реализуется программным путем на специализированном микропроцессоре.

Легко видеть, что фаззи устройство является детерминированным преобразователем аналоговых сигналов, выход которого определяется системой из k n-мерных поверхностей

$$Y(X) = \{y_1(X), y_2(X), ..., y_k(X)\}.$$

В случае достаточно простых функций принадлежности (а их большинство) для реализации фаззи контроллера в виде аналогового устройства достаточно обеспечить кусочно-линейную аппроксимацию между парами точек пространства, вычисленных как смежные значения многозначной логической функции.

2. Аппаратная реализация фаззи контроллеров

Пусть $m = 2k + 1$ лингвистических переменных $a_j \in A$ соответствуют значениям аналоговой переменной $x_i \in X$. Тогда на основе системы фаззи правил можно специфицировать систему *m*-значных логических функций как

$$B(A) = \{b_1(A), b_2(A), ..., b_k(A)\}.$$

Эта система может быть реализована в виде схемы из многозначных пороговых элементов – суммирующих усилителей с насыщением. Многопороговый элемент с характеристикой типа рис.3,б автоматически обеспечивает линейную аппроксимацию можду смежными логическими уровнями, если подавать на его входы аналоговые переменные.

2.4 Примеры реализации фаззи устройств на пороговых элементах

2.4.1. Пример 1

Это контроллер крена оси самоходной тележки. Взят из книги *Fuzzy Control Systems*, Abraham Kandel (Edited by), Lotfi A. Zadeh (Foreword by), CRC Press LLC, Sept. 1993, pp. 81 – 86.

2. Аппаратная реализация фаззи контроллеров

На вход контроллера поступают две аналоговых переменных: *e* и *ce*. Фаззи-фикатор преобразует каждую из них в семь лингвистических переменных: NB – negative big, NM – negative middle, NS – negative small, ZO – zero, PS – positive small, PM – positive middle, PB – positive big. Выход имеет те же семь градаций.

***Таблица* 1**: Таблица фаззи правил.

	e							
ce		NB	NM	NS	ZO	PS	PM	PB
	NB	ZO	PS	PM	PB	PB	PB	PB
	NM	NS	ZO	PS	PM	PB	PB	PB
	NS	NM	NS	ZO	PS	PM	PB	PB
	ZO	NB	NM	NS	ZO	PS	PM	PB
	PS	NB	NB	NM	NS	ZO	PS	PM
	PM	NB	NB	NB	NM	NS	ZO	PS
	PB	NB	NB	NB	NB	NM	NS	ZO

2. Аппаратная реализация фаззи контроллеров

Разобьем напряжение питания устройства равномерно на семь логических уровней, соответствующих лингвистическим уровням, и пронумеруем уровни от -3 to +3. Преобразуем табл.1 в табл.2, задающую семи-значную логическую функцию.

***Таблица* 2**: Семи-значная логическая функция

	e							
ce		-3	-2	-1	0	1	2	3
	-3	0	1	2	3	3	3	3
	-2	-1	0	1	2	3	3	3
	-1	-2	-1	0	1	2	3	3
	0	-3	-2	-1	0	1	2	3
	1	-3	-3	-2	-1	0	1	2
	2	-3	-3	-3	-2	-1	0	1
	3	-3	-3	-3	-3	-2	-1	0

Из таблицы видно, что эта функция симметрична относительно побочной диагонали, и ее значения могут быть посчитаны как e – ce. Эта зависимость показана на рис.7.

2. Аппаратная реализация фаззи контроллеров

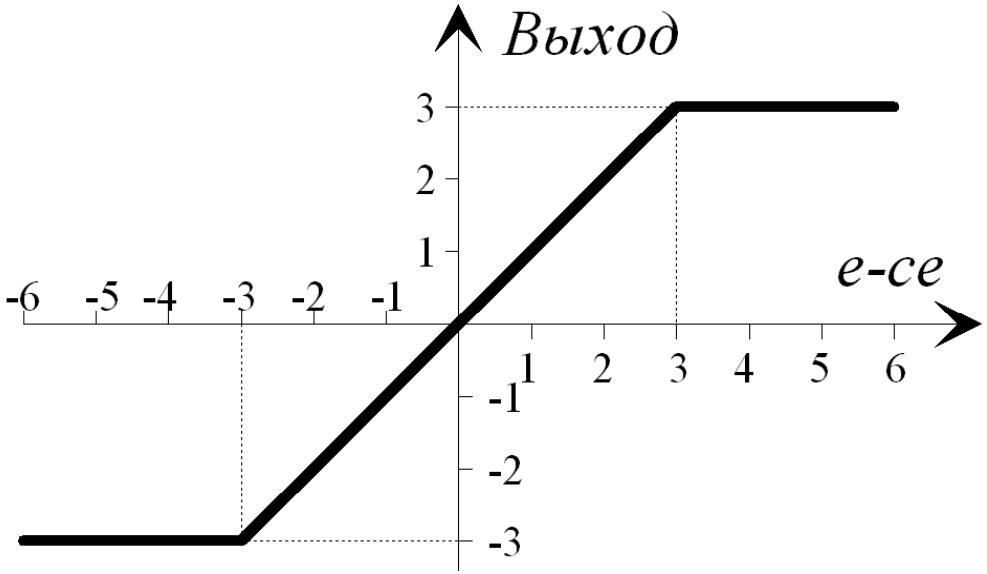


Рис.7 График функции, представленной в табл.2.

Сравнивая рис.3,b и рис.7, становится очевидным, что эта функция может быть реализована с помощью двух суммирующих усилителей, причем один из них реализует функцию дополнения, или инвертирования переменной.

Инвертирование логической переменной, принимающей значения из интервала $-k \div +k$, есть операция диаметрального отрицания $\bar{x} = -x$, или в терминах напряжений $\overline{V_{out}} = V_{dd} - V_{in}$. Схема контроллера приведена на рис.8.

2. Аппаратная реализация фаззи контроллеров

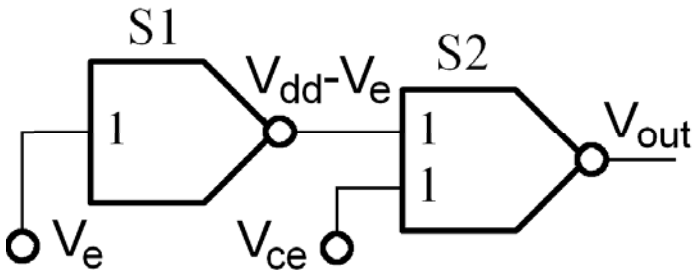


Рис.8 Реализация контроллера на суммирующих усилителях.

2.4.2 *Example 2. Фаззи контроллер для стиральной машины*

Этот пример взят из Aptronix Incorporated (http://www.aptronix.com/fuzzynet).

A. Спецификация контроллера
Входные переменные:
Загрязнение одежды: *Большое* (*Б*), *Среднее* (*С*), *Маленькое* (*S*);
Тип загрязнения: *Засаленное* (*З*), *Среднее* (*С*), *Незасаленное* (*НЗ*).
Выходная переменная – *время стирки* (в минутах): *Очень большое* (*ОБ*), *Большое* (*Б*), *Среднее* (*С*), *Короткое* (*К*), Очень короткое (*ОК*).
Фаззи правила приведены в табл.3.

2. Аппаратная реализация фаззи контроллеров

Таблица 3: Матрица лингвистических переменных

Время стирки		*Загрязнение одежды*		
		М	*С*	*Б*
Тип загряз-нения	*НЗ*	*ОК*	*К*	*С*
	С	*С*	*С*	*Б*
	З	*Б*	*Б*	*ОБ*

Табл.3 преобразуем в таблицу переменных многозначной логики (табл.4). can

Таблица 4: Матрица многозначных переменных

T		*Y*		
		-2	0	+2
X	-2	-2	-1	0
	0	0	0	+1
	+2	+1	+1	+2

Выходная переменная «*Время стирки*» (*T*) имеет 5 логических уровней, а входные переменные, обозначенные как *X* и *Y*, имеют по три уровня. Поскольку диапазон представления этих переменных одинаков в терминах напряжений, логические уровни входов *X* и *Y* имеют по три градации: -2, 0, +2.

2. Аппаратная реализация фаззи контроллеров

B. Декомпозиция многозначной функции

Представим матрицу времени стирки T в виде суммы двух матриц:

$$\underset{T}{\begin{vmatrix} -2 & -1 & 0 \\ 0 & 0 & +1 \\ +1 & +1 & +2 \end{vmatrix}} = \underset{\varphi_1}{\begin{vmatrix} -1 & -1 & 0 \\ 0 & 0 & +1 \\ +1 & +1 & +2 \end{vmatrix}} + \underset{\varphi_2}{\begin{vmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}} \qquad (3)$$

или $T = S(-\varphi_1 - \varphi_2)$, где S – функция суммирующего усилителя с насыщением.

Рассмотрим функцию одной переменной $\varphi_3(Y) = S_1(0{,}5 \cdot Y - 2) = \left| +2 \;\; +2 \;\; +1 \right|$. В этой функции переменная Y (загрязнение одежды) представлена вектором-строкой $Y = \left| -2 \quad 0 \quad +2 \right|$.

Построим следующее промежуточное выражение выражение:

$$\varphi_3(Y) - 0{,}5 \cdot X - 2 = \begin{vmatrix} +1 & +1 & 0 \\ 0 & 0 & -1 \\ -1 & -1 & -2 \end{vmatrix} = -\varphi_1, \qquad (4)$$

в котором переменная X (тип загрязнения) представлена вектором-столбцом:

2. Аппаратная реализация фаззи контроллеров

$$X = \begin{vmatrix} -2 \\ 0 \\ +2 \end{vmatrix}.$$

Теперь введем функцию

$$\varphi_4(X,Y) = S_2(X+Y+4) = \begin{vmatrix} 0 & -2 & -2 \\ -2 & -2 & -2 \\ -2 & -2 & -2 \end{vmatrix}$$

и сформируем вторую вспомогательную функцию

$$0{,}5 \cdot \varphi_4(X,Y) + 1 = \begin{vmatrix} +1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} = -\varphi_2. \qquad (5)$$

Окончательно из (3), (4) и (5) получим

$$T = S[\varphi_3(Y) - 0{,}5 \cdot X - 1 + 0{,}5 \cdot \varphi_4(X,Y)] =$$
$$= S_4[S_1(0{,}5Y - 2) + 0{,}5 \cdot S_2(X+Y+4) + 0{,}5 \cdot S_3(X) - 1].$$

На рис.9 показана реализующая эту функцию схема, построенная из четырех многозначных пороговых элементов. Результат SPICE моделирования представлен на рис.10 в виде поверхности отклика.

2. Аппаратная реализация фаззи контроллеров

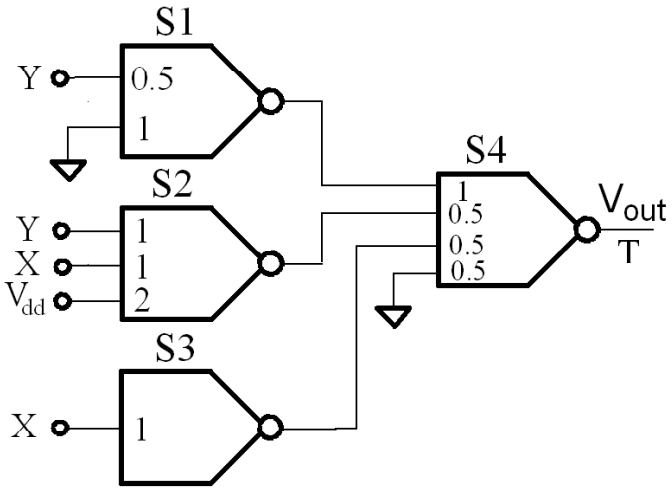


Рис.9 Схема контроллера для стиральной машины.

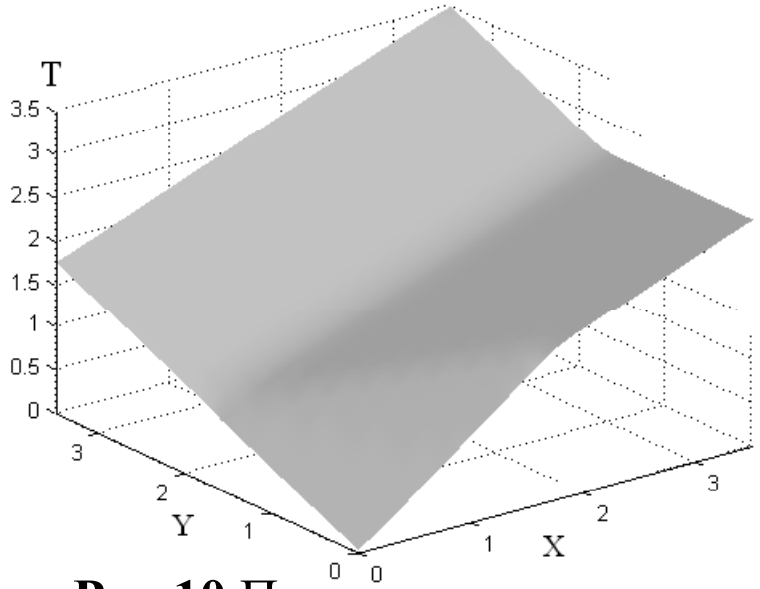


Рис.10 Поверхность отклика контроллера.

На рис.10 все переменные представлены напряжениями. Соответствие напряжений логическим значениям приведено в табл.5. Легко видеть линейную аппроксимацию между смежными логическими значениями.

Таблица 5: Соответствие напряжений логическим значениям.

-2	-1	0	1	2
0V	0.875V	1.75V	2.625V	3.5V

Применение многозначной логики при проектировании фаззи-контроллеров

Лекция 10

3. Универсальный метод реализации правил фаззи вывода

Поскольку суммирующий усилитель с насыщением является функционально полным в произвольно-значной логике, он может быть использован в качестве базового элемента для построения фаззи устройств.

Рассмотрим технологию проектирования аналоговых схем, реализующих функции многозначной логики для фаззи контроллеров.

Будем рассматривать функции логики нечетной значности, т.е. $m = 2k + 1$. Пусть $X = \{x_1, x_2, \cdots, x_n\}, (-k \le x_j \le +k),$ – множество входных m-значных переменных и $y = F(X)$ – выходная переменная. Тогда для функции m-значной логики можно построить аналог декомпозиции Шеннона для функций бинарной логики:

$$y_i = F_i(X) = \bigcup_{\alpha=-k}^{+k} [\text{if } x_j = \alpha \text{ then } y = F(x_j = \alpha, X \setminus x_j)]. \qquad (6)$$

Это выражение позволяет исключить одну переменную. Его можно расширить таким образом, чтобы иметь возможность исключения подмножества переменных путем введения правил типа:

$$\text{if} \quad Z = A \quad \text{then} \quad y = F(Z = A, X \setminus Z), \qquad (7)$$

3. Универсальный метод реализации правил фаззи вывода

где Z – подмножество переменных ($Z \subset X$), а A – набор значений переменных множества Z.

Если построить базовую схему, реализующую правило (7), то с ее помощью можно построить фаззи устройство непосредственно по системе фаззи правил. Однако заметим, что (6) и (7) представляют многозначные функции в виде кусочно-постоянных зависимостей. На рис.11,а приведен пример такой 7-значной кусочно-постоянной функции.

