1. **Методы анализа и синтеза комбинационных схем.**

Техническим аналогом булевой функции в вычислительной технике является, так называемая, ***комбинационная схема***, на вход которой поступают и с выхода снимаются электрические сигналы в виде одного из уровней напряжения, соответствующих значениям логического 0 и логической 1.



Для выяснения, что же такое комбинационная схема, рассмотрим схему S, имеющую *m* входов и *n* выходов (рис. 1). На её входы могут быть поданы наборы значений входных переменных *Xi* {0,1}, , а на выходах формируются выходные переменные *Yj*∈{0,1}, .

Схема S называется ***комбинационной***, если каждую из *n* функций её выходов *Y1,Y2, ..., Yn* можно представить как булеву функцию входных переменных *X1, X2, ..., Xm.*

Комбинационная схема описывается с помощью системы уравнений *(1)*, где *Fi* – булева функция.



*(1)*

Как следует из определения комбинационной схемы, значения выходных переменных *Yj* в произвольный момент времени однозначно определяются значениями входных переменных *Xi*.

Структурно комбинационная схема может быть представлена как совокупность элементарных логических схем – логических элементов (ЛЭ). ЛЭ выполняют над входными переменными элементарные логические операции типа *И-НЕ*, *И*, *ИЛИ*, *ИЛИ-НЕ* и т.д. Число входов логического элемента соответствует числу аргументов воспроизводимой им булевой функции. Графическое изображение комбинационной схемы, при котором показаны связи между различными элементами, а сами элементы представлены условными обозначениями, называется ***функциональной*** схемой.

В ходе разработки комбинационных схем приходится решать задачи анализа и синтеза.

***Задача анализа*** состоит в определении статических и динамических свойств комбинационной схемы. В статике определяются булевы функции, реализуемые комбинационной схемой по известной ей структуре. В динамике рассматривается способность надёжного функционирования схемы в переходных процессах при смене значений переменных на входах схемы, т.е. определяется наличие на выходах схемы возможных нежелательных импульсных сигналов, которые не следуют непосредственно из выражений для булевых функций, реализуемых схемой.

***Задача синтеза*** заключается в построении из заданного набора логических элементов комбинационной схемы, реализующей заданную систему булевых функций.

Решение задачи синтеза не является однозначным, можно предложить различные варианты комбинационных схем, реализующих одну и ту же систему булевых функций, но отличающихся по тем или иным параметрам. Разработчик комбинационных схем из этого множества вариантов выбирает один, исходя из дополнительных критериев: минимального количества логических элементов, необходимых для реализации схемы, максимального быстродействия и т.д. Существуют различные методы синтеза комбинационных схем, среди которых наиболее разработан канонический метод.

**1.1. Канонический метод синтеза комбинационных схем.**

Как отмечалось выше, комбинационная схема (КС) может иметь несколько выходов. При каноническом методе предполагается, что каждая выходная функция реализуется своей схемой, совокупность которых и даёт требуемую КС. Поэтому синтез сложной КС с n выходами заменяется синтезом n схем с одним выходом.

Согласно каноническому методу синтез КС включает в себя ряд этапов.

**1.** Подлежащая реализации булева функция (или её отрицание) представляется в виде СДНФ.

**2.** С использованием методов минимизации определяется минимальная ДНФ (МДНФ) или минимальная КНФ (МКНФ). Из полученных двух минимальных форм выбирается более простая.

**3.** Булеву функцию в минимальной форме согласно п.2 представляют в заданном (или выбранном разработчиком) базисе .

**4.** По представлению функции в заданном базисе строят комбинационную схему.

Необходимо отметить, что подлежащая реализации булева функция *F(X1,X2,...,Xm)* может быть задана не на всех возможных наборах аргументов *X1, X2, ..., Xm*. На тех наборах, где функция неопределенна, её доопределяют так, чтобы в результате минимизации получить более простую МДНФ или МКНФ. При этом упростится и сама КС. Кроме того, довольно часто с целью получения ещё более простого представления функции МДНФ, полученная в п.2, представляется в так называемой скобочной форме, т.е. выносятся за скобки общие части импликант МДНФ.

Рассмотрим канонический метод синтеза на примере построения схемы полного одноразрядного двоичного сумматора.

Как известно из курса машинной арифметики, *полный одноразрядный сумматор* - это устройство, которое осуществляет сложение по *mod 2* соответствующих разрядов *(X1,X2)* двоичных чисел с учётом переноса *(Рm)* в данный разряд из соседнего младшего разряда суммы. Сумматор вырабатывает цифру результата (S) в данном разряде и перенос *(Рс)* в соседний старший разряд суммы. Таблица истинности такого сумматора (т.е. представление булевой функции, которую он реализует, в виде СДНФ) представлена ниже.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| X2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| PmТабл.1. Таблица истинности полного одноразрядного двоичного сумматора. | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| S | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| Pc | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |



Необходимо получить булевы функции *S=F1(X1,X2,Рm)* и *Рс=F2(X1,X2,Рm)*. Карты Карно для этих функций приведены ниже (рис.2).

Как следует из приведённых карт, МДНФ соответствующих функций имеет вид:

S=*Pm+**X*2+*X*1+ *X*1 *X*2 *Pm*

*(2)*

Pc= *X*1 *X*2+*X*1 *Pm+X*2 *Pm*

Полученная система булевых функций представлена в базисе И, ИЛИ, НЕ. Соответствующая ей КС приведена на рисунке 4.

Полученную комбинационную схему можно упростить, вынеся за скобки общие части в выражениях для S и Рc, однако существенного результата это не даст (желательно самостоятельно в этом убедиться).

Значительно упростить схему можно, если воспользоваться другим базисом, например логическим элементом "ИСКЛЮЧАЮЩЕЕ ИЛИ". В этом случае выражение для S можно записать S = (X1+X2+ Рm)mod2= X1⊕ X2⊕ Рm. Тогда схема для S будет иметь вид (рис.3).



Иногда для синтеза КС с несколькими выходами может использоваться следующий приём. Будем считать, что при синтезе схемы сумматора функция S является функцией четырёх переменных: S=f(*X*1,*X*2,*Рm*,*Рс*). Таблица истинности для этого случая принимает вид изображенный в таблице 2.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Pm | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Pc | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| S | 0 | X | 1 | X | 1 | X | X | 0 | 1 | X | X | 0 | X | 0 | X | 1 |

Таблица 2. Таблица истинности сумматора.

Неопределённые значения для S соответствуют наборам, которые никогда не могут быть в реальной схеме. Карта Карно для функции S=f(*X*1,*X*2,*Pm,Pc*) представлена на рис.5.



В результате минимизации, получается :

S=+*X*2+*X*1+ *X*1 *X*2 *Pm* = (*Pm*+*X*2+*X*1)+ *X*1 *X*2 *Pm*

*(3)*

Сравнивая выражения *(2)* и *(3)*, отмечаем, что функция S=f(*X*1,*X*2,*Pm*,*Pc*) проще, чем функция S=f1(*X*1,*X*2,*Pm*). Схему, соответствующую *(3)*, предлагается построить самостоятельно.

Т.о. задача синтеза имеет обычно несколько решений. Для сравнения различных вариантов комбинационных схем используют их основные характеристики: сложность и быстродействие.

**1.2. Характеристики комбинационных схем.**

***Сложность схемы*** оценивается количеством оборудования, составляющего схему. При разработке схем на основе конкретной элементной базы количество оборудования обычно измеряется числом корпусов (модулей) интегральных микросхем, используемых в схеме. В теоретических разработках ориентируются на произвольную элементную базу и поэтому для оценки затрат оборудования используется оценка сложности схем по Квайну.

***Сложность (цена)*** по Квайну определяется суммарным числом входов логических элементов в составе схемы.

При такой оценке единица сложности – один вход логического элемента. Цена инверсного входа обычно принимается равной двум. Такой подход к оценке сложности оправдан по следующим причинам:

1. сложность схемы легко вычисляется по булевым функциям, на основе которых строится схема: для ДНФ сложность схемы равна сумме количества букв,(букве со знаком отрицания соответствует цена 2), и количества знаков дизъюнкции, увеличенного на 1 для каждого дизъюнктивного выражения.
2. все классические методы минимизации булевых функций обеспечивают минимальность схемы именно в смысле цены по Квайну.

Практика показывает, что схема с минимальной ценой по Квайну обычно реализуется наименьшим числом конструктивных элементов – корпусов интегральных микросхем.

***Быстродействие комбинационной схемы*** оценивается максимальной задержкой сигнала при прохождении его от входа схемы к выходу, т.е. определяется промежутком времени от момента поступления входных сигналов до момента установления соответствующих значений выходных. Задержка сигнала кратна числу элементов, через которые проходит сигнал от входа к выходу схемы. Поэтому быстродействие схемы характеризуется значением *rτ*, где *τ* - задержка сигнала на одном элементе. Значение r определяется количеством уровней комбинационной схемы, которое рассчитывается следующим образом. Входам КС приписывается уровень ***нулевой***. Логические элементы, связанные только со входами схемы относятся к уровню ПЕРВОМУ. Элемент относится к уровню *k*, если он связан по входам с элементами уровней *k*-1, *k*-2, и т.д. Максимальный уровень элементов *r* определяет количество уровней КС, называемое ***рангом схемы***. Пример определения ранга r схемы приведён на рисунке 6.