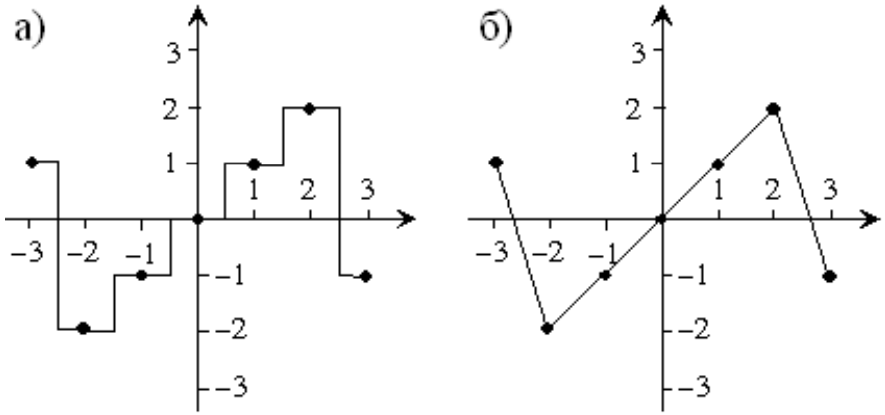


Рис.11 Пример 7-значной функции: а) кусочно-постоянное представление; б) кусочно-линейное представление.

3. Универсальный метод реализации правил фаззи вывода

Принимая во внимание процедуры фаззификации и дефаззификации в фаззи логике, соответствующие многозначные логические функции должны иметь, по крайней мере, кусочно-линейную аппроксимацию между смежными логическими уровнями. На рис.11,б показано такое представление функции с равномерно распределенными логическими уровнями в диапазоне напряжений входной и выходной переменной.

3.1 Маскирование входов суммирующих усилителей

Перепишем систему уравнений, определяющую поведение суммирующего усилителя с насыщением, в следующем виде:

$$S(A\cdot X+\beta)=\begin{cases}+k, & \text{если} \quad \sum_{i=1}^{n}\alpha_i\cdot x_i+\beta\le -k\\ -\sum_{i=1}^{n}\alpha_i\cdot x_i-\beta, & \text{если} \quad -k<\sum_{i=1}^{n}\alpha_i\cdot x_i+\beta<k\\ -k, & \text{если} \quad +k\le\sum_{i=1}^{n}\alpha_i\cdot x_i+\beta\end{cases} \qquad (8)$$

3. Универсальный метод реализации правил фаззи вывода

Здесь $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ – множество весовых коэффициентов, $X = \{x_1, x_2, ..., x_n\}$ – множество аналоговых или многозначных переменных, β – константа, символизирующая порог, и $\pm k$ – уровни насыщения (в m-значной логике $m = 2k + 1$).

Введем функцию-маску $M_\alpha(x)$:

$$M_\alpha(x) = \begin{cases} +k, & \text{если } x \le \alpha - 1 \\ k(\alpha - x), & \text{если } \alpha - 1 < x < \alpha + 1 \\ -k, & \text{если } \alpha + 1 \le x \end{cases}, \text{ где} \qquad (9)$$

α $(-k \le \alpha \le k)$ – фиксированное значение переменной x. При $x = \alpha$ $M_\alpha(\alpha) = 0$, при $x = \alpha + 1$ $M_\alpha(\alpha + 1) = -k$, а при $x = \alpha - 1$ $M_\alpha(\alpha - 1) = +k$.

На рис.12 приведен пример функции-маски $M_{-1}(x) = 0$ для $m = 7$.

Принимая во внимание, что напряжение питания V_{dd} имеет логическое значение, равное $+k$, а потенциал земли V_{gnd} – логическое значение $-k$, маск-функция может быть реализована на суммирующем усилителе как

3. Универсальный метод реализации правил фаззи вывода

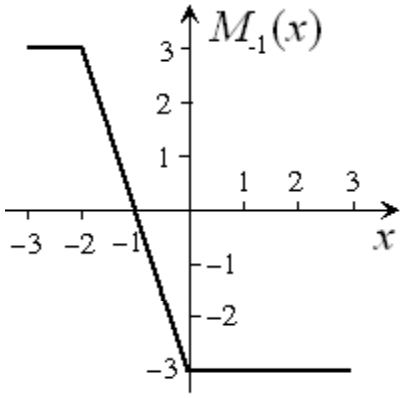


Рис.12 График $M_{-1}(x)$ для $m = 7$.

$$M_{\alpha}(x) = \begin{cases} S(kx - \alpha \cdot V_{dd}), & \text{если } \alpha < 0 \\ S(kx), & \text{если } \alpha = 0 \\ S(kx + \alpha \cdot V_{gnd}), & \text{если } \alpha > 0 \end{cases}, \qquad (10)$$

где x ($V_{gnd} \le x \le V_{dd}$) измеряется в вольтах.

При использовании маск-функции можно реализовать правило типа

$$\text{if } x = \alpha \text{ then } y = F(x = \alpha, Y) \text{ else } y = 0 \qquad (11)$$

с помощью схемы, представленной на рис.13, которая выделяет значение функции $F(x = \alpha, Y)$ в точке $x = \alpha$.

3. Универсальный метод реализации правил фаззи вывода

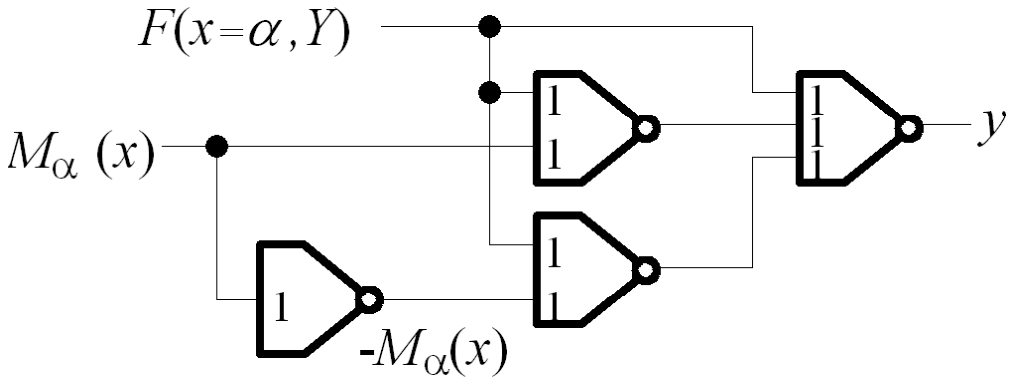


Рис.13 Схема, реализующая правило (11).

Эту схему можно записать в аналитической форме:

$$y = S\{S[M_\alpha(x) + F(x=\alpha, Y)] + S[S(M_\alpha(x)) + F(x=\alpha, Y)] + F(x=\alpha, Y)\}.$$

Например, в случае $\alpha = -1$, $F(x=-1, Y) = 2$ и $m = 7$ поведение схемы на рис.13 может быть пояснено последовательностью графиков на рис.14.

Анализируя реализацию правила (11) легко видеть, что в ней условие $x = \alpha$ заменено эквивалентным условием равенства нулю маск-функции $M_\alpha(x) = 0$.

3. Универсальный метод реализации правил фаззи вывода

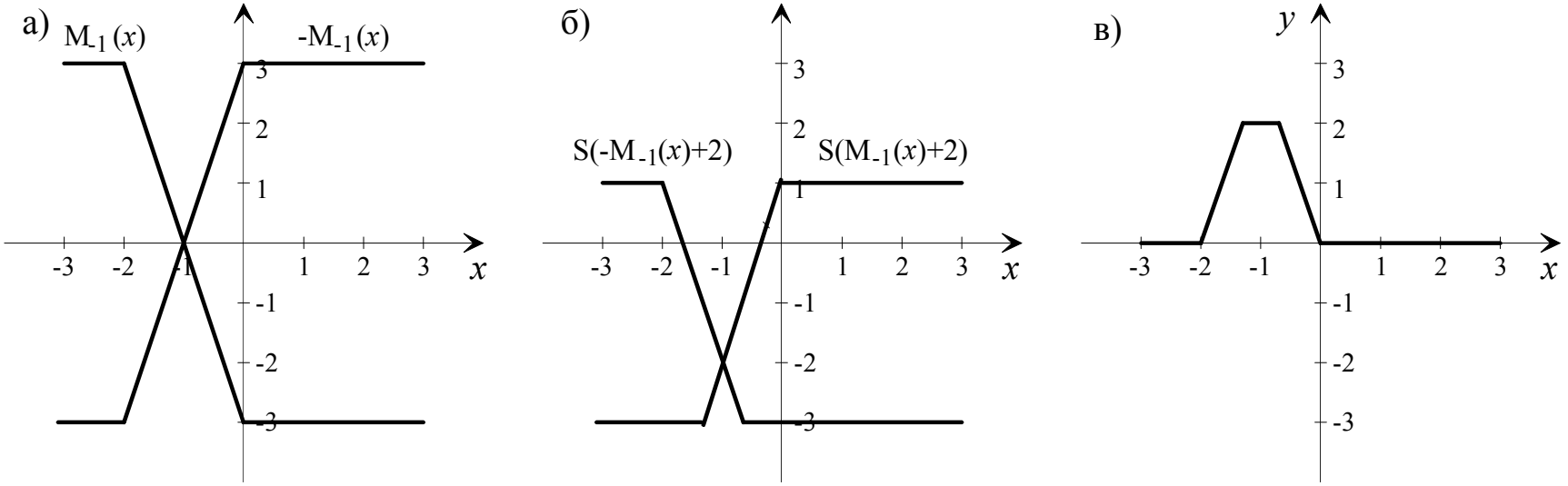


Рис.14 Последовательность графиков, поясняющих работу схемы рис.13.

3.2 Маск-функции других типов

При применении разложения (6) с целью сокращения числа переменных, от которых зависят составляющие декомпозиции *m*-значной логической функции, необходимо выполнить *m* правил типа (11) для выделения одной переменной *x* и найти *m* составляющих декомпозиции

$$F_\alpha(x=\alpha, Y),\ -k \le \alpha \le +k.$$

3. Универсальный метод реализации правил фаззи вывода

Если функция $F(x,Y)$ не изменяет своего значения на некотором интервале изменения выделяемой переменной x, то число этих правил может быть сокращено. Условная часть единственного правила, соответствующего такому интервалу, зависит от места расположения интервала в диапазоне изменения переменной x *и* имеет одну из трех форм: $\alpha \le x \le \beta$, $x \le \beta$, $\alpha \le x$, $-k \le \alpha < \beta \le +k$.

Для условия $\alpha \le x \le \beta$, построим следующую маск-функцию:

$$M_{\alpha,\beta}(x) = \begin{cases} +k, & \text{если } x \ge \beta + 1 \\ -M_{\alpha-1}(x) - M_{\beta+1}(x), & \text{если } \alpha - 1 < x < \beta + 1. \\ -k, & \text{если } x \le \alpha - 1 \end{cases} \qquad (12)$$

В случаях, когда $\alpha = -k$ или $\beta = +k$, эта функция приобретает вид:

$$M_{-k,\beta}(x) = \begin{cases} +k, & \text{если } x \ge \beta + 1 \\ +k - M_{\beta+1}(x), & \text{если } x < \beta + 1 \end{cases} \quad \text{или} \qquad (13)$$

$$M_{\alpha,+k}(x) = \begin{cases} -\mathrm{M}_{\alpha\text{-}1}(x) - k, & \text{если } \alpha - 1 < x \\ -k, & \text{если } x \le \alpha - 1 \end{cases}. \qquad (14)$$

Легко видеть, что на указанных интервалах маск-функци принимают логическое значение 0.

3. Универсальный метод реализации правил фаззи вывода

Реализации маск-функций (12), (13) и (14) на суммирующих усилителях:

$$M_{\alpha,\beta}(x) = S[M_{\alpha-1}(x) + M_{\beta+1}(x)], \qquad (15)$$

$$M_{-k,\beta}(x) = S[V_{gnd} + M_{\beta+1}(x)], \qquad (16)$$

$$M_{\alpha,+k}(x) = S[M_{\alpha-1}(x) + V_{dd}]. \qquad (17)$$

Покажем как интервальное маскирование может быть применено при реализации правил вида

$$\text{if } \alpha \le x \le \beta \text{ then } y = F(\alpha \le x \le \beta, Y) = \Phi(Y) \text{ else } y = 0.$$

В этом правиле условие заменяется маск-функцией, равной нулю,

$$\text{if } M_{\alpha,\beta}(x) = 0 \text{ then } y = F(\alpha \le x \le \beta, Y) = \Phi(Y) \text{ else } y = 0$$

и схема, реализующая это правило, строится в соответствии со схемой рис.13.

Аналитическая запись полученной схемы имеет вид:

$$y = S\{S[M_{\alpha,\beta}(x) + \Phi(Y)] + S[S(M_{\alpha,\beta}(x)) + \Phi(Y)] + \Phi(Y)\}. \qquad (18)$$

Последовательность рисунков на рис.15 иллюстрирует реализацию правила «$\text{if } -2 \le x \le +1 \text{ then } y = -2 \text{ else } y = 0$» для случая 9-значной логики (m =9).

3. Универсальный метод реализации правил фаззи вывода

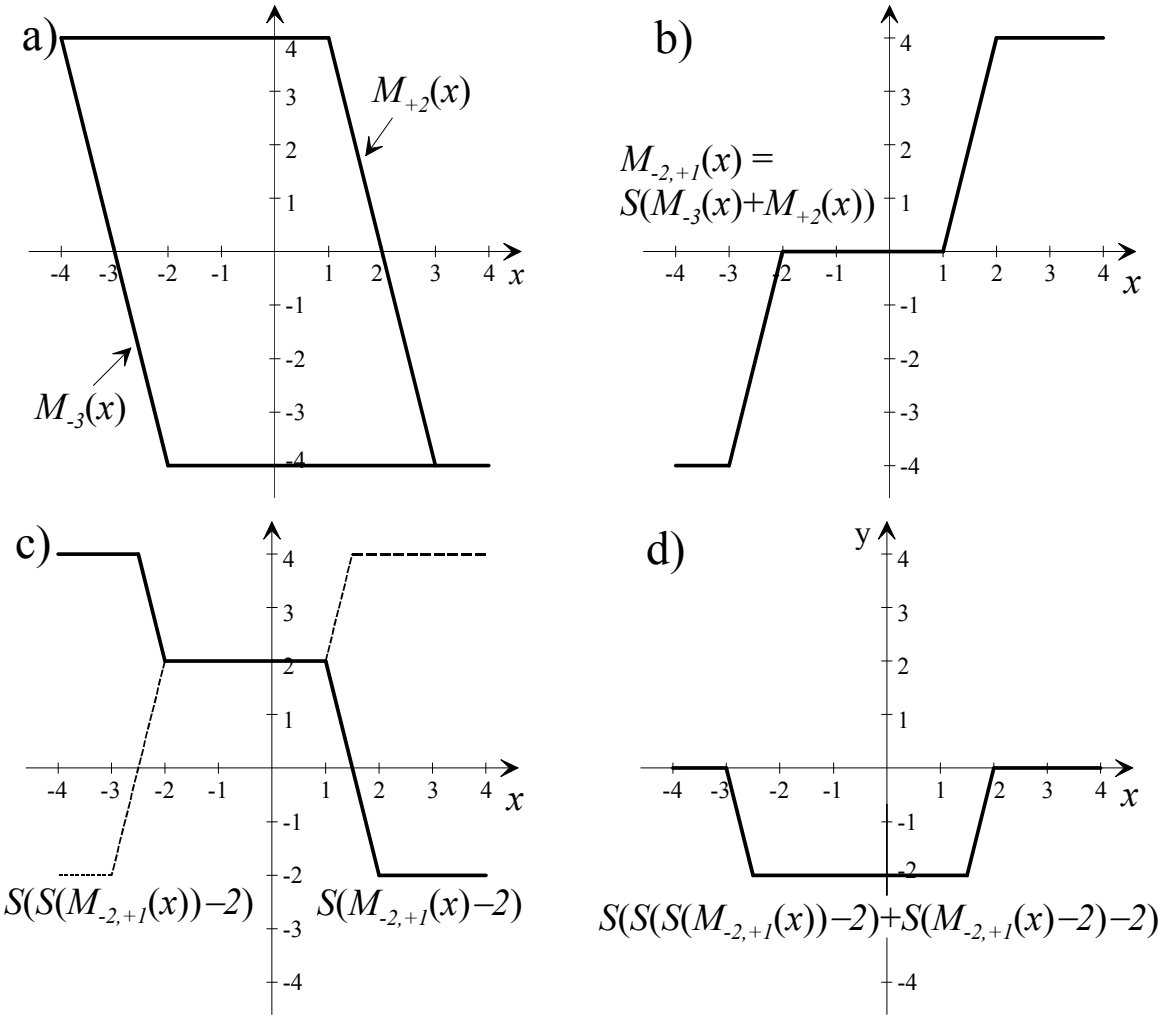


Рис.15 Пример реализации правила с использованием маск-функции.

3. Универсальный метод реализации правил фаззи вывода

3.3 Пример применения интервального маскирования

Рассмотрим пример проектирования фаззи контроллера, управляющего силой шагового двигателя робота (спецификация взята из [4, стр. 123-128]).

Фаззи контроллер реализует функцию двух аналоговых переменных: *ошибка положения* and *ошибка силы*, обозначенных соответственно как *x* и *y*. Каждая из этих переменных представлена 7-ю лингвистическими переменными: NL, NM, NS, ZE, PS, PM, PL, функции принадлежности которых показаны на рис.16.

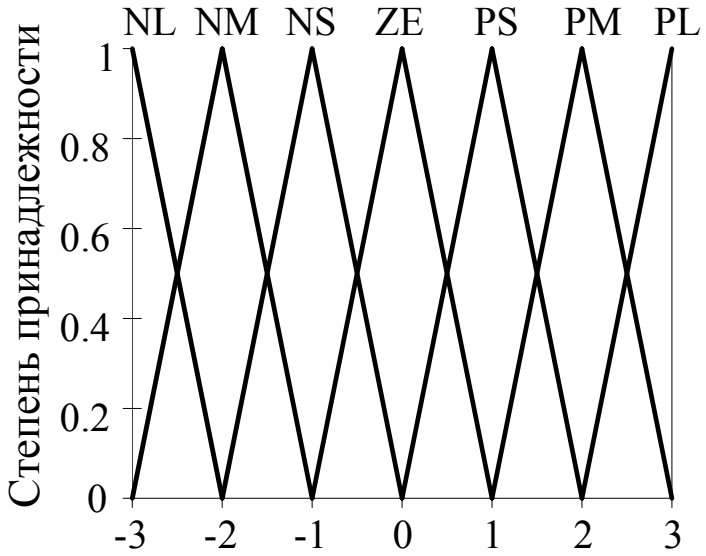


Рис.16 Функции принадлежности лингвистических переменных для *x* и *y*.

3. Универсальный метод реализации правил фаззи вывода

Матрица фаззи правил для выходных лингвистических переменных представлена в табл.6.

Таблица 6: Матрица правил для фаззи вывода.

		Ошибка положения (x)						
		NL	NM	NS	ZE	PS	PM	PL
Ошибка силы (y)	NL	NM	NL	NL	NL	NL	NL	NM
	NM	NS	NM	NM	NM	NM	NM	NS
	NS	ZE	NS	NS	NS	NS	NS	ZE
	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
	PS	ZE	PS	PS	PS	PS	PS	ZE
	PM	PS	PM	PM	PM	PM	PM	PS
	PL	PM	PL	PL	PL	PL	PL	PM

Преобразуем табл.6 в табл.7, специфицирующую 7-значную логическую функцию.

Анализируя табл.7, легко заметить, что она составлена из двух типов колонок, каждая из которых представляет функцию одной переменной y (см. табл.8).

3. Универсальный метод реализации правил фаззи вывода

Таблица 7: Матрица 7-значной логиеской функции.

		x						
		-3	-2	-1	0	1	2	3
y	-3	-2	-3	-3	-3	-3	-3	-2
	-2	-1	-2	-2	-2	-2	-2	-1
	-1	0	-1	-1	-1	-1	-1	0
	0	0	0	0	0	0	0	0
	1	0	1	1	1	1	1	0
	2	1	2	2	2	2	2	1
	3	2	3	3	3	3	3	2

Таблица 8: Две функции переменной y

	y						
	-3	-2	-1	0	1	2	3
$F_1(y)$	-2	-1	0	0	0	1	2
$F_2(y)$	-3	-2	-1	0	1	2	3

На рис.17 изображены графики этих функций. Легко видеть, что функция $F_1(y)$ выглядит как маск-функция $M_{-1,1}(y)$, но имеет другой наклон лучей.

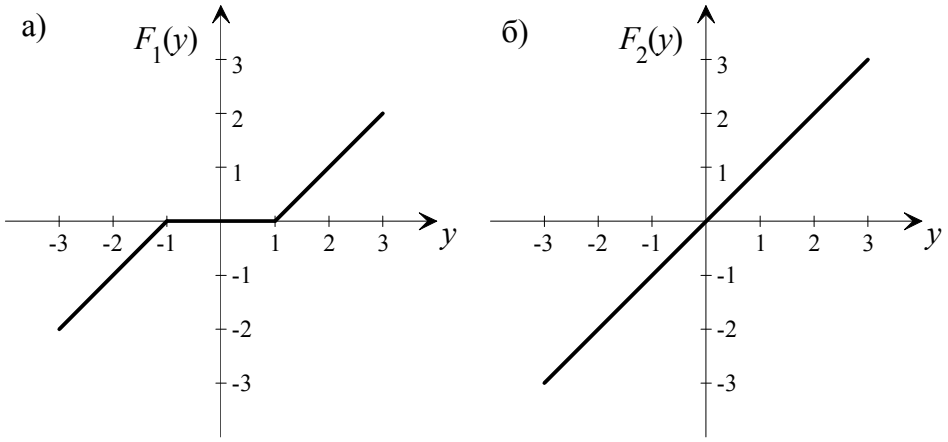


Рис.17 Графики функций, заданных табл.8.

3. Универсальный метод реализации правил фаззи вывода

По аналогии с (9), (10), (12), (24) и рис.15,a и рис.15,b можно построить функцию $F_1(y)$ в соответствии с графиком на рис.18.

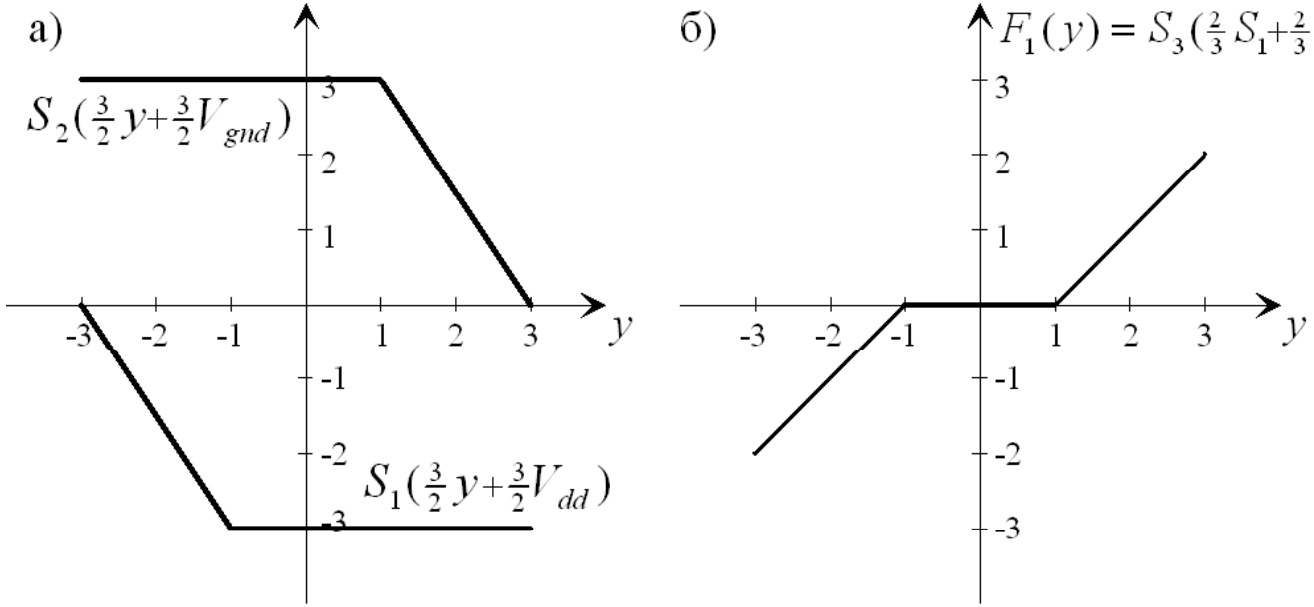


Рис.18 Построение функции $F_1(y)$.

В результате функции $F_1(y)$ и $F_2(y)$ могут быть реализованы как

$$F_1(y)=S_3[\tfrac{2}{3}S_2(\tfrac{3}{2}y+\tfrac{3}{2}V_{gnd});\tfrac{2}{3}S_1(\tfrac{3}{2}y+\tfrac{3}{2}V_{dd})],\ F_2(y)=y. \quad (19)$$

3. Универсальный метод реализации правил фаззи вывода

В соответствии с табл.8 поведение контроллера может быть записано в разложении относительно переменной x с помощью единственного правила:

$$\text{if } -2 \le x \le +2 \text{ then } Выход = F_2(y) \text{ else } Выход = F_1(y).$$

Для реализации этого правила с помощью суммирующих усилителей представим его в виде двух правил:

1) if $M_{-2,+2}(x) = 0$ then $Выход = F_2(y)$ else $Выход = 0$.

2) if $M_{-2,+2}(x) \neq 0$ then $Выход = 0$ else $Выход = F_1(y)$.

Первое правило может быть реализовано в соответствии с (15), (18) и (19) как

$$\begin{cases} M_{-2,+2}(x) = S_4(M_{-3}(x) + M_{+3}(x)), \\ -M_{-2,+2}(x) = S_5(M_{-2,+2}(x)), \\ \Phi_1 = S_{11}\{S_6[M_{-2,+2}(x) + F_2(y)] + S_7[-M_{-2,+2}(x) + F_2(y)] + F_2(y)\}. \end{cases}$$

Второе правило может быть реализовано в соответствии со схемой рис.13, на входы которой подаются $M_{-2,+2}(x)$ и $F_1(y)$, а вход $F_1(y)$ выходного усилителя имеет вес, равный 2:

3. Универсальный метод реализации правил фаззи вывода

$$\Phi_2 = -S_{10}\{S_9[M_{-2,+2}(x) + F_1(y)] + S_8[-M_{-2,+2}(x) + F_1(y)] + 2F_1(y)\}.$$

Окончательно, выходной сигнал контроллера вычисляется как

$$Выход = \Phi_1 + \Phi_2 = S_{11}(-\Phi_1 - \Phi_2).$$

На рис.19 представлена структурная схема реализации контроллера, на элементах которой указаны веса входов.

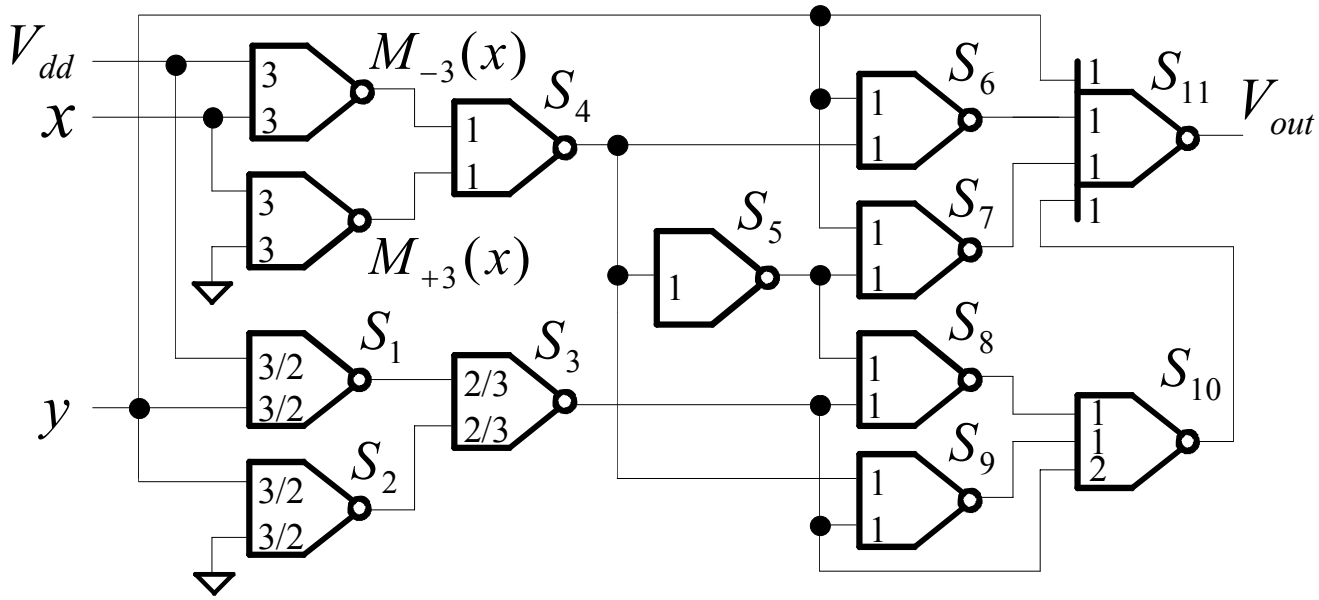


Рис.19 Структурная схема контроллера.

3. Универсальный метод реализации правил фаззи вывода

Результаты моделирования контроллера на рис.19 приведены на рис 20. Параметры моделирования: V_{dd} =3.5V, x изменяется линейно от 0V до 3.5V, y изменяется дискретно с шагом, определяющим логические уровни.