Как известно, любая булева функция может быть представлена в ДНФ, которой соответствует двухуровневая комбинационная схема. Следовательно, быстродействие любой КС в принципе можно довести до 2τ.

Минимизация булевой функции с целью уменьшения сложности схем обычно приводит к необходимости представления функций в скобочной форме, которой соответствуют схемы с *r*>2. Т.е., уменьшение затрат оборудования в общем случае приводит к снижению быстродействия схем.

**1.3. Системы (серии) логических элементов и их**

**основные характеристики.**

При построении КС устройств вычислительной техники используются различные логические элементы, которые должны согласоваться по входным и выходным сигналам, напряжению питания и т.д. Для этой цели логические элементы объединяют в серии.

***Серией (системой, комплексом)*** логических элементов ЭВМ называется предназначенный для построения цифровых устройств функционально полный набор логических элементов, объединяемый общими электрическими, конструктивными и технологическими параметрами, использующий одинаковый способ представления информации, одинаковый тип межэлементных связей. Система элементов чаще всего избыточна по своему функциональному составу, что позволяет строить схемы более экономичные по количеству использованных элементов.

В состав серии входят элементы для выполнения логических операций, запоминающие элементы, элементы, реализующие функции узлов ЭВМ, а также специальные элементы для усиления, восстановления и формирования сигналов стандартной формы.

Конструктивно логические элементы представляют собой микроминиатюризованные интегральные электронные схемы (микросхемы), сформированные в кристалле кремния с помощью специальных технологических процессов.

В большинстве современных серий элементов имеются микросхемы малой степени интеграции (ИС до 100 элементов на кристалл), средней степени (СИС – до 1000 элементов на кристалл), большой степени интеграции (БИС – до 10000 элементов на кристалл) и сверхбольшой степени интеграции (СБИС – более 10000 элементов на кристалл). Логические элементы в виде ИС реализуют совокупность простых логических операций: *И, ИЛИ, И-ИЛИ, И-НЕ, ИЛИ-НЕ* и т.д. Логические элементы на СИС и БИС реализуют узлы ЭВМ, на СБИС – микроЭВМ.

*Основными параметрами серии логических элементов являются:*

 - питающие напряжения и сигналы для представления логического 0 и логической 1;

 - коэффициенты объединения по входу;

 - нагрузочная способность (коэффициент разветвления по выходу);

 - помехоустойчивость;

 - рассеиваемая мощность;

 - быстродействие.

Серия элементов характеризуется количеством используемых ***питающих напряжений*** и их номинальными значениями. Обычно логическому 0 соответствует низкий уровень напряжения, а логической 1 – высокий. Для наиболее часто используемых серий напряжение питания составляет +5В, уровень логической единицы 2,4-5В, уровень логического 0 – 0-0,4В.

***Коэффициент объединения по входу*** (Коб) определяет максимально возможное число входов логического элемента, иными словами, функцию скольких переменных может реализовать этот элемент. Обычно Коб принимает значение от 2 до 4, реже Коб=8. Увеличение числа входов связано с усложнением схемы элементов и приводит к ухудшению других параметров – помехоустойчивости, быстродействия и т.д.

***Коэффициент разветвления по выходу*** (Краз) показывает на сколько логических входов может быть одновременно нагружен выход данного логического элемента. Обычно Краз для наиболее часто используемых серий равен 10. Иногда вместо Краз задается предельно допустимое значение выходного тока логического элемента в состоянии 0 или 1.

***Помехоустойчивость*** – это способность элемента правильно функционировать при наличии помех. Она определяется максимально допустимым напряжением помехи, при котором не происходит сбоя в его работе. Обычно это напряжение порядка 0,6-0,9 В.

***Быстродействие*** логических элементов является одним из важнейших параметров и характеризуется временем задержки распространения сигнала. Этот параметр существенно зависит от технологии изготовления микросхем и лежит в диапазоне от единиц до сотен наносекунд.

Наиболее часто употребляемые типы интегральных микросхем – это потенциальные элементы транзисторно-транзисторной логики (ТТЛ) - серии К155, К555, К531, К1533 и т.д., транзисторной логики с эмиттерными связями (ЭСТЛ) – это серии К500,К1500, элементы на КМОП транзисторах - серии К176, К561,К564 и т.д.

При синтезе КС на реальных логических элементах необходимо обязательно учитывать ограничения на Коб и Краз.



**1.4. СИНТЕЗ КС С УЧЕТОМ ОГРАНИЧЕНИЙ НА .**

При построении КС может оказаться, что выход - го логического элемента нагружен  входов других ЛЭ (рис.7а). Это означает, что - тый логический элемент перегружен и необходимо принять меры, устраняющие указанное явление. Существуют два способа обеспечения заданного:

1. использование дополнительных развязывающих усилителей;
2. дублирование перегруженного элемента.

Схема с использованием дополнительных развязывающих усилителей представлена на рис.7.б. Количество p дополнительных усилителей, необходимых для обеспечения заданного , определяется по формуле:



 ***Недостаток рассматриваемого способа в том, что в цепь распространения сигнала вносится дополнительная задержка, что не всегда допустимо.***

 Схема с использованием дублирования перегружаемого элемента представлена на рис.7.в. Количество p дополнительных элементов, выполняющих ту же функцию, что и К-тый элемент, определяется по формуле:



 При таком способе обеспечения дополнительная задержка не вносится, но увеличивается нагрузка на элементы, формирующие сигналы  и , что может привести к перегрузке этих элементов и введению дополнительных элементов для обеспечения заданного Краз.

1.5. СИНТЕЗ КС С УЧЕТОМ ОГРАНИЧЕНИЯ НА .

Представлению функции в виде ДНФ соответствует двухуровневая КС (если считать, что на ее вход могут поступать как прямые так и инверсные входные сигналы), на первом уровне которой элементы И , а их выходы объединяются на втором уровне элементом ИЛИ . Такое построение КС обеспечивает ее максимальное быстродействие, так как ранг схемы минимален. Однако, не всегда возможно на первом уровне и, особенно, на втором выбрать логические элементы с требуемым, т.к. может оказаться, что ЛЭ с таким не выпускаются промышленностью. В этом случае необходимо с помощью нескольких элементов с меньшим получить эквивалент с большим  либо, что предпочтительней, преобразовать БФ, перейдя от ДНФ к скобочной форме. Этот переход сопровождается уменьшением логических элементов, требуемого для построения схемы. Осуществить такой переход можно с помощью *факторного алгоритма*, суть которого рассмотрим на примере.

Пусть задана некоторая булева функция в виде



Для реализации этой функции по приведенному выражению необходимо использовать 3 логических элемента 4И, один логический элемент 5И, один логический элемент 4ИЛИ.

С помощью факторного алгоритма получим скобочную форму для заданной функции. Для этого обозначим все конъюнкции буквами:

  

и будем рассматривать их как некоторые множества. Находим попарные пересечения множеств:

, , , , , .

Полученные пересечения показывают общие части отдельных конъюнкций. Выбираем пересечение, которое имеет наибольшую длину (если такое отсутствует, то выбирают то, которое чаще всего встречается). В данном случае это  . Поэтому из конъюнкций А и В выносим общую часть. Тогда имеем:

.

 Обозначим F =  и находим пересечения:

, , .

 Следовательно, для исходной функции имеем:

.

 Обозначим ,

Пересечение. Следовательно, окончательно имеем:





 Для реализации функции по последнему выражению необходимо 5 элементов 2И, 1 элемент 3И, 3 элемента 2ИЛИ ( рис.8 ).

Как видно из полученной схемы для ее реализации необходимы элементы с  = 2 или 3 (в отличие от исходной с  = 4 или 5). Однако ранг схемы увеличился до 7, что приводит к увеличению задержки срабатывания схемы.

1.6. Анализ комбинационных схем.

Задачи анализа КС возникают при необходимости проверить правильность синтеза (на этапе проектирования) или определить БФ, реализуемую КС (при анализе или ремонте схем). Все существующие методы анализа делятся на ***прямые*** и ***косвенные***.

В результате анализа КС ***прямым*** методом получается множество наборов входных переменных, обеспечивающих заданное значение на выходе, что позволяет записать в алгебраическом виде БФ, реализуемую схемой. К прямым методам относится метод π- алгоритма.

Применение ***косвенных*** методов дает возможность определить реакцию схемы на заданный набор входных переменных в статике или проанализировать переходный процесс смены одного входного набора на другой. Примерами косвенных методов анализа, являются методы синхронного и асинхронного моделирования.

Все упомянутые методы анализа являются машинoориентированными, что позволяет выполнить анализ схемы на ЭВМ.