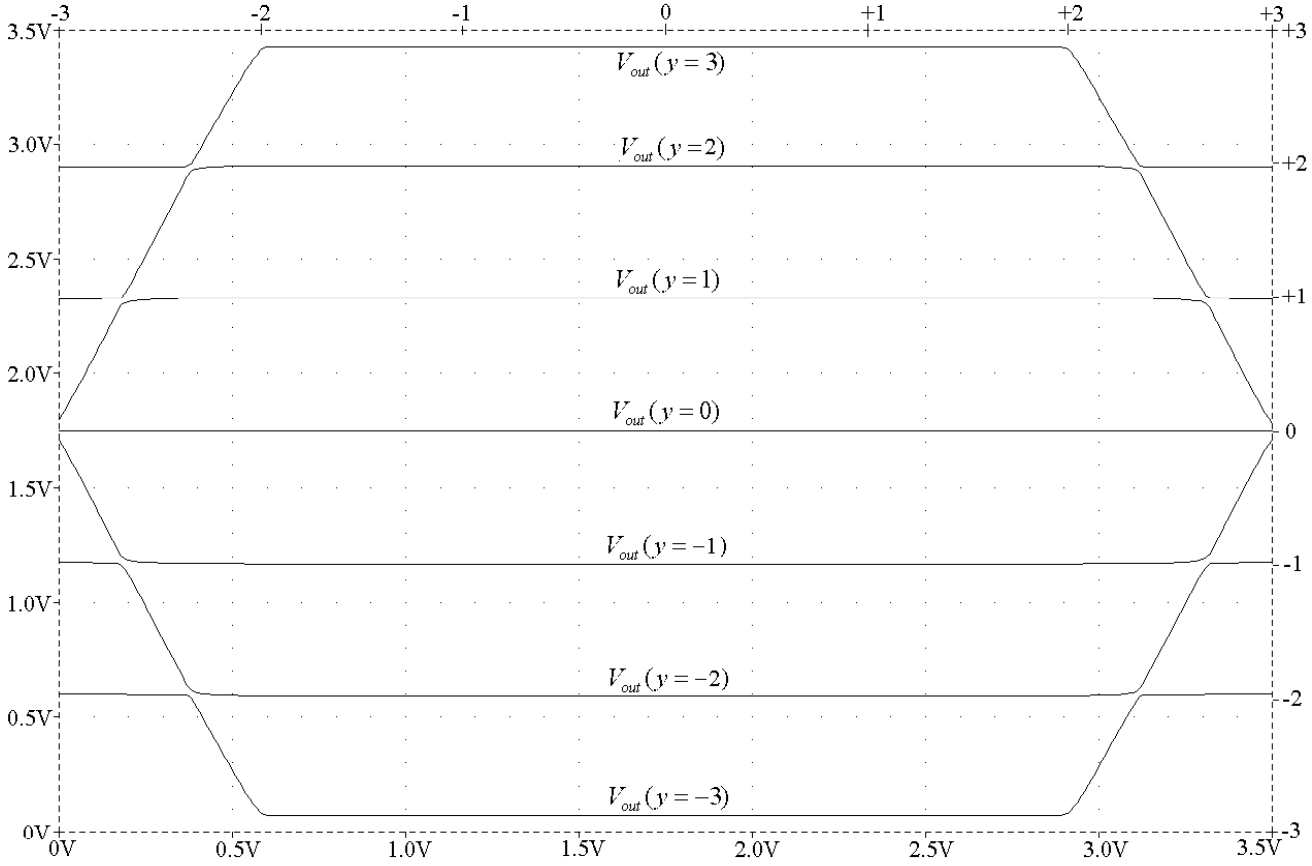


Рис.20 Результаты SPICE моделирование контроллера.

4. Частные методы реализации правил фаззи вывода

Универсальный метод реализации многозначных логических функций пригоден для реализации произвольных функций, но часто приводит к неоправданно сложным решениям в силу своей универсальности. По этой причине был предложен набор частных методов, использующих некоторые свойства матриц, задающих многозначные функции.

Итак, исходное множество фаззи правил представляется в форма матрицы или набора матриц, определяющих многозначные логические функции. Как правило, эти матрицы не могут быть реализованы непосредственно. Их необходимо декомпозировать на матрицы с относительно простой конфигурацией расположения элементов, для которых могут быть найдены простые реализации.

Топология значимых элементов внутри компонентных матриц может быть специфирована, например, как: 1) значимые элементы расположены симметрично относительно одной из главных диагоналей, 2) элементы с различными значениями могут быть разделены прямой линией, 3) элементы с одинаковыми значениями образуют прямоугольник, 4) лежат на одной линии, 5) значимый элемент является единственным в матрице и т.д.

4. Частные методы реализации правил фаззи вывода

Лучший способ проиллюстрировать применение частных методов – это показать возможные декомпозиции матриц на множество реализуемых матриц на реальном примере проектирования.

Возьмем описание достаточно сложного контроллера из одного из патентов корпорации Toyota Motors. Контроллер предназначен для управления двигателем и вычисляет коэффициент R управления временем регенерации на основе коэффициента K_p разности давлений и потребления топлива Q_f. Множество фаззи правил представлено в таблице 9.

Таблица 9: Множество фаззи правил для времени регенерации.

R		K_p						
		NB	NM	NS	ZO	PS	PM	PB
Q_f	NB	NB	NB	NM	NS	ZO	PB	PB
	NM	NB	NM	NS	NS	ZO	PB	PB
	NS	NM	NM	NS	ZO	ZO	PS	PB
	ZO	NM	NS	ZO	ZO	PS	PM	PB
	PS	NS	ZO	ZO	PS	PS	PM	PB
	PM	ZO	ZO	PS	PM	PM	PM	PB
	PB	PS	PS	PM	PM	PM	PB	PB

4. Частные методы реализации правил фаззи вывода

Анализ приведенных в патенте функций принадлежности лингвистических переменных, представляющих входные и выходные аналоговые сигналы, показывает, что лингвистические переменные с максимальным весом равномерно распределены в пределах диапазонов изменений соответствующих аналоговых сигналов. Следовательно, эти лингвистические переменные могут быть заменены логическими значениями, как это показано в табл.10.

Таблица 10: 7-значная логическая функция

x\y	-3	-2	-1	0	1	2	3
-3	-3	-3	-1	-1	0	3	3
-2	-3	-2	-1	-1	0	3	3
-1	-2	-2	-1	0	0	1	3
0	-2	-1	0	0	1	2	3
1	-1	0	0	1	1	2	3
2	0	0	1	2	2	2	3
3	1	1	2	2	2	3	3

В этой таблице лингвистические переменные для сигналов K_p и Q_f заменены логическими значениями переменных x и y.

4. Частные методы реализации правил фаззи вывода

4.1 Выделение симметричной компонентной матрицы

Представим табл.10 в виде матрицы M, которую, в свою очередь, представим в виде суммы двух компонентных матриц: $M = M_1 + M_2$.

$$
\underset{M}{\begin{vmatrix}
-3 & -3 & -2 & -1 & \pm 0 & +3 & +3 \\
-3 & -2 & -1 & -1 & \pm 0 & +3 & +3 \\
-2 & -2 & -1 & \pm 0 & \pm 0 & +1 & +3 \\
-2 & -1 & \pm 0 & \pm 0 & +1 & +2 & +3 \\
-1 & \pm 0 & \pm 0 & +1 & +1 & +2 & +3 \\
\pm 0 & \pm 0 & +1 & +2 & +2 & +2 & +3 \\
+1 & +1 & +2 & +2 & +2 & +3 & +3
\end{vmatrix}}
=
\underset{M_1}{\begin{vmatrix}
-3 & -3 & -2 & -2 & -1 & \pm 0 & \pm 0 \\
-3 & -2 & -2 & -1 & \pm 0 & \pm 0 & +1 \\
-2 & -2 & -1 & \pm 0 & \pm 0 & +1 & +1 \\
-2 & -1 & \pm 0 & \pm 0 & +1 & +1 & +2 \\
-1 & \pm 0 & \pm 0 & +1 & +1 & +2 & +2 \\
\pm 0 & \pm 0 & +1 & +1 & +2 & +2 & +3 \\
\pm 0 & +1 & +1 & +2 & +2 & +3 & +3
\end{vmatrix}}
+
\underset{M_2}{\begin{vmatrix}
\leq 0 & \leq 0 & \pm 0 & +1 & +1 & +3 & +3 \\
\leq 0 & \pm 0 & +1 & \pm 0 & \pm 0 & +3 & \geq 2 \\
\pm 0 & \pm 0 & \pm 0 & \pm 0 & \pm 0 & \pm 0 & \geq 2 \\
\pm 0 & \pm 0 & \pm 0 & \pm 0 & \pm 0 & +1 & \geq 1 \\
\pm 0 & \pm 0 & \pm 0 & \pm 0 & \pm 0 & \pm 0 & \geq 1 \\
\pm 0 & \pm 0 & \pm 0 & +1 & \pm 0 & \pm 0 & \geq 0 \\
+1 & \pm 0 & +1 & \pm 0 & \pm 0 & \geq 0 & \geq 0
\end{vmatrix}}
$$

Матрица M_1 симметрична относительно главной побочной диагонали, а матрица M_2 является остаточной, подлежащей дальнейшей декомпозиции.

Легко видеть, что матрица M_1 может быть представлена как функция одной переменной z: $f_1(z) = (x + y)/2$.

4. Частные методы реализации правил фаззи вывода

После выполнения линейной аппроксимации между соседними логическими уровнями эта функция будет иметь форму, представленную на рис.21.

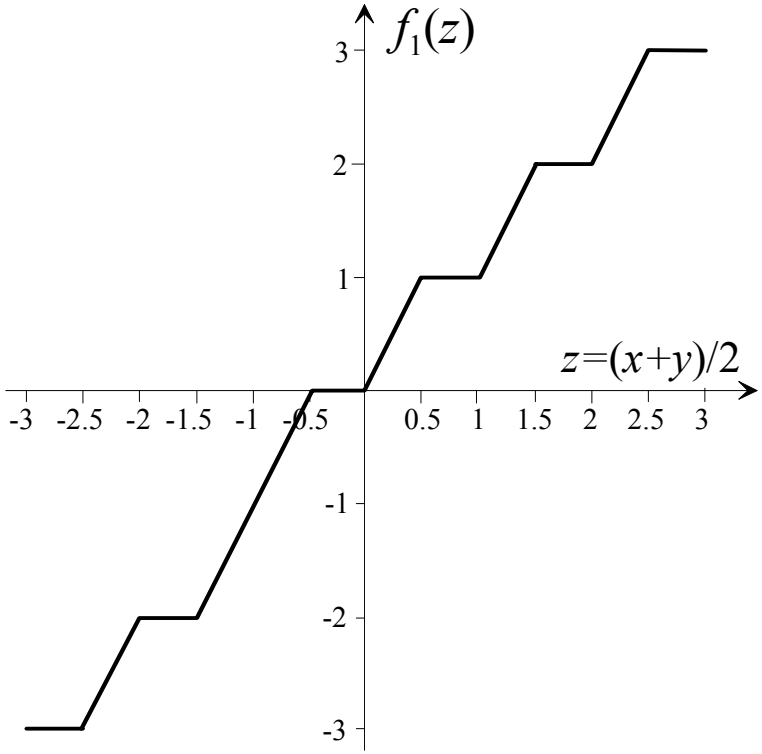


Рис.21 График функции $f_1(z)$.

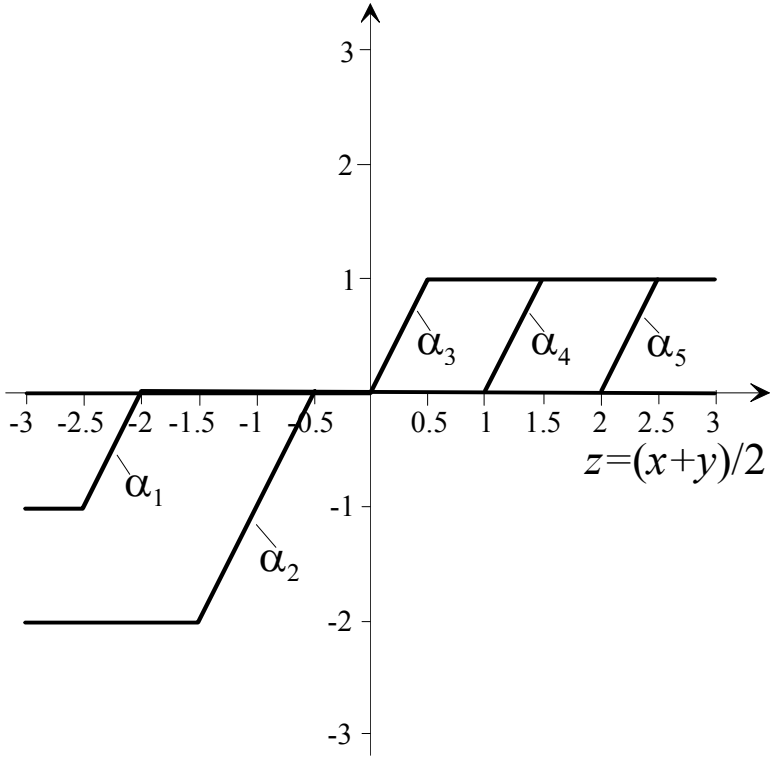


Рис.22 Пять составляющих подфункций.

Представим функцию $f_1(z)$ в виде суммы пяти подфункций $\alpha_j(z)$, показанных на рис.22: $f_1(z) = \sum_{j=1}^{5} \alpha_j(z)$.

4. Частные методы реализации правил фаззи вывода

Рассмотрим, как можно используя суммирующие усилители построить функции $\alpha_j(z)$, на примере формирования функции $\alpha_1(z)$. Обратимся к рис.23.

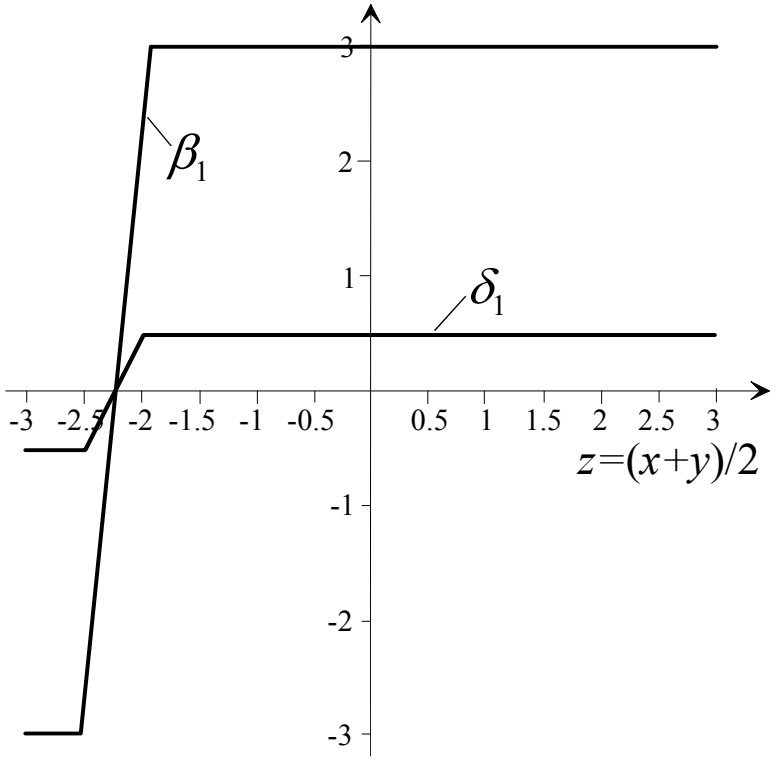


Рис.23 Построение функции $\alpha_1(z)$.