Для всех методов анализа необходимо описать схему в виде схемного списка, в который включается в общем случае следующие данные: номер ЛЭ в схеме; логическая функция, реализуемая ЛЭ; входные переменные для данного ЛЭ. Например, схема представленная на рис.9, может быть описана следующим списком:

****

**1.7. Анализ комбинационных схем методом π-алгоритма.**

При данном методе, как упоминалось выше, ищутся наборы входных переменных, обеспечивающих заданное значение на выходе КС. Наборы, обеспечивающие на выходе КС логическую 1, образуют так называемое ***единичное покрытие*** . Аналогично, входные наборы, обеспечивающие на выходе КС логический 0, образуют ***нулевое покрытие*** . Рассмотрим покрытияи для простейшего логического элемента 2И, выполняющего функцию Y=X1X2. Таблица истинности для этой функции:

 Табл.3 Таблица истинности функции Y=X1X2



Как видно из приведенной таблицы только при единственном наборе X1=1 и X2=1 на выходе ЛЭ будет 1, т.е. единичное покрытие включает только один набор ={1 1}. На выходе ЛЭ будет 0 при трех наборах, образующих нулевое покрытие:



Это покрытие можно упростить, заметив, что первый набор склеивается со вторым и третьим, т.е.



Т.о. для ЛЭ 2И можно сказать, что 1 на его выходе будет только при обеих единицах на входах, а для обеспечения 0 на выходе достаточно подать хотя бы на один вход 0. Рассуждая аналогично, получим таблицу покрытий и для основных ЛЭ, представленных ниже в табл. 4.

Таблица 4.

ЛЭ Y Y Y Y Y Y Y

 НЕ 2И 2И – НЕ 2ИЛИ 2ИЛИ–НЕ ИСК. ИЛИ 3И – НЕ

 X X1 X2 X1 X2 X1  X2 X1 X2 X1 X2 X1 X2 X3

1

X

&

X1

X2

&

X1

X2

1

X1

X2

1

X1

X2

=1

X1

X2

&

X1

X2

X3

 1 0 X 1 1 0 0 1 X 0 0 1 1 1

 X 0 X 1 1 1

 0 1 1 0 X 1 X 0 0 0 1 0 X X

 X 0 X 1 1 0 X 0 X

 X X 0

При анализе схемы методом π - алгоритма, задавшись определенным значением на выходе, заменяют его соответствующим покрытием элемента, формирующего выходной сигнал. В результате этого определяется, какие должны быть сигналы на выходах элементов, подключенных к выходному ЛЭ. В свою очередь, сигналы на выходах этих элементов можно заменить соответствующими покрытиями, т.е. определить значения выходных сигналов для других ЛЭ и т.д. Этот процесс продолжается до тех пор, пока не получатся покрытия, состоящие только из входных переменных, называемых опорными. Совокупность таких покрытий и дает соответствующее покрытие схемы.

Пример анализа КС (рис 9. ) методом π - алгоритма представлен в табл. 5. В последней колонке этой таблицы приведен оператор подстановки, в результате работы которого сигнал на выходе ЛЭ заменяется соответствующим покрытием. Необходимо обратить внимание, что все значения переменных, записанные в одной строке, должны одновременно быть в наличии для обеспечения заданного значения выходного сигнала. По-

этому, при замене одного из значений в строке соответствующим покрытием, все остальные значения для других переменных в этой строке должны присутствовать совместно с этим покрытием.

На основании полученного единичного покрытия можно записать БФ, реализуемую схемой:



Таблица 5 Анализ схемы методом **π** – алгоритма.



а) Получение первого покрытия



б) Получение нулевого покрытия 

В дальнейшем можно сравнить полученную БФ с той, по которой строилась схема и проверить правильность ее построения. При анализе схемы может оказаться, что некоторая переменная, получившая на одном из предыдущих шагов некоторые значения на данном шаге должна принять противоположное значение. Возникшее противоречие говорит о том, что данный путь является тупиковым и его необходимо исключить из дальнейшего рассмотрения. Если ни при одной комбинации входных переменных не обеспечивается значение 1(0) на выходе, то это означает, что схема реализует константу 0(1) соответственно.

**1. 8 Анализ КС методом синхронного моделирования.**

При данном методе считается, что все ЛЭ переключаются одновременно, без задержки. В результате применения метода определяется установившееся значение сигнала на выходе схемы.

Рассмотрим метод синхронного моделирования на примере схемы ( рис.9 ).

На первом этапе схему разбиваем на уровни и записываем в порядке возрастания уровня уравнения, описывающие функционирование ЛЭ:

|  |  |  |
| --- | --- | --- |
| №уровня | №элемента | уравнение |
| 1 | 12 | e1 = X1 X2e2 =  |
| 2 | 3 | e3 =  |
| 3 | 4 | Y = e4 = e3 + X5 |

Проанализируем схему при подаче на вход набора X1=0, Х2=0, Х3=0, Х4=1, Х5=1. Для этого решаем записанные уравнения в порядке возрастания уравнения. Имеем:

;

;

;

.

Следовательно, при подаче на вход набора {00011}, на выходе будет Y=1. Аналогично можно промоделировать работу схемы при подаче на вход любого другого набора.

**1.9 Анализ КС методом асинхронного моделирования.**

Реальный ЛЭ переключается за какое-то конечное время, зависящее от технологии изготовления, условий эксплуатации, емкостей нагрузки и т.д. Прохождение сигнала последовательно через несколько ЛЭ будет приводить к накоплению времени задержки и возникновению сдвига во времени выходного сигнала по отношению ко входному. Наличие задержки и порождаемого ею временного сдвига сигналов может приводить к появлению на выходе отдельных ЛЭ и всей схемы в целом кратковременных сигналов, не предусмотренных БФ, реализуемой схемой. Как иллюстрацию, рассмотрим схему рис.11, а .



Рис. 11 а)



Рис. 11 . Статический риск сбоя.

а)- схема, б)- временные диаграммы.

t1-время задержки инвертора

t2-время задержки элемента 2И

Данная схема реализует функцию , т.е. константу 0 независимо от входного сигнала X. Однако в переходном процессе в результате задержки срабатывания ЛЭ возможна ситуация, когда на обоих входах элемента 2И будут логические единицы, что может привести к появлению на выходе схемы логической 1 (см. рис.11 б). Рассмотренный случай возможен при задержке срабатывания второго элемента больше, чем первого. Такое явление называется ***риском сбоя***. Различают статистический и динамический риски сбоя.

При ***статическом*** риске сбоя до и после переходного процесса состояние выходного сигнала одно и то же, а во время переходного процесса возможно кратковременное появление противоположного сигнала.

При ***динамическом*** риске сбоя до и после переходного процесса состояния выходного сигнала противоположные, но в переходном процессе выходной сигнал несколько раз меняет свое значение. Динамический риск сбоя возможен в схеме (рис.12 а) при смене набора (Х1=0, Х2=1, Х3=1) на набор (Х1=1, Х2=0, Х3=0) и иллюстрируется диаграммами (рис.12 б).

В данном примере динамический риск сбоя на выходе КС сопровождается статическим на выходе элемента 1. Как видно из временных диаграмм риск сбоя имеет место при наличии определенного временного сдвига между сигналами, поступающими на вход ЛЭ. Нежелательные сигналы на выходе могут и отсутствовать при другом соотношении временных сигналов, однако принципиальная возможность их появления является фактором снижающим надежность работы схемы. Поэтому очень важно уметь обнаруживать и устранять такие явления.



Для анализа процесса переключения КС при смене входных наборов и обнаружения рисков сбоя используется метод ***асинхронного моделирования***. При этом методе считается, что каждый элемент переключается с одинаковой задержкой. Анализ включает такие этапы:

1.Каждому элементу схемы присваивается уровень, причем уровень 1 имеют элементы, все входы которых являются независимыми входами схемы.

2.Записываются уравнения, описывающие каждый ЛЭ в порядке убывания уровня.

3.Для исходного входного набора А(X1, X2, … , Xn ) определяется значение сигналов на выходах всех ЛЭ схемы. Пусть данный набор А заменяется набором В(X1, X2, … , Xn ).

4.Помечаются те уравнения, в правой части которых хотя бы одна из переменных изменила свое значение.

5.Решаются помеченные уравнения в порядке их записи в схеме . После решения уравнение считается непомеченным.

6.Если после решения всех уравнений системы переменные, входящие в левые части уравнений, изменили свои значения, то вновь помечаются те уравнения, в правые части которых входят эти переменные. Затем осуществляется переход к п.5. В противном случае моделирование данного входного набора считается законченным. Выполнение п.5 называется ***тактом моделирования.***

Анализ схемы (рис.13) методом асинхронного моделирования приведен ниже. Для данной схемы входной набор А(1011110) заменяется набором В(1101011).



Рис. 13. Комбинационная схема для метода асинхронного моделирования.

Уравнения, описывающие ЛЭ:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1-ый такт | 2-ой такт | 3-ий такт |
| Y= e6 = e4 + e5 + X5e5 = e3 X7e3=X5 X6e2=X5 X4e1=X1 X2 | \*\*-\*\*\* | \*\*\*--- | \*----- |

 Табл. 6 Таблица моделирования схемы

 Выходы Такты моделирования Прим.

 0 1 2 3

 e6 1 0 1 0 дин.

 e5 0 1 0 0 стат.

 e4 0 0 0 0

 e3  1 0 0 0

 e2 1 0 0 0

 e1 0 1 0 1

Как следует из результатов моделирования, при смене набора А набором В на выходе элемента 4 имеет место статический риск сбоя, а на выходе схемы - динамический риск сбоя.

Радикальным способом устранения рисков сбоя является введение стробирования для снятия выходного сигнала КС. Стробирующий импульс подается после окончания переходного процесса в КС (т.е. когда на выходе КС уже установилось необходимое значение выходного сигнала), что исключает влияние возможных сбоев на вырабатываемый схемой сигнал.

**ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ТЕОРИИ АБСТРАKТНЫХ АВТО­МАТОВ.**

Ознакомившись в курсах "Программирование" и "Машинная арифметика" с принципами работы ЭЦВМ, можно ука­зать на две основные особенности таких вычислительных машин: оперирование данными, представленными в цифровой форме и автоматическая работа по заранее составленной программе. Эти особенности показывают, что любая ЭЦВМ является цифровым автоматом (ЦА). Понятие ЦА служит обобщением для всех видов устройств обработки цифровой информации, имеющих программное управление.

Цифровой автомат - устройство, характеризующееся набором внутренних состояний в которое оно попадет под воз­действием команд заложенной в него программы. Переход автомата из одного состояния в другое осуществляется в определенный момент времени.

Математической моделью ЦА (а в общем случае любого дискретного устройства) является так называемый абстракт­ный автомат, определенный как 6-компонентный кортеж: S=(A,Z,W,δ,λ,а1) у которого:

1. A={a1, a2, ... ,am} - множество состояний (внутренний алфавит)

2. Z={z1, z2, ... ,zf} - множество входных сигналов (входной алфавит)

3. W={w1, w2, ..., wg} - множество выходных сигналов (выходной алфавит)

4. δ **:** A•Z→A - функция переходов, реализующая отображение Dδ **⊆**А•Z в А. Иными словами функция δ некоторым парам состояние - входной сигнал (аm, zf) ставит в соответствие состояния автомата аs= δ (am, zf), as**∈**A.

5. λ **:** A•Z→W - функция выходов, реализующая отображение Dλ **⊆**А•Z на W. Функция λ некоторым парам состояние - входной сигнал (аm, zf) ставит в соответствие выходные сигналы автомата Wg=λ(аm, zf) , Wg**∈**W.

6. ai **∈**A - начальное состояние автомата.

Под ***алфавитом*** здесь понимается непустое множество попарно различных символов. ***Элементы алфавита*** называются ***буквами***, а ***конечная упорядоченная последовательность букв - словом*** в данном алфавите.



Абстрактный автомат (рис.14) имеет один вход и один выход. Автомат работает в дискретном времени, принимающем целые неотрицательные значения t = 0,1,2,... В каждый момент t дискретного времени автомат находится в некотором состоянии a(t) из множества состояний автомата, причем в начальный момент t = 0 он всегда находится в начальном со­стоянии a(0)=a1. В момент t, будучи в состоянии a(t), автомат способен воспринять на входе букву входного алфавита z(t) **∈** Z. В соответствии с функцией выходов он выдаст в тот же момент времени t букву выходного алфавита W(t)=λ(a(t), z(t)) и в соответствии с функцией переходов перейдет в следующее состояние a(t+1)=δ[a(t), z(t)], a(t) **∈**A, w(t) **∈**W.

Смысл понятия абстрактного автомата состоит в том, что он реализует некоторое отображение множества слов вход­ного алфавита Z во множество слов выходного алфавита W. Т.е. если на вход автомата, установленного в начальное состояние a1, подавать буква за буквой некоторую последовательность букв входного алфавита z(0), z(1),... - входное слово, то на выходе автомата будут последовательно появляться буквы выходного алфавита w(0), w(1),... - выходное слово. Т.о. выходное слово = ϕ(входное слово), где ϕ - отображение, осуществляемое абстрактным автоматом.

На уровне абстрактной теории понятие "работа автомата" понимается как преобразование входных слов в выходные. Можно сказать, что в абстрактном автомате отвлекаемся от его структуры - содержимого прямоугольника (рис. 14 ), рассматривая его как "черный ящик", т.е. основное внимание уделяем поведению автомата относительно внешней среды.

Понятие состояния в определении автомата введено в связи с тем, что часто возникает необходимость в описании пове­дения систем, выходы которых зависят не только от состояния входов в данный момент времени, но и от некоторой пре­дыстории, т.е. от сигналов, которые поступали на входы системы ранее. Состояния как раз и соответствуют некоторой памяти о прошлом, позволяя устранить время как явную переменную и выразить выходной сигнал как функцию со­стояния и входа в данный момент времени.

На практике наибольшее распространение получили два класса автоматов - автоматы Мили (Mealy) и Мура (Moore).

Закон функционирования автомата Мили задается уравнениями:

a(t+1) = δ(a(t), z(t)); w(t) = λ(a(t), z(t)), t = 0,1,2,...

Закон функционирования автомата Мура задается уравнениями:

a(t+1)=δ(a(t), z(t)); w(t) = λ(a(t)), t = 0,1,2,...

Из сравнения законов функционирования видно, что, в отличие от автомата Мили, выходной сигнал в автомате Мура зависит только от текущего состояния автомата и в явном виде не зависит от входного сигнала. Для полного задания автомата Мили или Мура дополнительно к законам функционирования, необходимо указать начальное состояние и оп­ределить внутренний, входной и выходной алфавиты.

Кроме автоматов Мили и Мура иногда оказывается удобным пользоваться совмещенной моделью автомата, так на­зываемым С- автоматом.

Под абстрактным С- автоматом будем понимать математическую модель дискретного устройства, определяемую восьми­компонентным вектором S=( A, Z, W, U, δ, λ1, λ2, а1 ), у которого:

1. A={a1, a2, ... ,am} - множество состояний;

2. Z={z1, z2, ... ,zf} - входной алфавит;

3. W={w1, w2, ..., wg} - выходной алфавит типа 1;

4. U={u1, u2,...,uh} - выходной алфавит типа 2;

5. δ **:** A • Z → A - функция переходов, реализующая отображение Dδ **⊆**А•Z в А;

6. λ1 **:** A • Z → W - функция выходов, реализующая отображение Dλ1 **⊆**А•Z в W;

7. λ2 **:** A → U - функция выходов, реализующая отображение Dλ2 **⊆** А в U;

8. а1 ∈ А - начальное состояние автомата.

Абстрактный С- автомат можно представить в виде устройства с одним входом, на который поступают сигналы из входного алфавита Z, и двумя выходами, на которых появляются сигналы из алфавитов W и U. Отличие С - автомата от моделей Мили и Мура состоит в том, что он одновременно реализует две функции выходов λ1 и λ2, каждая из которых характерна для этих моделей в отдельности. Закон функционирования С- автомата можно описать следующими уравнениями:

а( t + 1) = δ( a( t ), z( t )); w( t ) = λ1( a ( t ), z( t )); u( t ) = λ2( a( t )); t = 0, 1, 2, ...

Выходной сигнал Uh=λ2( am ) выдается все время, пока автомат находится в состоянии am. Выходной сигнал Wg=λ1( am, zf ) выдается во время действия входного сигнала Zf при нахождении автомата в состоянии am.

Рассмотренные выше абстрактные автоматы можно разделить на:

1. полностью определенные и частичные;
2. детерминированные и вероятностные;
3. синхронные и асинхронные;

***Полностью определенным*** называется абстрактный цифровой автомат, у которого функция переходов и функция выходов определены для всех пар ( ai, zj ).

***Частичным*** называется абстрактный автомат, у которого функция переходов или функция выходов, или обе эти функ­ции определены не для всех пар ( ai, zj ).

К ***детерминированным*** относятся автоматы, у которых выполнено условие однозначности переходов : автомат, находя­щийся в некотором состоянии ai, под действием любого входного сигнала zj не может перейти более, чем в одно состоя­ние.

В противном случае это будет ***вероятностный автомат***, в котором при заданном состоянии ai и заданном входном сиг­нале zj возможен переход с заданной вероятностью в различные состояния.

Для определения ***синхронных и асинхронных*** автоматов вводится понятие ***устойчивого состояния***. Состояние as автомата называется устойчивым, если для любого состояния ai и входного сигнала zj таких, что δ( ai, zj ) = as имеет место δ( as, zj ) = as, т.е. состояние устойчиво, если попав в это состояние под действием некоторого сиг­нала zj, автомат выйдет из него только под действием другого сигнала zk, отличного от zj.

Автомат, у которого все состояния устойчивы ***- асинхронный***.

Автомат называется ***синхронным***, если он не является асинхронным.

Абстрактный автомат называется ***конечным***, если конечны множества А = {a1, a2, ..., am}, Z = {z1, z2, ..., zf}, W = {w1, w2, ..., wg}. Автомат носит название ***инициального***, если в нем выделено начальное состояние a1.

**СПОСОБЫ ОПИСАНИЯ И ЗАДАНИЯ АВТОМАТОВ.**

Для того, чтобы задать автомат, необходимо описать все компоненты кортежа S=(A, Z, W, δ, λ, а1 ). Множества А, Z, W описываются и задаются простым перечислением своих элементов. Среди многообразия различных способов заданий функций δ и λ ( следовательно и всего автомата в целом ) наибольшее распространение получили табличный и графиче­ский.

При табличном способе задания автомат Мили описывается с помощью двух таблиц. Одна из них (таблица переходов ) задает функцию δ, т.е. a( t +1) = δ( a( t ), z( t )) ( табл.7), вторая (таблица выходов ) - функцию λ, т.е. W( t )=λ( a( t ), z( t )) ( табл. 8 ).