Функция $\beta_1(z)$ на рис.23 может быть реализована как $\beta_1(z) = -S(k_1 \cdot z + a_1)$.

Для $z = -2{,}25$, $\beta_1 = 0$ получим

$$a_1 = 2{,}25 \cdot k_1.$$

Принимая во внимание, что $k_1 = 6/0{,}5 = 12$, находим $a_1 = 27$ и, следовательно,

$$\beta_1(x, y) = -S(6 \cdot x + 6 \cdot y + 27),$$

$$\delta_1(x, y) = \tfrac{1}{6} \cdot \beta_1(x, y).$$

Окончательно получим

$$\alpha_1(x, y) = \delta_1(x, y) - 0{,}5 = \tfrac{1}{6} \cdot \beta_1(x, y) - 0{,}5.$$

4. Частные методы реализации правил фаззи вывода

Подобным образом находим все остальные функции $\alpha_j(x,y)$.

$$\alpha_2(x,y)=\tfrac{1}{3}\beta_2(x,y)-1=-\tfrac{1}{3}S(3\cdot x+3\cdot y+6)-1;$$
$$\alpha_3(x,y)=\tfrac{1}{6}\beta_3(x,y)+0{,}5=-\tfrac{1}{6}S(6\cdot x+6\cdot y-3)+0.5;$$
$$\alpha_4(x,y)=\tfrac{1}{6}\beta_4(x,y)+0{,}5=-\tfrac{1}{6}S(6\cdot x+6\cdot y-15)+0.5;$$
$$\alpha_5(x,y)=\tfrac{1}{6}\beta_5(x,y)+0{,}5=-\tfrac{1}{6}S(6\cdot x+6\cdot y-27)+0.5.$$

Окончательно, принимая во внимание взаимную компенсацию констант, получим

$$f_1(x,y)=\sum_{j=1}^{5}\alpha_j(x,y)=$$
$$S\{-\tfrac{1}{6}\cdot[\beta_1(x,y)+2\cdot\beta_2(x,y)+\beta_3(x,y)+\beta_4(x,y)+\beta_5(x,y)]\}=$$
$$=S_6[\tfrac{1}{6}S_1(6x+6y+27)+\tfrac{1}{3}S_2(3x+3y+6)+\tfrac{1}{6}S_3(6x+6y-3)+$$
$$+\tfrac{1}{6}S_4(6x+6y-15)+\tfrac{1}{6}S_5(6x+6y-27)]. \qquad (20)$$

Реализация функции $f_1(x,y)$, представляющей матрицу M_1, содержит 6 суммирующих усилителей.

4. Частные методы реализации правил фаззи вывода

4.2 Выделение матрицы с элементами, разделяемыми линией

Этот метод ориентирован на реализацию матриц, составленных из двух значений элементов, которые могут быть разделены прямой линией.

После выделения симметричной компоненты матрица M_2 является остаточной.

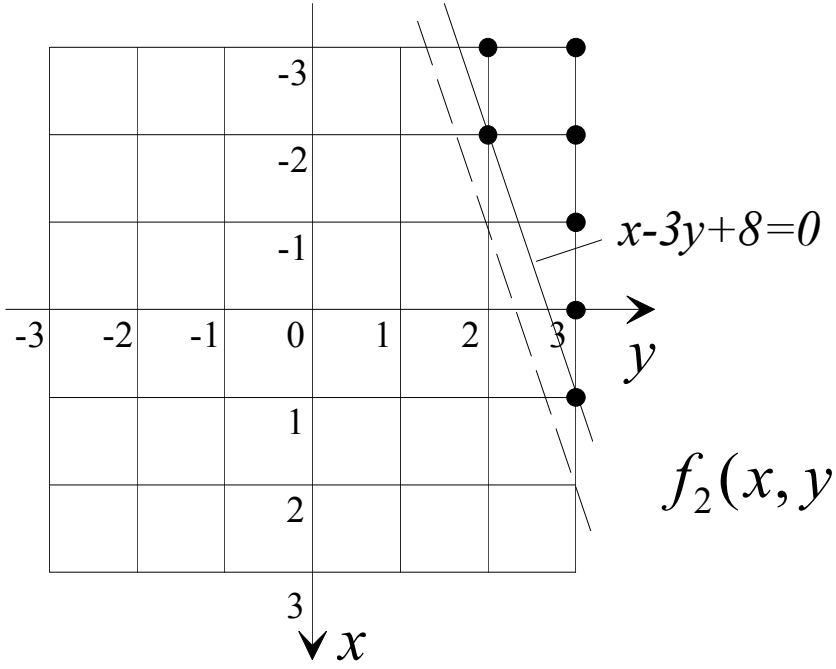
$$
\underset{M_2}{\begin{vmatrix}
\le 0 & \le 0 & 0+1+1 & +3+3 \\
\le 0 & 0+1 & 0 & 0 & +3 \ge 2 \\
0 & 0 & 0 & 0 & 0 & 0 \ge 2 \\
0 & 0 & 0 & 0 & 0 & +1 \ge 1 \\
0 & 0 & 0 & 0 & 0 & 0 \ge 1 \\
0 & 0 & 0+1 & 0 & 0 \ge 0 \\
+1 & 0+1 & 0 & 0 \ge 0 \ge 0
\end{vmatrix}}
=
\underset{M_3}{\begin{vmatrix}
0 & 0 & 0 & 0 & 0+3+3 \\
0 & 0 & 0 & 0 & 0+3+3 \\
0 & 0 & 0 & 0 & 0 & 0+3 \\
0 & 0 & 0 & 0 & 0 & 0+3 \\
0 & 0 & 0 & 0 & 0 & 0+3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}}
+
\underset{M_4}{\begin{vmatrix}
\le 0 & \le 0 & 0+1+1 \ge 0 \ge 0 \\
\le 0 & 0+1 & 0 & 0 \ge 0 \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 \ge 0 \\
0 & 0 & 0 & 0 & 0 & +1 \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 \ge 0 \\
0 & 0 & 0+1 & 0 & 0 \ge 0 \\
+1 & 0+1 & 0 & 0 \ge 0 \ge 0
\end{vmatrix}}
$$

Эта матрица содержит элементы типа $\le a$ и $\ge b$. Это означает, что вместо логических значений a и b возможно подставлять любые значения меньше a и больше b. Представим матрицу M_2 в виде суммы двух матриц M_3 и M_4.

4. Частные методы реализации правил фаззи вывода

Реализуем матрицу M_3. Матрица M_4 является новой остаточной матрицей. Обратимся к рисунку рис.24. Матрица M_3 состоит из элементов, имеющих только два логических значения (0 и +3). Эти значения могут быть разделены с помощью двух параллельных прямых линий: $x-3y+8=0$ и $x-3y+8=1$.

Введем новую переменную $w=x-3y+8$. Во всех точках, лежащих на линии $x-3y+8=0$ и выше, $w \le 0$ и во всех точках, лежащих на линии $x-3y+8=1$ и ниже, $w \le 1$.



Матрица M_3, представляющая функцию $f_2(x,y)$, может быть реализована как

$$f_2(x,y)=S_8\{S_7[3(-x+3y-8)]-3\}. \quad (21)$$

В этой реализации все значащие элементы матрицы равны +3.

Рис.24 Разделение элементов матрицы M_3.

Применение многозначной логики при проектировании фаззи-контроллеров

Лекция 11

4. Частные методы реализации правил фаззи вывода

4.3 Выделение матрицы с прямоугольной конфигурацией значимых элементов

Введем понятие *пирамидальной функции*. Это функция, которой соответствует матрица с единственным значимым элементом. Такая функция показана на рис.25. Ей соответствует правило типа

$$\text{if } (x = a)\,\&\,(y = b) \text{ then } f(x,y) = c \text{ else } f(x,y) = 0. \qquad (22)$$

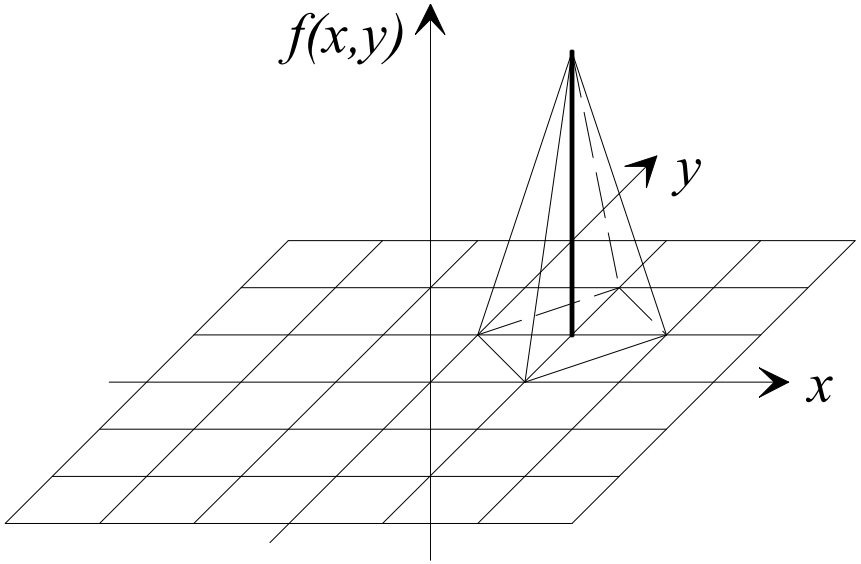


Рис.25 Пирамидальная функция.

Пирамидальная функция принимает некоторое фиксированное значение c $(-k \le c \ne 0 \le +k)$ в точке (a,b).

В точках, являющихся непосредственными соседями точки (a,b), и в остальных точках пространства матрицы значение этой функции равно нулю. Переход от значения c к значению 0 является линейным.

4. Частные методы реализации правил фаззи вывода

Для того, чтобы построить пирамидальную функцию обратимся к рис.26, на котором показаны две компонентные функции $M_{a+1}(x)$ и $-M_{a+1}(x)$ $(2k+1)$-значной логики $(-k \le x \le +k)$ проекции пирамиды на плоскость $y = b$ $(c = k)$:

$$M_{a+1}(x) = S[k \cdot (x - a - 1)], \quad -M_{a-1}(x) = S[k \cdot (-x + a - 1)]. \qquad (23)$$

Нетрудно проверить, что функция $\gamma(x) = S[M_{a+1}(x) - M_{a-1}(x) - 2k]$ имеет форму , представленную на рис.27.

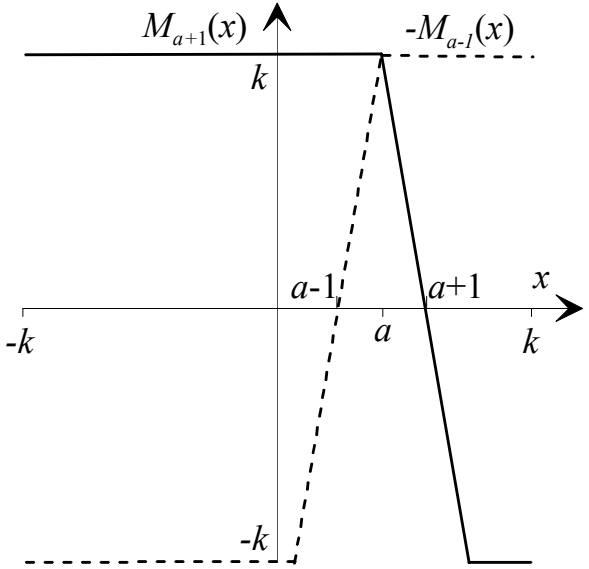


Рис.26 Компонентные функции проекции пирамиды на плоскость $y = b$, $c = k$.

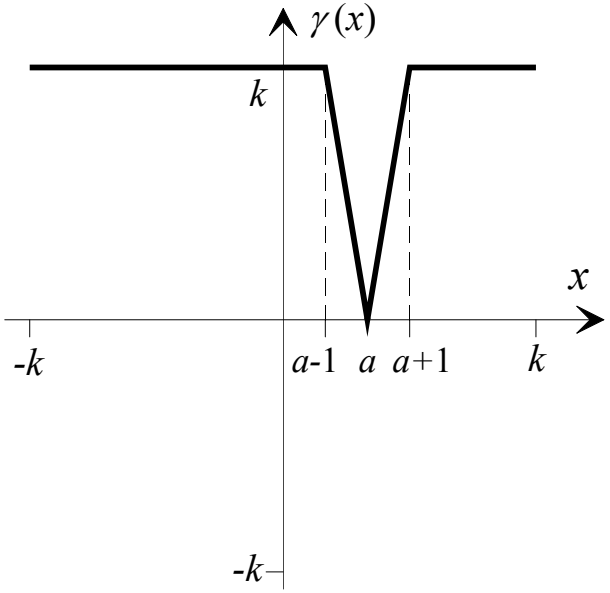


Рис.27 График функции $\gamma(x)$.

4. Частные методы реализации правил фаззи вывода

Аналогичным образом построим компонентные функции проекции пирамиды на плоскость $x = a$ для случая $c = k$ и функцию $\gamma(y)$:

$$M_{b+1}(y) = S[k \cdot (y - b - 1)], \quad -M_{b-1}(y) = S[k \cdot (-y + b - 1)],$$
$$\gamma(y) = S[M_{b+1}(x) - M_{b-1}(x) - 2k]. \qquad (24)$$

Построим функцию двух переменных, проекция которой на плоскости $y = b$ и $x = a$ имеют вид рис.27:

$$\gamma(x, y) = S[M_{a+1}(x) - M_{a-1}(x) + M_{b+1}(y) - M_{b-1}(y) - 4k]. \qquad (25)$$

Теперь нетрудно построить пирамидальную функцию высотой “c”, представленную на рис.25:

$$f(x, y) = \frac{c}{k} S[\gamma(x, y) - k] \qquad (26)$$

Таким образом, высота пирамиды регулируется весовым коэффициентом входа следующего усилителя. Знак пирамиды может быть при необходимости изменен при построении промежуточной функции $\gamma(x,y)$.

4. Частные методы реализации правил фаззи вывода

Для обеспечения монотонности кусочно-линейной аппроксимации между смежными логическими уровнями необходимо обеспечить “хорошее сшивание” пирамидальных функций с уже реализованными функциями. Пересечения пирамидальной функции (25) типа рис.25 с плоскостями $y = \{b; b \pm 0{,}5; b \pm 1\}$ и с плоскостями $x = \{a; a \pm 0{,}5; b \pm 1\}$ представлены на рис.28, и рис.28,b соответственно.