Каждому столбцу из приведенных таблиц поставлено в соответствие одно состояние из множества А, каждой строке - один входной сигнал из множества Z. На пересечении столбца am и строки zf в табл.7 записывается состояние as, в которое должен перейти автомат из состояния am под действием входного сигнала Zf, т.е. as = δ(am, zf). На пересечении столбца am и строки zf в табл.8 записывается выходной сигнал Wg, выдаваемый автоматом в состоянии am при поступ­лении на вход сигнала zf, т.е. Wg = λ( am, zf ).

Для приведенных таблиц множества, образующие автомат A={a1, a2, a3,a4}, Z={z1, z2}, W={w1, w2, w3, w4, w5}. Автомат Мили может быть задан одной совмещенной таблицей переходов и выходов (табл.9), в которой каждый элемент as / wg записанный на пересечении столбца am и строки zf, определяется следующим образом:



as=δ(am, zf); wf=λ(am, zf).

Автомат Мура задается одной отмеченной таблицей переходов (табл.10), в которой каждому столбцу приписаны не только состояние am , но еще и выходной сигнал Wg, соответствующий этому состоянию, где Wg=λ(am).



Для частичных автоматов Мили и Мура в рассмотренных таблицах на месте не определенных состояний и выходных сигналов ставится прочерк. В таких автоматах выходной сигнал на каком-либо переходе всегда не определен, если неопределенным является состояние перехода. Кроме того, выходной сигнал может быть неопределенным и для некоторых существующих переходов.

Для задания С - автоматов также используется табличный метод. В этом случае таблица переходов (табл.11) аналогична таблице переходов автомата Мили, а в таблице выходов каждое состояние отмечено соответствующим выходным сигналом ui выходного алфавита типа 2 (табл.12)



При графическом способе автомат задается в виде ориентированного графа, вершины которого соответствуют состояниям, а дуги - переходам между ними. Дуга, направленная из вершины am, задает переход в автомате из состояния am в состояние as. В начале этой дуги записывается входной сигнал Zf∈Z, вызывающий данный переход as=δ(am,zf). Для графа автомата Мили выходной сигнал wg∈W, формируемый на переходе, записывается в конце дуги, а для автомата Мура - рядом с вершиной am, отмеченной состоянием am, в котором он формируется. Если переход в автомате из состояния am в состояние as производится под действием нескольких входных сигналов, то дуге графа, направленной из am в as, приписываются все эти входные и соответствующие выходные сигналы. Граф С- автомата содержит выходные сигналы двух типов и они обозначаются на графе как на графах соответствующих автоматов. Графы автоматов, заданных своими таблицами переходов и выходов
(табл. 7÷12) представлены на рисунках 15,16,17.





**СВЯЗЬ *МЕЖДУ МОДЕЛЯМИ МИЛИ И МУРА.***

Рассмотрим некоторый автомат Мили, заданный таблицами переходов и выходов. Таблица переходов а) и выходов б) автомата Мили



Подадим на вход автомата, установленного в состояние а1, входное слово ξ=z1 z2 z2 z1 z2 z2. Так как δ(а1, z1) = a2, λ(a1, z1) = W2, то под воздействием входного сигнала z1 автомат перейдет в состояние а2 и выдаст на переходе выходной сигнал W2. Затем, находясь в состоянии а2 под воздействием сигнала Z2 перейдет в состояние а1=δ(а2, z2) и выдаст сигнал W1=λ(a2, z2) и т.д. В табл. 13 приведена последовательность состояний, которые автомат проходит, воспринимая входное слово ξ, и выходные сигналы, вырабатываемые на этих переходах.



Назовем выходное слово ω = λ (a1, ξ) *реакцией* автомата Мили в состоянии а1 на входное слово ξ.

В нашем случае ω = w2 w1 w2 w2 w1 w2

Как видно, из приведенного примера, в ответ на входное слово длины k автомат Мили выдаст последовательность состояний длины k +1 и выходное слово длины k.



В общем виде поведение автомата Мили, установленного в состояние am, можно описать следующим образом (табл. 14).

Аналогично можно описать поведение автомата Мура, находящегося в состоянии a1, при приходе входного слова

ξ = Zi1, Zi2, . . . , Zik ,учитывая, что W( t ) = λ(a( t )):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Входное слово | Zi1 | Zi2 | Zi3 | Z |
| Последовательность cостояний | am | ai2 = δ (am, Zi1 ) | ai3 = δ (ai2, Zi2 ) | ai4 = δ(ai3, Zi3 ) |
| Выходное слово | wi1 = λ (am, Zi1) | wi2 = λ (ai2, Zi2) | wi3 = λ (ai3, Zi3) | wi4 = λ (ai4) |

Очевидно, что для автомата Мура выходной сигнал Wi1= λ(am) в момент времени i1 не зависит от входного сигнала Zi1 и определяется только состоянием am. Следовательно, сигнал Wi1 никак не связан с входным словом ξ.

В связи с этим под реакцией автомата Мура, установленного в состояние am, на входное слово ξ = Zi1, Zi2, . . . , Zik  будем понимать выходное слово той же длины ω = λ(am, ξ) = Wi2 Wi3 ...Wik+1, сдвинутое по отношению к ξ на один такт.

Рассмотрим пример. Пусть задан автомат Мура:

Подадим на вход этого автомата ту же последовательность, что и для автомата Мили: ξ=z1 z2 z2 z1 z2 z2. Последовательность смены состояний и вырабатываемых выходных сигналов представлена в таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | w1 | w2 | w3 | w4 |
|  | a1 | a2 | a3 | a4 |
| z1 | a2 | a3 | a4 | a4 |
| z1 | a4 | a1 | a1 | a1 |



Сравнивая реакции автомата Мили (табл. 13) и автомата Мура (табл.15), отмечаем, что эти реакции на одно и то же слово ξ совпадают. Следовательно автоматы Мили и Мура реализуют одно и то же преобразование слов входного алфавита. Такие автоматы называются эквивалентными. Строгое определение эквивалентности следующее:

***два автомата с одинаковыми входными и выходными алфавитами называются эквивалентными, если после установки их в начальное состояние их реакции на любое входное слово совпадают.***

*Для каждого автомата Мили может быть построен эквивалентный ему автомат Мура и наоборот.*

Переход от автомата Мура к эквивалентному ему автомату Мили тривиален и легко осуществляется при графическом способе задания автомата. Для получения графа автомата Мили необходимо выходной сигнал Wg, записанный рядом с вершиной as исходного автомата Мура, перенести на все дуги, входящие в эту вершину. На рис. 18 приведен граф автомата Мили, эквивалентного автомату Мура (рис. 16)



Легко убедиться, что полученный автомат Мили действительно эквивалентный исходному автомату Мура. Для этого достаточно рассмотреть реакцию обеих автоматов на произвольную входную последовательность, что предлагается выполнить самостоятельно. Необходимо также отметить, что в эквивалентном автомате Мили количество состояний такое же, как и в исходном автомате Мура.

Переход от автомата Мили к эквивалентному ему автомату Мура более сложен. Это связано с тем, что в автомате Мура в каждом состоянии вырабатывается только один выходной сигнал. Как и в предыдущем случае, переход наиболее наглядно делать при графическом способе задания автомата. В этом случае каждое состояние ai исходного автомата Мили порождает столько состояний автомата Мура, сколько различных выходных сигналов вырабатывается в исходном автомате при попадании в состояние ai. Рассмотрим переход от автомата Мили Sa к автомату Мура Sb на примере автомата (рис. 15).

Как следует из рис. 15 для автомата Sa при попадании в состояние а1 вырабатываются выходные сигналы W2, W4, W5, при попадании в а2 – W1, W2, a3 – W2, a4 – W3. Каждой паре состояние ai - выходной сигнал Wj, который вырабатывается при попадании в это состояние, поставим в соответствие состояние bk эквивалентного автомата Мура Sb с тем же выходным сигналом Wj : b1 = (a1, W2), b2 = (a1, W4), b3 = (a1, W5), b4 = (a2, W1), b5 = (a2, W2), b6 =(a3, W2), b7 = (a4, W3), т.е. каждое состояние ai автомата Мили порождает некоторое множество Ai состояний эквивалентного автомата Мура: A1 = { b1, b­2, b3 }, A2 = { b4, b5 }, A3 = { b6 }, A4 = { b7 }. Как видно, в эквивалентном автомате Мура количество состояний 7. Для построения графа Sb поступаем следующим образом. Т.к. в автомате Sa есть переход из состояния а1 в состояние а2 под действием сигнала z1 с выдачей W1, то из множества состояний A1 = {b1, b2, b3}, порождаемых состоянием а1 автомата Sa в автомате Sb должен быть переход в состояние (a3, W2) = b6 под действием сигнала Z2 и т.д. Граф эквивалентного автомата Мура представлен на рис.19.



Если от автомата Мура Sb, эквивалентного автомату Мили Sa (рис. 19) вновь перейти к автомату Мили, то получим автомат Мили S1 (рис. 20)



Вследствие транзитивности отношения эквивалентности два автомата Мили S1 и Sa также будут эквивалентными, но у последнего будут на 3 состояния больше. Т.о., эквивалентные между собой автоматы могут иметь различное число состояний, в связи с чем возникает задача нахождения минимального (т.е. с минимальным числом состояний) автомата в классе эквивалентных между собой автоматов.

**МИНИМИЗАЦИЯ ЧИСЛА ВНУТРЕННИХ СОСТОЯНИЙ ПОЛНОСТЬЮ ОПРЕДЕЛЕННЫХ АВТОМАТОВ.**

Рассмотрим метод минимизации полностью определенных автоматов, предложенный Ауфенкампом и Хоном.

Основная идея этого метода заключается в разбиении всех состояний исходного абстрактного автомата на попарно непересекающиеся классы эквивалентных состояний и замене каждого класса эквивалентности одним состоянием. Т.о. получающийся в результате минимальный автомат имеет столько состояний, на сколько классов эквивалентности разбиваются состояния исходного автомата.

Для пользования методом введем несколько определений.

Два состояния абстрактного автомата называются 1-эквивалентными в том случае, если реакции автомата в этих состояниях на всевозможные входные слова совпадают.

Объединение всех 1-эквивалентных состояний абстрактного автомата образует 1-класс эквивалентности.

1-эквивалентные состояния автомата называются 2-эквивалентными, если они переводятся любым входным сигналом также в 1-эквивалентные состояния.

Объединение всех 2-эквивалентных состояний образует 2-класс эквивалентности.

По индукции можно распространить определение до i-эквивалентных состояний и i-классов эквивалентности.

Если для некоторого i разбиения состояний автомата на ( i +1) - классы совпадает с разбиением на i-классы, то оно является разбиением и на ∞ - классы эквивалентности.

Разбиение множества внутренних состояний автомата на ∞ - классы и является требуемым разбиением на классы эквивалентности, при этом такое разбиение может быть получено за конечное число шагов.

Все вышеизложенное непосредственно применимо к минимизации автомата Мили. При минимизации полностью определенных автоматов Мура вводится понятие 0-эквивалентности состояний и разбиение множества состояний на 0-эквивалентные классы: к такому классу относятся одинаково отмеченные состояния автомата Мура.

Если два 0-эквивалентных состояния любым входным сигналом переводится в два 0-эквивалентных состояния, то они называются 1-эквивалентными. Все дальнейшие классы эквивалентности состояний для автомата Мура определяются аналогично приведенному для автоматов Мили.



Рассмотрим пример минимизации автомата Мили, заданного таблицами переходов и выходов :

Из таблицы выходов получаем разбиение на 1-классы эквивалентности π1, объединяя в эквивалентные классы Bi состояния с одинаковыми столбцами:

π1 = {B1, B2}; B2 = {a1, a2, a5, a6}; B2 = {a3, a4}



Для получения 2-эквивалентных состояний строим таблицу 1-разбиения (табл.17), заменяя в таблице переходов состояния a1 соответствующими классами эквивалентности B1 или B2.

Из полученной таблицы 1-разбиения получаем 2-классы эквивалентности Ci и разбиение π2 = {C1, C2, C3}, где С1 = {a1, a1}, C2 = {a5, a6}, C3 = {a3, a4}. Сравнивая π2 и π1, отмечаем, что эти разбиения отличаются друг от друга. Поэтому аналогично строим таблицу 2-разбиения (табл. 18), опять заменяя в таблице переходов состояния ai соответствующими классами эквивалентности Ci.



Из полученной таблицы 2-разбиения получаем 3-классы эквивалентности Di и разбиение π3 ={ D1, D2, D3}, где D1 = {a1, a2}, D2 = {a5, a6}, D3 = {a3, a4}.



Сравнивая π3 и π2, замечаем, что D1 = C1, D2 = C2, D3 = C3, π3 = π3. Следовательно получили разбиение на ∞- эквивалентные классы. Т.к. всего три таких класса, то минимальный автомат будет содержать всего три состояния. Выбираем из каждого класса Di по одному состоянию и получаем множество состояний A' минимального автомата. Пусть, например, A'={a1, a4, a5}. Для получения минимального автомата из первоначальных таблиц переходов и выходов (табл. 16) вычеркиваем столбцы, соответствующие "лишним состояниям" a2, a3, a6. В результате получается минимальный автомат Мили, эквивалентный исходному автомату (табл. 19).

Минимизацией числа внутренних состояний автомата заканчивается этап абстрактного синтеза.

**Структурный синтез ЦА.**

***Задачи синтеза ЦА.***

***Канонический метод структурного синтеза ЦА.***

***Элементарные цифровые автоматы с памятью***

***(триггерные устройства) и их свойства.***

Вслед за этапом абстрактного синтеза автоматов следует этап структурного синтеза, целью которого является построение схемы, реализующей автомат из элементов заданного типа. Если абстрактный автомат был лишь математической моделью, проектируемого устройства, то в структурном автомате учитывается структура входных и выходных сигналов автомата, а также его внутренне устройство на уровне логических схем. *Основной задачей структурной теории автоматов является разработка общих методов построения структурных схем автоматов.*

 В отличие от абстрактного автомата, имеющего один вход и один выход, на которые поступают сигналы во входном и выходят в выходном *W={W1,..,WG}* алфавитах, структурный автомат имеет *L* входных каналов *х1,х2,..,хL* и *N* выходных *y1,y2,…,yN* на каждом из которых присутствует сигнал структурного алфавита.



Обычно в качестве структурного используется двоичный алфавит.

В этом случае каждому входному сигналу *ZF* абстрактного автомата соответствует некоторый двоичный вектор *(lf1,lf2,..,lfL)*, где *lfL*∈{0,1}.

Очевидно, что для представления (кодирования) входных сигналов *Z1,..,ZF* абстрактного автомата различными двоичными векторами должно быть выполнено условие

*L  ] log2F [*,

аналогично

*N  ] log2G [*

Например , *Z={Z1,Z2,Z3,Z4} W={W1,W2,W3}*.

Тогда *L*  *log2*4=2 *N  log2*3=2

Закодировать входные и выходные сигналы можно ,например, так:

*Z1* = 00 *W1* = 00

*Z2* = 01 *W2* = 01

*Z3* = 10 *W3* = 11



*Z4* = 11

Cледовательно, структурный автомат с двумя входами *x1* и *x2* и двумя выходами *y1* и *y2* может быть представлен в виде:

**Задача синтеза структуры автомата.**

На этапе структурного синтеза предварительно выбираются элементарные автоматы, путем композиции которых строят логические схемы полученных на этапе абстрактного синтеза автоматов Мили и Мура. Если решение задачи структурного синтеза существует, говорят, что заданная система автоматов *структурно полна*.

Рассмотрим *канонический метод структурного синтеза* при котором используются элементарные автоматы некоторого специального вида – автоматы с памятью, имеющие более одного состояния, и автоматы без памяти – с одним состоянием. Первые автоматы называются ***элементами*** ***памяти***, вторые – ***комбинационные или логические элементы***.

Теоретическим обоснованием канонического метода структурного синтеза автоматов служит ***теорема о структурной полноте:***



Для правильной работы схем сигналы на входе запоминающих элементов не должны непосредственно участвовать в образовании выходных сигналов, которые по цепям обратной связи подавались бы в тот же самый момент времени на эти входы. Поэтому запоминающими элементами должны быть не автоматы Мили, а автоматы Мура. Таким образом, структурно полная система элементарных автоматов должна содержать хотя бы один автомат Мура. В то же время, для синтеза автоматов с минимальным числом элементов памяти, необходимо в качестве таких элементов выбирать автоматы Мура, имеющие полную систему переходов и полную систему выходов – полные автоматы.

***Полнота системы переходов*** означает, что **для любой** упорядоченной *пары состояний* автомата *найдётся входной сигнал, переводящий первый элемент этой пары во второй,* т.е в таком автомате в каждом столбце таблицы переходов должны встречаться все состояния автомата.

***Полнота системы выходов*** автомата Мура состоит в том, что *каждому состоянию автомата поставлен в соответствие свой особый выходной сигнал*, отличный от выходных сигналов других состояний. Т.о. в таком автомате число выходных сигналов равно числу состояний автомата. В связи с этим, в автоматах памяти будем использовать одни и те же обозначения и для состояний, и для выходных сигналов.

Канонический метод структурного синтеза предполагает представление структурной схемы автомата в виде двух частей: памяти и комбинационной схемы.

Память состоит из элементарных автоматов Мура П1,....,ПZ,....,ПR. После выбора элементов памяти каждое состояние синтезируемого автомата *А* кодируется набором их состояний. Если все автоматы П1...,ПR одинаковы, что в общем случае необязательно, то их число



где *M –* число состояний синтезируемого автомата *А*, а *b* – число состояний элементарного автомата памяти. Обычно для элементарного автомата *b*=2, тогда .

Например, переход автомата *А*, имеющего 5 элементов памяти, алфавит состояний которых – двоичный, из одного состояния (*A*m)=01011 в другое (*A*3)= 11000, заключается в изменении состояний соответствующих автоматов памяти: первый элемент памяти переходит из 0 в 1, второй – из 1 в 1, третий из 0 в 0, четвёртый – из 1 в 0, пятый - из 1 в 0.