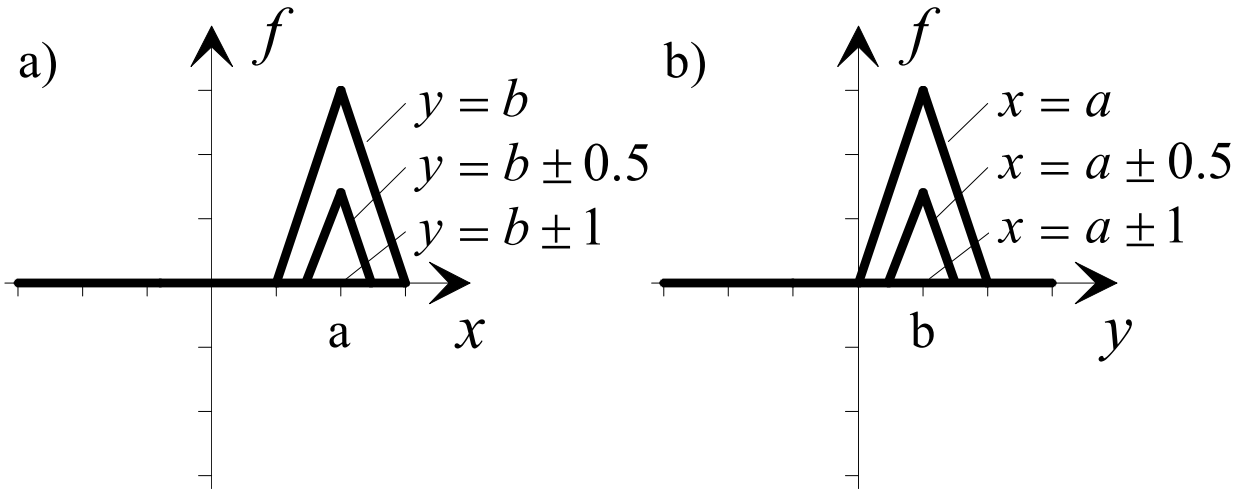


Рис.28 Возможные сечения пирамидальной функции a) по координате x и b) по координате y.

4. Частные методы реализации правил фаззи вывода

Иногда использование пирамидальной функции (25) не обеспечивает монотонности кусочно-линейной аппроксимации между логическими уровнями при “сшивании” пирамиды с другими функциями. В этом случае может потребоваться другой вид пересечений пирамидальной функции с плоскостями, по каждой из координат. Варианты таких пересечений приведены на рис.29.

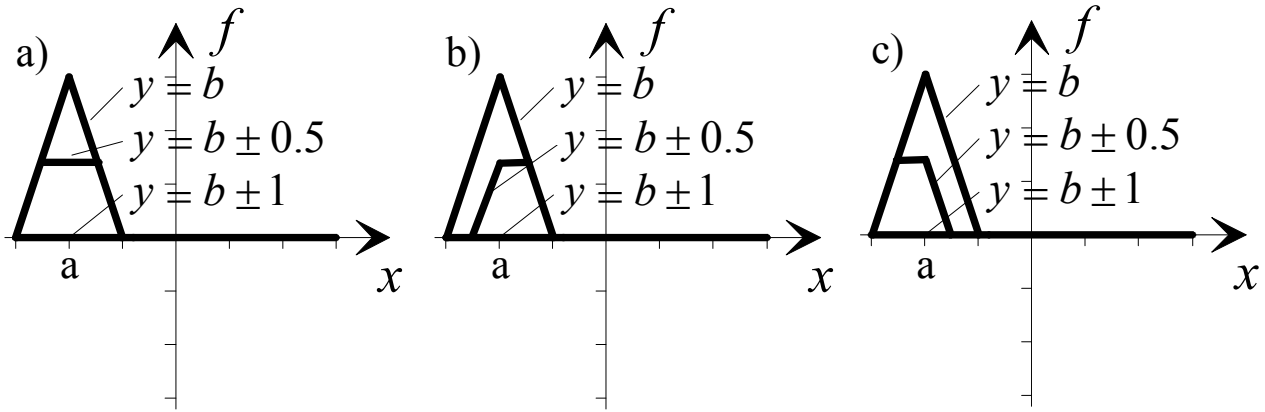


Рис.29 Возможные графики пирамидальной функции по одной из координат: a) центральная трапеция, b) правосторонняя трапеция, c) левосторонняя трапеция.

Для того, чтобы пересечение пирамиды с плоскостями по каждой из координат превращалась в центральную трапецию, достаточно подставить в функции (23)-(26) вместо x, y и a, b соответственно $z = x + y$, $w = x - y$ и $c = a + b$, $d = a - b$.

4. Частные методы реализации правил фаззи вывода

Это означает переход к пирамидальной функции, показанной на рис.30. По аналогии с (23) – (26) получим для такой пирамиды высотой k

$$f(x,y) = S\{S[S(k(x+y-a-b-1)) + S(k(-x-y+a+b-1)) + S(k(x-y-a+b-1)) + S(k(-x+y+a-b-1)) - 4k] - k\}. \quad (27)$$

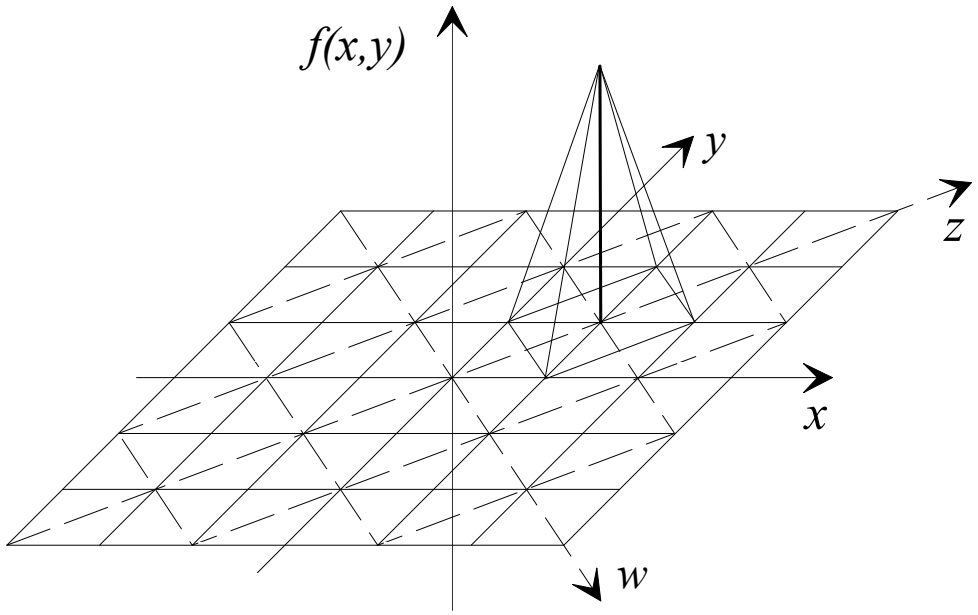


Рис.30 Пирамидальная функция в координатах $z = x + y,\ w = x - y$.

Для реализации функции, которая имеет формы графиков по одной из переменных (например, x) в виде правосторонней трапеции (рис.29,b), введем две вспомогательные функции $\varphi[x, \xi(y)]$ и $\psi[x, \xi(y)]$, показанные на рис.31.

Нетрудно проверить, что эти вспомогательные функции могут быть реализованы как

4. Частные методы реализации правил фаззи вывода

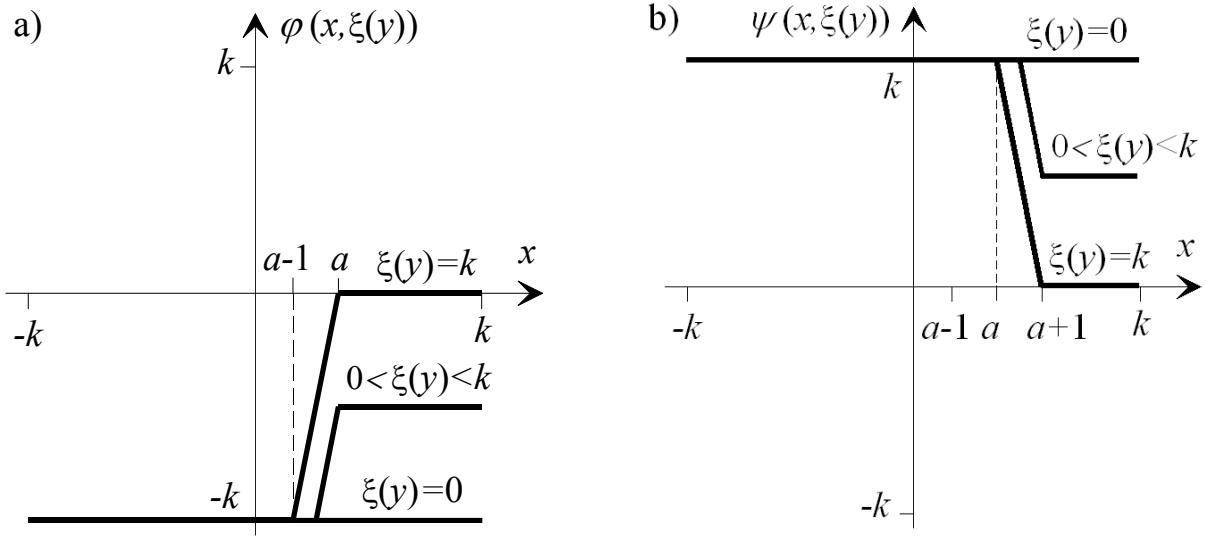


Рис.31 Вспомогательные функции a) $\varphi[x,\xi(y)]$ и b) $\psi[x,\xi(y)]$.

$$\varphi[x,\xi(y)] = S[M_{a-1}(x) + 2k - \xi(y)],$$
$$\psi[x,\xi(y)] = S[-M_a(x) - 2k + \xi(y)]. \qquad (28)$$

Функция $f(x,y) = \varphi[x,\xi(y)] + \psi[x,\xi(y)]$ имеет форму правосторонней трапеции по оси x, изображенную на рис.29,b. Если функция $\xi(y)$ имеет треугольную форму, $\xi(b) = \pm$ k и $\xi(y < b-1) = \xi(y > b+1) = 0$, то $f(x,y)$ – пирамидальная функция.

4. Частные методы реализации правил фаззи вывода

Рассмотрим способ построения пирамиды с левосторонней трапецией по оси x (рис.29.c). Снова построим две вспомогательные функции $\varphi[x,\xi(y)]$ и $\psi[x,\xi(y)]$, представленные на рис.32.

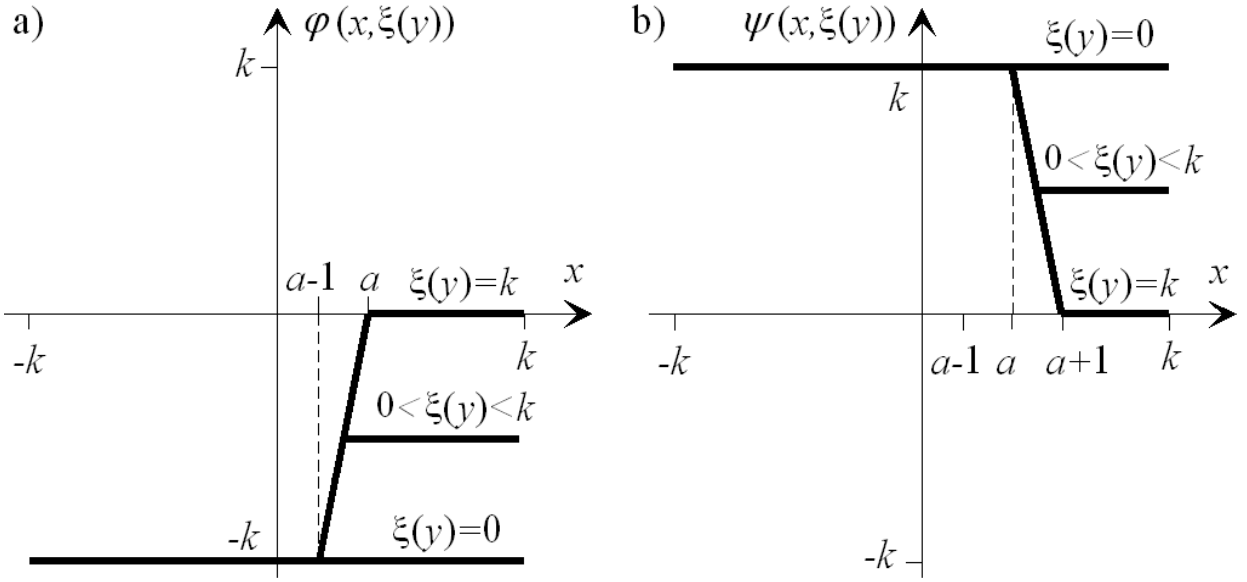


Рис.32 Вспомогательные функции a) $\varphi[x,\xi(y)]$ и b) $\psi[x,\xi(y)]$.

Реализуем вспомогательные функции как

$$\varphi[x,\xi(y)] = S[M_a(x) - 2k + \xi(y)] - 2k + \xi(y),$$
$$\psi[x,\xi(y)] = S[-M_{a+1}(x) + 2k - \xi(y)] + 2k - \xi(y).$$

4. Частные методы реализации правил фаззи вывода

Тогда функция

$$f(x,y)=\varphi[x,\xi(y)]+\psi[x,\xi(y)]= \\ =S[M_a(x)-2k+\xi(y)]+S[-M_a(x)+2k-\xi(y)]. \qquad (29)$$

имеет форму левосторонней трапеции по оси x, изображенную на рис.29,b.

Пирамидальные функции могут быть полезны не только при реализации правил типа (22), но также и при реализации правил с интервальными условиями типа

$$\text{if } (a_1 \le x \le a_2 \text{ and } b_1 \le y \le b_2) \\ \text{then } f(x,y)=k \text{ else } f(x,y)=0. \qquad (30)$$

Интервальные условия могут быть получены из реализации точечных условий правила (22) простым изменением констант в маск-функциях. Правило (30) представляет матрицы с прямоугольными конфигурациями значимых элементов, которые представляют собой усеченные пирамиды.

Заметим, что функции типа (25) могут быть построены для произвольного числа переменных. Реализация пирамидальной функции двух переменных требует 6 усилителей.