Переходы автоматов памяти, соответствующие переходам в автомате *А*, происходят под действием сигналов возбуждения памяти, поступающих с выхода комбинационной схемы на вход памяти автомата. Так на рисунке *X*=(*X*1,*X*2,..,*X*L) и *Y*=(*Y*1,*Y*2,...,*Y*N) – векторные структурные входной и выходной сигналы автомата, *U*=(*U*1,*U*2,...,*U*T) – векторная функция возбуждения памяти и *Q*=(*Q*1,...,*Q*T) – вектор выходного сигнала обратной связи от элементов памяти автомата.

Рассмотрим отдельно элемент памяти Пz, таблица переходов которого дана в таблице. Множество выходных сигналов элементов памяти совпадает с множеством внутренних состояний.



Полнота переходов очевидна из таблицы (в каждом столбце все состояния встречаются). При рассмотрении автомата на абстрактном уровне его можно представить в виде рис.22 *а*.



При переходе от абстрактного автомата к структурному, входные и выходные сигналы должны быть закодированы наборами сигналов структурного алфавита (входного или выходного соответственно). При двоичном структурном алфавите автомат Пz будет иметь два входных  и два выходных  канала.

Итак, сами компоненты *U*z и *Q*z при *Z* = 1,...,*R* векторов сигналов возбуждения памяти *U* и сигналов обратной связи от памяти *Q* также могут быть представлены в виде векторов:

*U*z = (*U*Z1,*U*Z2,...,*U*ZK) и *Q*Z = (*Q*Z1,*Q*Z2,...,*Q*ZR).

Если не оговорено особо, то используется двоичный структурный алфавит как для входных и выходных каналов синтезируемого автомата, так и для входных и выходных каналов автоматов памяти. Алфавит состояний автоматов памяти также обычно двоичный.

При построении функций возбуждения памяти автомата используют ***функцию входов элемента памяти*** μ(*b*i,*b*j), ставящую в соответствие каждой паре состояний (*b*i,*b*j) сигнал, который должен быть подан на вход этого автомата для перевода его из состояния *b*i в состояние *b*j. Функцию входов удобно задавать в виде таблицы. Для элемента памяти (функция переходов которого приведена ранее) *функция входов* имеет вид:



Если входные сигналы элемента памяти *q*1,...,*q*p закодированы наборами (*U*Z1,...,*U*ZK) сигналов на его входных каналах, то элементами таблицы, задающей функцию входов вместо *q*i будут соответствующие наборы. Так, если *q*1 = 00, *q*2 = 01, *q*3 = 10, то соответствующая *f* входов будет иметь вид рис.23a.

**Элементарные цифровые автоматы – элементы памяти.**

В качестве элементов памяти структурного автомата обычно используются триггеры.

**Триггер** – это устройство, имеющее два устойчивых состояния, в которые он переходит под действием определённых входных сигналов.

Обычно в триггерах выделяют два вида входных сигналов (и соответственно входов): информационные и синхросигналы.

Информационные сигналы определяют новое состояние триггера и присутствуют в любых триггерах. По типу информационных сигналов осуществляется классификация триггеров: *D, T, RS, JK, RST, DV* и т.д.

Синхросигнал не является обязательным и вводится в триггерах с целью фиксации момента перехода триггера в новое состояние, задаваемое информационными входами. Обычно, при синтезе ЦА используются триггеры с синхровходом, поэтому в дальнейшем будем рассматривать только такие триггеры.

На синхровход триггера поступают тактирующие импульсы задающего генератора, синхронизирующего работу ЦА. Период следования импульсов соответствует одному такту автоматного времени ЦА.

Рассмотрим основные типы триггеров, используемые для синтеза ЦА: *D, T, RS, JK*.

 ***D*-триггер** – элемент задержки – имеет один информационный вход *D* и один выход *Q* и осуществляет задержку поступившего на его вход сигнала на один такт.

Условное обозначение и таблица переходов *D*-триггера представлена на рис. .



|  |  |  |
| --- | --- | --- |
| *D* | *Q t* | *Q t+1* |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Таблица переходов *D-*триггера.

Рис. 24.

Из приведенной таблицы переходов для данного триггера *Q*t+1 = *f*(*Q*t,*D*t) можно получить таблицу функций его входов *D*t = ϕ(*Q*t, *Q*t+1).

|  |  |  |
| --- | --- | --- |
| *Q t* | *Q t+1* | *D t* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1Таблица функции входов *D-*триггера. | 0 | 0 |
| 1 | 1 | 1 |

Как видно из таблицы, состояние, в которое переходит триггер (средний столбец), совпадает с поступившим на его вход сигналом *D*(*t*) (правый столбец). В связи с этим таблица функций возбуждения памяти синтезируемого автомата с использованием *D*-триггеров будет полностью совпадать с кодированной таблицей переходов этого автомата. Промышленность выпускает *D*-триггера в интегральном исполнении. Например,



K155TM2 (рис. 25).Таких триггеров два в одном корпусе. Вход С –вход синхронизации, *Q*,*‾Q* – выходы, *Q –* прямой,  ** – инверсный. *R, S –* входы установки в 0 и 1 соответственно. При подаче на вход *R* и *S* логического нуля триггер устанавливается в соответствующие состояния независимо от сигнала на входах *D* и *C.*

***T*-триггер** – триггер со счетным входом – имеет один информационный вход *Т* и один выход *Q* и осуществляет суммирование по модулю два значений сигнала *T* и состояния *Q* в заданный момент времени.

 Условное обозначение и таблица переходов *T-*триггера представлена на рис 26.



|  |  |  |
| --- | --- | --- |
| *T* | *Q t* | *Q t+1* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Таблица переходов *T-*триггера.

Таблица функций входов триггера *T*t = *f*(*Q*t, *Q*t+1) представлена в таблице.

|  |  |  |
| --- | --- | --- |
| *Q t* | *Q t+1* | *T t* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1Таблица функции входов *T-*триггера. | 0 | 1 |
| 1 | 1 | 0 |

На основании этой таблицы можно получать функцию возбуждения элементов памяти при синтезе автомата на базе *T*-триггера. Например, если автомат перешел из состояния *a*i = 010 в состояние *a*j = 110, то для обеспечения этого перехода функции возбуждения должны быть:

для первого триггера при переходе из 0 в 1 *T*1 = 1,

для второго триггера при переходе из 1 в 1 *T*2 = 0,

для третьего триггера при переходе из 0 в 0 *T*3 =0 и т.д.

В чистом виде промышленность не выпускает *T*-триггера.

***RS*-триггер** – триггер с раздельными входами.

Данный триггер имеет два входных канала *R* и *S* и один выходной *Q*. Вход *S* (set) называется входом установки в единицу, вход *R* (reset) – входом установки в нуль. Условное обозначение и таблица переходов *RS*-триггера представлена на рис. 27.

В таблице переходов при подаче комбинации *S* = *R* = 1 состояние перехода *Q*t+1 не определено и эта комбинация сигналов является запрещенной для *RS*-триггера.

Таблицу переходов можно более компактно изобразить в виде (см. табл. 21б) Анализируя табл.21 б,в отмечаем что, например, переход триггера из 0 в 0требует подачи комбинации *R*=0, *S*=0 или *R*=1,*S*=0, т.е. можно сказать что этот переход будет при *R*=X (безразличное состояние) , *S*=0.

Аналогично рассуждая по отношению к другим переходам получим следующую таблицу функций входов.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *R* | *S* | *Q t* | *Q t+1* |  |  | *R* | *S* | *Q t+1* |
| 0 | 0 | 0 | 0 |  |  | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |  |  | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |  |  | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |  |  | 1 | 1 | – |
| 1 | 0 | 0 | 0 |  |  |  |  | б) |
| 1 | 0 | 1 | 0 |  |  |  |  |  |
| 1 | 1 | 0 | – |  |  |  |  |  |
| 1 | 1 | 1 | – | а) |  |  |  |  |

Таблица переходов *RS-*триггера.

*RS-*триггер

Т

T

C

Q

Рис. 27.

Таблица функций входов *RS-*триггера.

в)

Табл 21. а), б), в).

|  |  |  |  |
| --- | --- | --- | --- |
| *Q t* | *Q t+1* | *Rt* | *S* |
| 0 | 0 | X | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | X |

На основании таблицы можно получить функцию возбуждения памяти автомата при синтезе на базе *RS*-триггеров. Например, если автомат переходит из состояния *ai*= 010 в состояние *aj*=110, то для обеспечения такого перехода функции возбуждения должны быть:

для первого триггера при переходе из 0 в 1 *R*1 =0, *S*1 = 1;

для второго триггера при переходе из 1 в 1 *R*2 =0, *S*2 = X;

для третьего триггера при переходе из 0 в 0 *R*3 =X, *S*3= 0.

Аналогично для любого другого перехода автомата.

В чистом виде синхронный RS - триггер, используемый для синтеза ЦА, промышленностью не выпускается.