4. Частные методы реализации правил фаззи вывода

4.4 Выделение матрицы со значащими элементами, лежащими на диагонали

Вернемся к примеру. Разобьем матрицу M_4 на две матрицы (M_5 and M_6) и реализуем матрицу M_5. Матрица M_6 – новая остаточная матрица.

$$M_4 = M_5 + M_6$$

$$\begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & \ge 0 \\ +1 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ +1 & 0 & 0 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix}$$

Матрицу M_5, в свою очередь, разобьем на две следующим образом $M_5 = (M_{51} + M_{52})/3$. Пусть такому разбиению соответствует выражение

$$f_3(x, y) = \tfrac{1}{3}[f_{31}(x, y) + f_{32}(x, y)].$$

4. Частные методы реализации правил фаззи вывода

$$M_5 = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0+1 & \\ 0 & 0 & 0 & 0 & 0+1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0+1 & 0 & 0 & 0 & \\ 0 & 0+1 & 0 & 0 & 0 & 0 & \end{vmatrix} = \frac{1}{3} \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0+3 & \\ 0 & 0 & 0 & 0 & 0+3 & 0 & \\ 0 & 0 & 0 & 0+3 & 0 & 0 & \\ 0 & 0 & 0+3 & 0 & 0 & 0 & \\ 0 & 0+3 & 0 & 0 & 0 & 0 & \end{vmatrix} + \frac{1}{3} \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

M_5 M_{51} M_{52}

Элементы со значением “+3” матрицы M_{51}, лежат на прямой $x + y - 2 = 0$. Эта матрица может быть представлена функцией одной переменной $f_{31}(z = x + y)$, которая соответствует правилу

$$\text{if } z = 2 \text{ then } f_{31}(z) = 3 \text{ else } f_{31}(z) = 0.$$

Функция $f_{31}(z)$ может быть построена как показано на рис.33, из которого легко понять, что

$$f_{31}(z) = \alpha_6(z) + 3 = S[\beta_6(z) + \delta_6(z) + 6] + 3,$$
$$\beta_6(z) = S(3z - 3), \quad \delta_6(z) = S(-3z + 9).$$

4. Частные методы реализации правил фаззи вывода

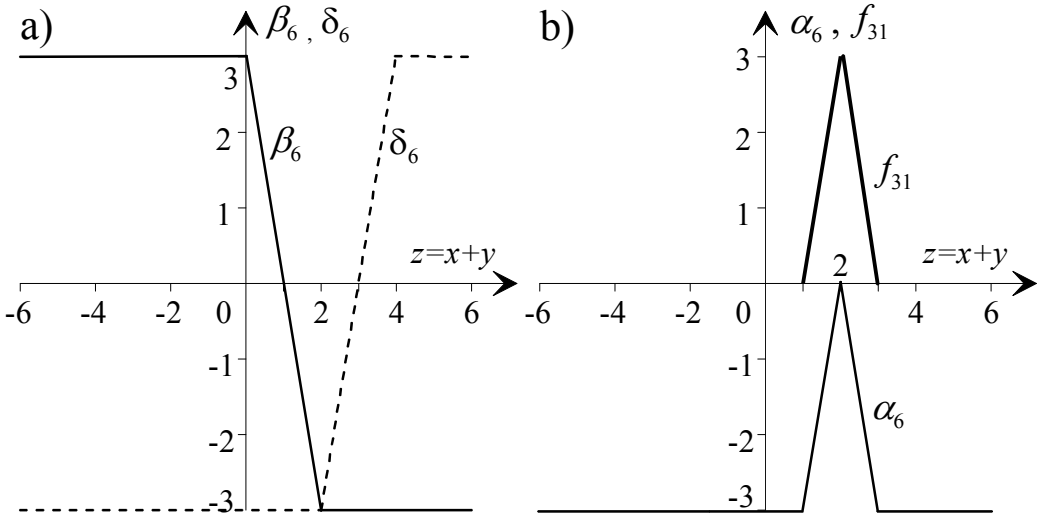


Fig.33 Способ реализации функции $f_{31}(z)$.

Функцию f_{32} целесообразно реализовать как пирамидальную функцию переменных $z=(x+y)$ и $w=(x-y)$, как показано на рис.30. Это обеспечивает хорошее “сшивание” функций f_{32} и f_{31}, а также f_3 и f_1. Подстановка переменных z и w в формулу (27) вместо $x+y$ и $x-y$ соответственно, a =1, b =1, $k=3$ и изменение знака функции дают

$$f_{32}(z,w)=S\{S[\beta_6(z)+\delta_6(z)+\beta_7(w)+\delta_7(w)+12]+3\},$$

где $\beta_6(z)$ и $\delta_6(z)$ уже реализованы и $\beta_7(w)=S(3w+3)$, $\delta_7(w)=S(-3w+3)$.

4. Частные методы реализации правил фаззи вывода

Окончательно, функция $f_3(x,y)$, соответствующая матрице M_5, может быть реализована как

$$\begin{aligned} f_3(x,y) &= \tfrac{1}{3}[f_{31}(x,y)+f_{32}(x,y)] = \tfrac{1}{3}[\alpha_6(x,y)+f_{32}(x,y)]+1 = \\ &= \tfrac{1}{3}S_{14}\{S_9(3x+3y-3)+S_{10}(-3x-3y+9)+ \\ &\quad + S_{13}[S_9(3x+3y-3)+S_{10}(-3x-3y+9)+ \\ &\quad + S_{11}(3x-3y+3)+S_{12}(-3x+3y+3)+12]+9\}+1. \end{aligned} \tag{31}$$

4.5 Реализация матрицы M_6

Разобьем матрицу M_6 на две матрицы (M_7 and M_8) и реализуем сначала матрицу M_7. В новой остаточной матрице M_8 все элементы определены.

Значащие элементы матрицы M_7 образуют прямоугольную конфигурацию и могут быть представлены в виде функции $f_4(x,y)$, определяющей правило

$$\text{if } (x=-3)\,\&\,(y\geq 0) \text{ then } f_4(x,y)=1 \text{ else } f_4(x,y)=0.$$

4. Частные методы реализации правил фаззи вывода

$$
\begin{array}{ccc}
M_6 & M_7 & M_8 \\
\left|\begin{array}{ccccccc}
\le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\
\le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\
+1 & 0 & 0 & 0 & 0 & \ge 0 & \ge 0
\end{array}\right| &
= \left|\begin{array}{ccccccc}
0 & 0 & 0 & +1 & +1 & +1 & +1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right| +
\left|\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & +1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right|
\end{array}
$$

По аналогии с построением формул (23) – (26) построим две вспомогательные функции: $\gamma_1(x)$ для условия $x = -3$ и $\gamma_1(y)$ для условия $y \ge 0$. Эти функции представлены на рис.34 и могут быть реализованы в виде

$$\gamma_1(x) = S[-M_{-3}(x)], \quad \gamma_1(y) = S[M_{-1}(y) + 3].$$

На основе этих функций можно построить функцию двух переменных:

$$\gamma_1(x, y) = S[-M_{-3}(x) + M_{-1}(y) + 3].$$

4. Частные методы реализации правил фаззи вывода

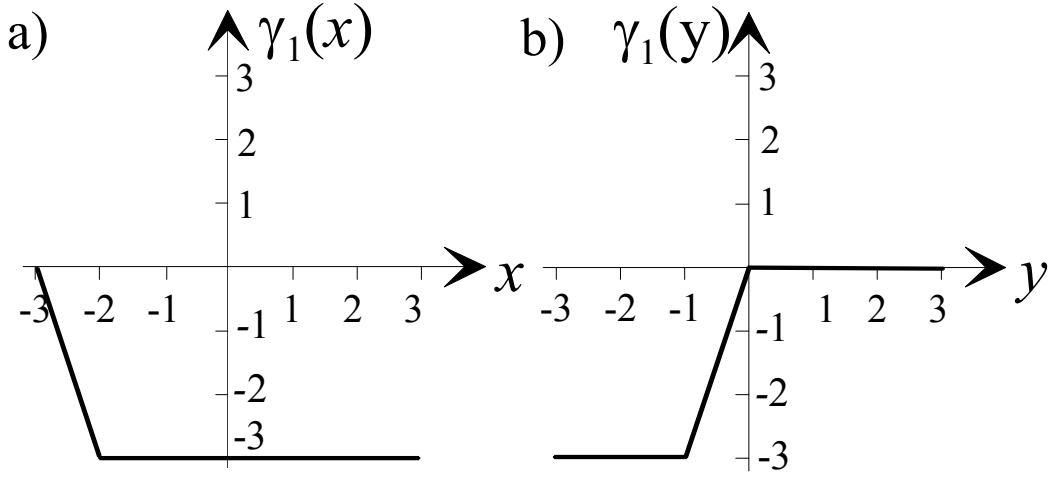


Рис.34 Две вспомогательные функции: a) $\gamma_1(x)$ для условия $x = -3$ и b) $\gamma_1(y)$ для условия $y \geq 0$.

Принимая во внимание, что для $k = 3$

$$-M_{-3}(x) = S(-3x - 9), \quad M_{-1}(y) = S(3y + 3),$$

нетрудно найти схему, реализующую M_7:

$$f_4(x, y) = \tfrac{1}{3} S_{17}[S_{16}(-3x - 9) + S_{15}(3y + 3) + 3] + 1. \qquad (32)$$

4. Частные методы реализации правил фаззи вывода

Последняя матрица M_8 имеет только два ненулевых элемента с координатами ($x = +3, y = -3$) и ($x = -2, y = -1$).

M_8

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Обозначим функции, реализующие эти элементы, как f_5 and f_6 соответственно. Способы построения и реализации этих функций зависят от координат соответствующих им элементов и условий их “хорошего сшивания” с фрагментами уже реализованных функций.

Для получения монотонной кусочно-линейной аппроксимации между логическими уровнями функций f_1 и f_5 графики функции f_5 вдоль осей ее аргументов должны иметь форму рис.29,a или рис29,b.

4. Частные методы реализации правил фаззи вывода

Функция $f_5(x,y)$ может быть реализована как пирамидальная функция в соответствии с (27), однако подход, показанный на рис.35, дает более простую схему.

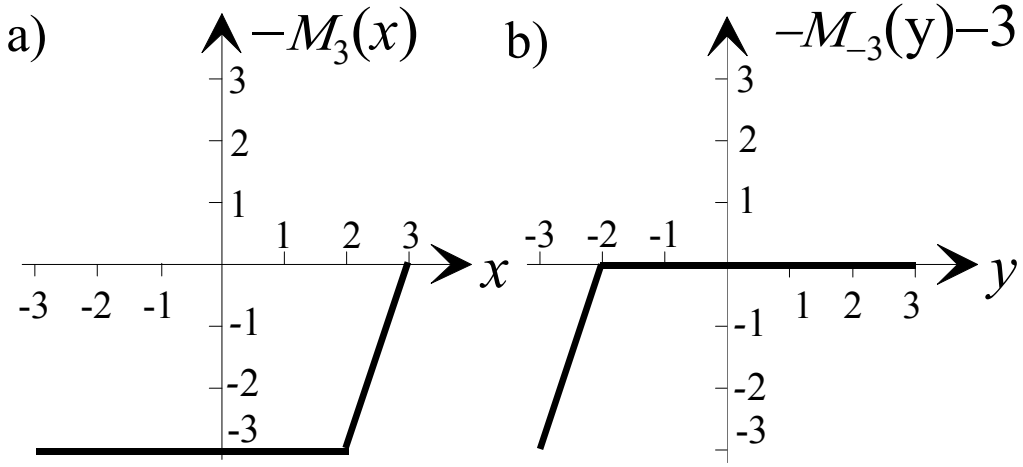


Рис.35 Вспомогательные функции: a) $-M_3(x)$ и b) $M_{-3}(y) - 3$.

Теперь нетрудно построить функцию $f_5(x,y)$

$$f_5(x,y) = \tfrac{1}{3}\{S[-M_{-3}(y) - 3 - M_3(x)] - M_3(x)\}$$

и реализующую ее схему

$$\begin{aligned} f_5(x,y) &= \tfrac{1}{3}\{S[-M_{-3}(y) - 3 - M_3(x)] - M_3(x)\} = \\ &= \tfrac{1}{3}S_{20}[S_{18}(-3y-9) + S_{19}(-3x+9) - 3] + \tfrac{1}{3}S_{19}(-3x+9). \end{aligned} \qquad (33)$$

4. Частные методы реализации правил фаззи вывода

Как показали эксперименты, монотонная кусочно-линейная аппроксимация между логическим уровнем в точке (-2,-1) и логическими уровнями в соседних точках функций f_1 и f_4 может быть достигнута, если пирамидальная функция f_6 в точке (-2,-1) реализуется в соответствии с формулой 23.

$$f_6(x,y) = \tfrac{1}{3} S\{S[S(3x+3y+6) + \mathrm{S}(-3x-3y-12) + \\ + S(3x-3y) + S(-3x+3y-6) - 12] - 3\}.$$

После некоторого преобразования, связанного с изменением порядка суммирования, с целю экономии одного суммирующего усилителя эта функция имеет вид:

$$f_6(x,y) = \tfrac{1}{3} S_{25}[S_{21}(-3x-3y-6) + S_{22}(3x+3y+12) + \\ + S_{23}(-3x+3y) + S_{24}(3x-3y+6) + 12] + 1. \qquad (34)$$

4. Частные методы реализации правил фаззи вывода

4.6 Реализация контроллера

Выпишем реализации составляющих декомпозиции исходной матрицы контроллера (формулы (20), (21), (31) – (34)).

$$f_1(x,y) = S_6[\tfrac{1}{6}S_1(6x+6y+27) + \tfrac{1}{3}S_2(3x+3y+6) + \tfrac{1}{6}S_3(6x+6y-3) + \\ + \tfrac{1}{6}S_4(6x+6y-15) + \tfrac{1}{6}S_5(6x+6y-27)],$$

$$f_2(x,y) = S_8\{S_7[3(-x+3y-8)]-3\},$$

$$f_3(x,y) = \tfrac{1}{3}S_{14}\{S_9(3x+3y-3) + S_{10}(-3x-3y+9) + \\ + S_{13}[S_9(3x+3y-3) + S_{10}(-3x-3y+9) + \\ + S_{11}(3x-3y+3) + S_{12}(-3x+3y+3)+12]+9\}+1,$$

$$f_4(x,y) = \tfrac{1}{3}S_{17}[S_{16}(-3x-9) + S_{15}(3y+3)+3]+1,$$

$$f_5(x,y) = \tfrac{1}{3}S_{20}[S_{18}(-3y-9) + S_{19}(-3x+9)-3] + \tfrac{1}{3}S_{19}(-3x+9),$$

$$f_6(x,y) = \tfrac{1}{3}S_{25}[S_{21}(-3x-3y-6) + S_{22}(3x+3y+12) + \\ + S_{23}(-3x+3y) + S_{24}(3x-3y+6)+12]+1.$$

4. Частные методы реализации правил фаззи вывода

Эту систему реализованных функций следует дополнить схемами изменения знака переменных: $-x = S_{26}(x)$ и $-y = S_{27}(y)$
и схемой формирования выходного сигнала контроллера:

$$f(x,y) = \sum_{j=1}^{6} f_j(x,y) = S_6 + S_8 + \tfrac{1}{3}S_{14} + \tfrac{1}{3}S_{17} + \tfrac{1}{3}S_{19} + \tfrac{1}{3}S_{20} + \tfrac{1}{3}S_{25} + 3 =$$
$$= S_{29}[S_{28}(S_6 + S_8 + \tfrac{1}{3}S_{14} + \tfrac{1}{3}S_{17} + \tfrac{1}{3}S_{19} + \tfrac{1}{3}S_{20} + \tfrac{1}{3}S_{25} + 3)].$$

Схему контроллера можно упростить за счет объединения суммирующих усилителей, работающих в линейном режиме. В данном случае могут быть совмещены усилители S_6 и S_8. По аналитическому представлению реализации контроллера можно изобразить его схему графически, которая здесь не приведена.

Каждый шаг проектирования схемы контроллера необходимо проверять с помощью какой-либо системы моделирования с целью исключения ошибок и проверки качества межуровневой аппроксимации. Схемы данного контроллера были доведены до транзисторного уровня и проверялись SPICE моделированием. Использовалась модель транзистора 0,4 мкм 7-го уровня из библиотеки MOSIS BSIM3v3.1. По этой причине была выбрана простейшая трехинверторная схема суммирующего усилителя с сопротивлением обратной связи 1,5Мом.

4. Частные методы реализации правил фаззи вывода

В реальных проектах рекомендуется использовать стандартные схемы операционных усилителей. Результаты SPICE моделирования схемы контроллера приведены на рис.36, на котором изображена поверхность отклика, построенная с помощью GNUplot.

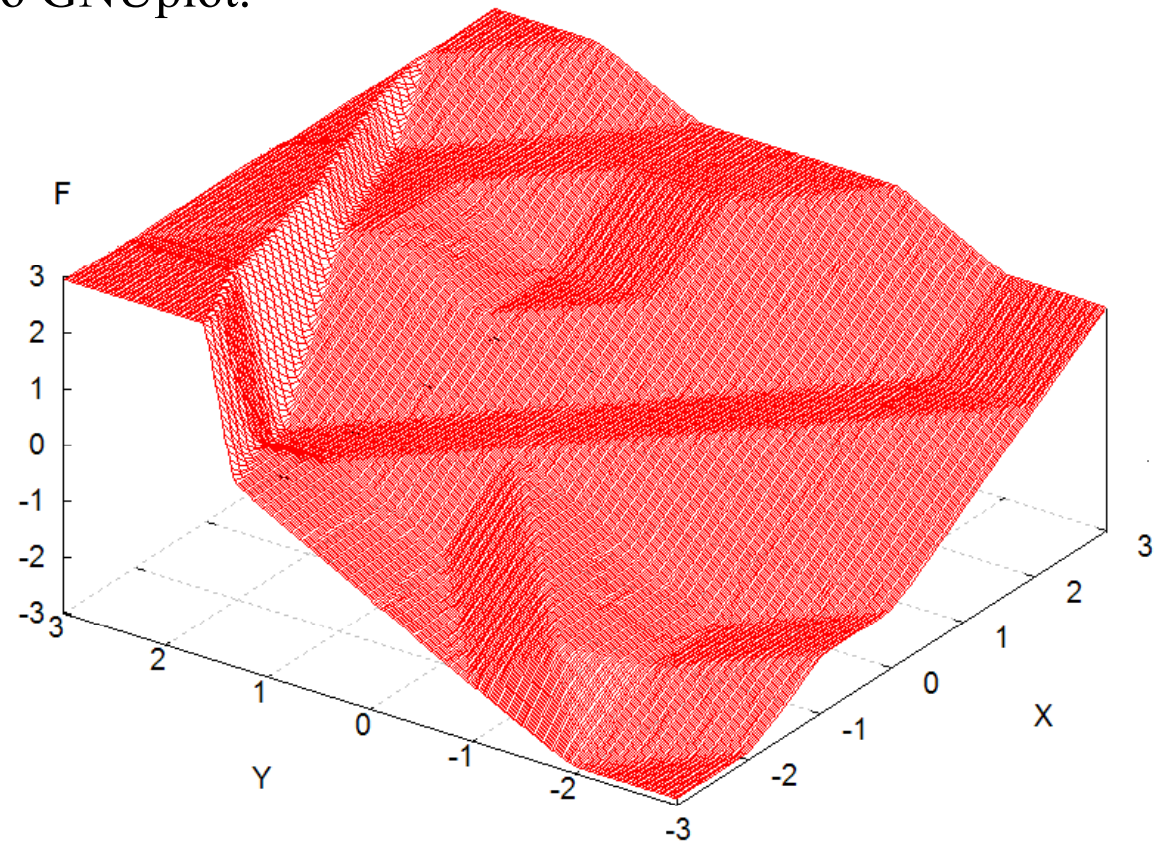


Рис.36 Поверхность отклика контроллера.

5. Преобразование аналоговых сигналов в многозначные логические переменные

Если лингвистические переменные с максимальными весами, соответствующие некоторой аналоговой переменной, равномерно распределены в диапазоне изменения аналоговой переменной и их функции принадлежности имеют кусочно-линейную форму, то таким лингвистическим переменным можно поставить в соответствие логические значения многозначной логической переменной.

В этом случае на вход схемы, реализующей функцию многозначной логики, можно подавать непосредственно аналоговую переменную вместо многозначной переменной.

Если же лингвистические переменные с максимальными весами распределениы неравномерно в диапазоне изменения соответствующей аналоговой переменной, то возможны два варианта:

1. Увеличить число логических градаций так, чтобы распределение стало равномерным. Увеличение значности логических переменных приводит к значительному усложнению реализации функций многозначной логики.

5. Преобразование аналоговых сигналов в многозначные логические переменные

2. Поставить в соответствие лингвистическим переменным с максимальными весами логические значения с тем же распределением, а затем построить схему, преобразующую неравномерное распределение уровней логической переменной в равномерное. Такую схему можно рассматривать как некоторый аналог фаззификатора.

Именно второй вариант и будет рассматриваться далее.

Способ построения преобразователя неравномерного распределения в равномерное проиллюстрируем на примере, взятом из описания фаззи контроллера для управления контейнерным краном (http://www.fuzzytech.com/e/e_a_pfd.html).

В качестве примера выбрана аналоговая переменная “угол”, для которой лингвистические переменные и их функции принадлежности приведены на рис.37.

Переменная “угол” изменяется в пределах ($-90^o \div +90^o$) . Пусть этому диапазону соответствует диапазон изменения напряжений (0V ÷ +3,5V), представляющих значения этой переменной.

5. Преобразование аналоговых сигналов в многозначные логические переменные

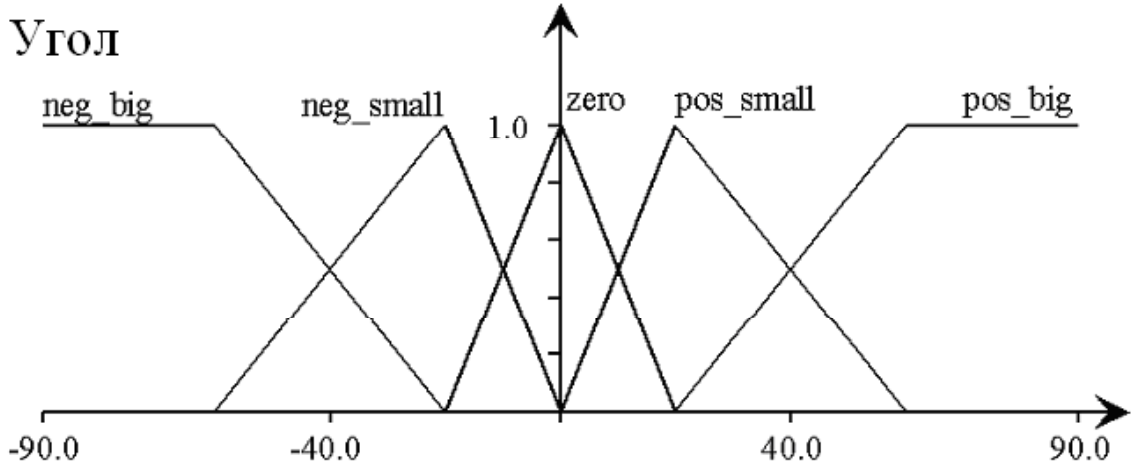


Рис.37 Функции принадлежности для переменной “угол”.

Соответствие лингвистических переменных и их логических значений значениям угла и напряжениям приведено в табл.11.

Таблица 11: Соответствие логических значений напряжениям

neg_big	neg_small	zero	pos_small	pos_big
-90°÷-60° (0÷0.58)*V*	-20° 1.361*V*	0° 1.75*V*	20° 2.139*V*	60°÷90° (2.917÷3.5)*V*
-2 0*V*	-1 0.875*V*	0 1.75*V*	+1 2.625*V*	+2 3.5*V*

5. Преобразование аналоговых сигналов в многозначные логические переменные

Функция преобразования неравномерно распределенных логических уровней к равномерно распределенным и линейной аппроксимации между логическими уровнями приведена на рис.38, а ее декомпозиция на составляющие функции – на рис.39.

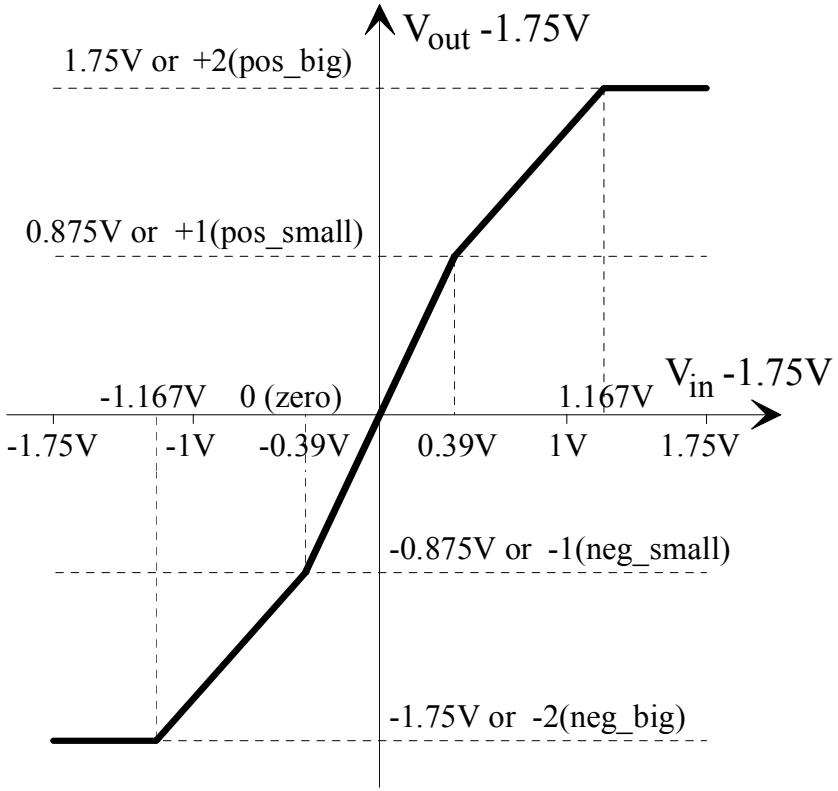


Рис.38 Кусочно-линейная функция преобразования для “угла”.

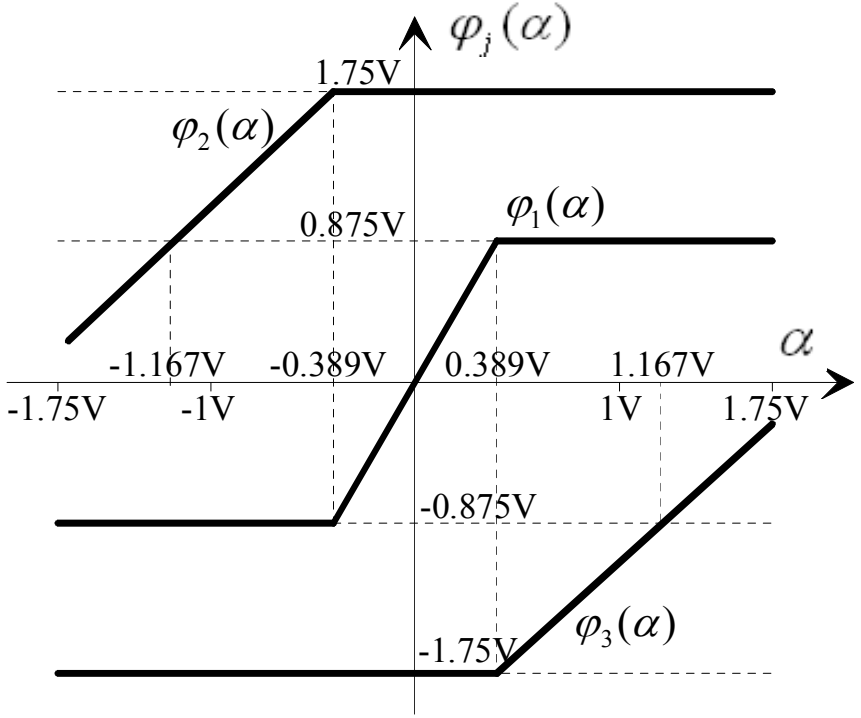


Рис.39 Составляющие функции.

5. Преобразование аналоговых сигналов в многозначные логические переменные

На обеих рисунках начало координат сдвинуто на средний уровень диапазона напряжений.

Принимая во внимание, что α – угол, представленный напряжением V_{in}, а $F(\alpha)$ – функция преобразования угла, значения которой представлены напряжением V_{out}, реализуем функцию $F(\alpha)$ следующим образом:

$$\varphi_1(\alpha) = -0.5S_1(4.5\alpha);$$
$$\varphi_2(\alpha) = -S_2(1.125\alpha + 2.19);$$
$$\varphi_3(\alpha) = -S_3(1.125\alpha - 2.19);$$
$$F_1(\alpha) = S(-\varphi_1(\alpha) - \varphi_2(\alpha) - \varphi_3(\alpha)) =$$
$$= S_4(\tfrac{1}{2}S_1(4.5\alpha) + S_2(1.125\alpha + 2.19) + S_3(1.125\alpha - 2.19)).$$

Здесь коэффициенты при α определяются как тангенс углов наклона соответствующих функций на рис.39, а для нахождения свободных членов нетрудно построить соответствующие уравнения.

Схема преобразователя для угла α изображена на рис.40.

5. Преобразование аналоговых сигналов в многозначные логические переменные

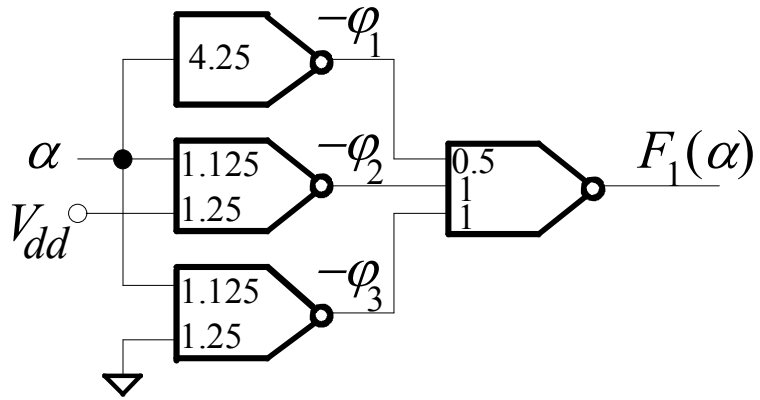


Рис.40 Преобразователь аналоговой переменной “угол” (α).

Веса входов V_{dd} и земли определяются как результат деления положительного свободного члена на $+k$ и отрицательного свободного члена на $-k$ соответственно.

Легко видеть, что рассмотренный подход легко может быть применен и для обратного преобразования многозначной логической переменной с равномерным распределением уровней по шкале напряжений в аналоговую переменную с неравномерным распределением и линейной аппроксимацией между уровнями. Такой преобразователь можно рассматривать как некоторый аналог дефаззификатора.

6. *Заключение*

В приведенных примерах реализации контроллеров используются суммирующие усилители простейшего типа. Однако при проектировании реальных контроллеров рекомендуется использовать более совершенные схемы операционных усилителей, например, дифференциального типа.

Таким образом показано, что все составные части фаззи контроллеров могут быть эффективно реализованы как аналоговые устройства с помощью суммирующих усилителей с насыщением. Такого типа реализация фаззи контроллеров обладает наименьшим временем отклика, более высокой надежностью, меньшей потребляемой мощностью, занимает меньшую площадь кристалла и т.д.