***JK- триггер –*** имеет два информационных входа *J* и *K* и один выход *Q*. Вход *J –* вход установки в 1, вход *K* – вход установки в 0, т.е. эти входы аналогичны соответствующим входам *RS*-триггера: *J* – соответствует *S*,  *K* – соответствует *R*. Однако, в отличие от *RS*-триггера, входная комбинация *J* = 1, *K*= 1 не является запрещённой. Условное обозначение и таблица переходов *JK*-триггера представлены на рис.28. и в табл. 22.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *J* | *K* | *Q t* | *Q t+1* |  |  | *J* | *K* | *Q t+1* |
| 0 | 0 | 0 | 0 |  |  | 0 | 0 | *Q t* |
| 0 | 0 | 1 | 1 |  |  | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |  |  | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |  |  | 1 | 1 | *Q t* |
| 1 | 0 | 0 | 1 |  |  |  |  | б) |
| 1 | 0 | 1 | 1 |  |  |  |  |  |
| 1 | 1 | 0 | 1 |  |  |  |  |  |
| 1 | 1 | 1 | 0 | а) |  |  |  |  |

Т

J

K

Q

*JK-*триггер

Таблица переходов *JK-*триггера.

Рис. 28.

Табл. 22 а), б).

Как следует из таблиц переходов, для комбинаций входных сигналов *JK* = 00÷10 триггер ведет себя как *RS*-триггер, а при комбинации *JK* = 11 – как *T*-триггер.

Анализируя таблицу переходов ( табл. 22 а), отмечаем, что переход триггера, например, из 0 в 1 требует подачи входных сигналов *J*=1, *K*=0 или *J*=1, *K*=1, т.е. *J*=1, *K*=Х (безразличное значение). Аналогично рассуждая по отношению к другим переходам, получим следующую таблицу функций входов *JK*-триггера.

|  |  |  |  |
| --- | --- | --- | --- |
| *Q t* | *Q t+1* | *J* | *K* |
| 0 | 0 | X | 0 |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | 0 | X |

Таблица функций выходов *JK-*триггера.

На основании последней таблицы можно получить функцию возбуждения элементов памяти при синтезе автомата на *JK*-триггерах. Например, при переходе автомата из состояния *ai*=010 в состояние *aj*=110, функции возбуждения должны быть:

для первого триггера при переходе из 0 в 1 *J*1 = 1, *K*1 = X;

для второго триггера при переходе из 1 в 1 *J*2 = X, *K*2 = 0;

для третьего триггера при переходе из 0 в 0 *J*3 = 0, *K*3 = X.

**Пример канонического метода структурного синтеза автомата.**

Выполним структурный синтез *частичного* автомата *А*, заданного своими таблицами переходов и выходов (табл. 23 и 24.).



*Синтез будем выполнять в следующем порядке:*

**1.** Выберем в качестве элементов памяти *D*-триггер, функция входов которого представлена в таблице стр. 33.

**2.** Закодируем входные, выходные сигналы и внутренние состояния автомата. Количество входных абстрактных сигналов *F* = 3, следовательно количество входных структурных сигналов *L*= ]**log**2*F* [ = ]**log**23[ = 2, т.е. *х*1, *х*2.

Количество выходных абстрактных сигналов *G* = 4, следовательно количество выходных структурных сигналов *N* =]log2*G*[ = ]log24[ = 2, т.е. *у*1, *у*2. Количество внутренних состояний абстрактного автомата *M* = 4, следовательно количество двоичных элементов памяти (триггеров) *R* = ] **log**2*M* [ = ]**log**24[ = 2.



Следовательно, структура ЦА с учетом того, что исходный автомат является автоматом Мили, в качестве элементов памяти используется *D*-триггер, может быть представлена в виде(рис. 29):

Кодирование входных, выходных сигналов и внутренних состояний представлена в таблицах:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *x*1 | *x*2 |  |  | *y*1 | *y*2 |  |  | *Q*1 | *Q*2 |  |
|  | *z*1 | 0 | 0 |  | *w*1 | 0 | 0 |  | *a*1 | 0 | 0 |  |
|  | *z*2 | 0 | 1 |  | *w*2 | 0 | 1 |  | *a*2 | 0 | 1 |  |
|  | z3 | 1 | 1 |  | *w*3 | 1 | 1 |  | *a*3 | 1 | 1 |  |
|  |  |  | *w*4 | 1 | 0 |  | *a*4 | 1 | 0 |  |

Кодирование, в общем случае, осуществляется произвольно. Поэтому, например, каждому из сигналов *Zi* можно поставить в соответствие любую двухразрядную комбинацию *х*1, *х*2. Необходимо только, чтобы разные выходные сигналы *Zi* кодировались разными комбинациями *х*1, *х*2. Аналогично для *Wi* и *ai*.

1. Получим кодированные таблицы переходов и выходов структурного автомата. Для этого в таблицах переходов и выходов исходного абстрактного автомата вместо *Zi*, *Wi*, *ai* cтавим соответствующие коды. Получим таблицы:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2Рис. 30 .Кодированная таблица переходов.Рис. 31 .Кодированная таблица выходов |  |  | *a*1 | *a*2 | *a*3 | *a*4 |  |  |  | *a*1 | *a*2 | *a*3 | *a*4 |
|  |  |  | 00 | 01 | 11 | 10 |  |  |  | 00 | 01 | 11 | 10 |
|  | *Z*1 | 00 | 00 | 10 | 10 | – |  | *Z*1 | 00 | 01 | 00 | 11 | – |
|  | *Z*2 | 01 | – | 11 | 00 | – |  | *Z*2 | 01 | – | 11 | 00 | – |
|  | *Z*3 | 11 | 01 | – | 01 | Q1Q2 |  | *Z*3 | 11 | 00 | – | 10 | y1y2 |

В кодированной таблице переходов заданы функции 



В кодированной таблице выходов заданны функции:

 

**4.** При каноническом методе синтез сводится к получению функций:

 

и последующем построении комбинационных схем, реализующих данную систему булевых функций.

Функции *у*1 и *у*2 могут быть непосредственно получены из таблицы выходов, например, в виде :

 

Однако выражения для *у*1 и *у*2 можно существенно упростить в результате минимизации, например, с помощью карт Карно:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | 0 | 0 | 1 | – |  | 00 | 1 | 0 | 1 | – |
|  | 01 | – | 1 | 0 | – |  | 01 | – | 1 | 0 | – |
|  | 11 | 0 | – | 1 | 0 |  | 11 | 0 | – | 0 | 1 |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |

*x*1*x*2

*Q*1*Q*2

*x*1*x*2

*Q*1*Q*2

Рис. 32 .

Карта Карно для *у*1.

Рис.33 .

Карта Карно для *у*2.

В результате минимизации имеем:

 

Для получения выражений для *D*1 и *D*2 необходимо получить таблицы функций возбуждения. Для чего в общем случае необходимо воспользоваться таблицей переходов и функциями входов элементов памяти. Зная код исходного состояния автомата и код

состояния перехода на основании таблицы входов триггера получаем требуемое значение функции возбуждения, обеспечивающее заданный переход. Однако для *D*-триггеров, как отмечалось ранее, таблица переходов совпадает с таблицей функции возбуждения. Тогда либо непосредственно из этой таблицы, либо в результате минимизации получаем требуемые значения *Di*. Обычно используется минимизация с помощью карт Карно:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2Рис. 34 .Карта Карно для *D*1.Рис. 35.Карта Карно для *D*2. |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | 0 | 1 | 1 | – |  | 00 | 0 | 0 | 0 | – |
|  | 01 | – | 1 | 0 | – |  | 01 | – | 1 | 0 | – |
|  | 11 | 0 | – | 0 | 1 |  | 11 | 1 | – | 1 | 1 |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |

В результате минимизации получаем:



**5.** На основании полученных в результате синтеза булевых выражений ((\*), (\*\*)) ,строим функциональную схему автомата. Для этого уравнения ((\*), (\*\*)) представим в виде:



*Функциональная схема автомата представлена на странице 41:*

Дополнительно на функциональной схеме показан сигнал , устанавливающий автомат в начальное состояние (в данном случае 00).

****

**Особенности синтеза автоматов на базе *T*, *RS, JK* триггеров*.***

Необходимо отметить, что синтез на базе указанных типов триггеров осуществляется аналогично выполненному синтезу на базе *D*-триггеров. В частности, п. 1÷3 (см. предыдущий параграф) абсолютно аналогичны. Кроме того, как следует из п.4 (см. предыдущий параграф) выходные сигналы не зависят от типа триггеров, поэтому выражение для *yi* будут одинаковыми для любого типа триггеров. Однако функции возбуждения будут различны для разных типов триггеров и получаются на основании таблицы переходов исходного автомата и функции входов выбранного триггера. Без особых пояснений ниже приведены таблицы функций входов, функций возбуждений и карты Карно для минимизации функций возбуждения при использовании для синтеза автомата предыдущего параграфа *T*-, *RS*-, *JK*-триггеров.

***T*-триггер.**

|  |  |  |
| --- | --- | --- |
| *Q t* | *Q t+1* | *T t* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Таблица функции входов *T-*триггера.

Таблица функций возбуждения.

*x*1*x*2

*Q*1*Q*2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 00 | 11 | 01 | – |
| 01 | – | 10 | 11 | – |
| 11 | 01 | – | 10 | 01 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | 0 | 1 | 0 | – |  | 00 | 0 | 1 | 1 | – |
|  | 01 | – | 1 | 1 | – |  | 01 | – | 0 | 1 | – |
|  | 11 | 0 | – | 1 | 0 |  | 11 | 1 | – | 0 | 1 |
|  | 10 | – | – | – | – |  | 10 | – | – | – | 0 |

Карта Карно для *Т*1.



Карта Карно для *Т*2.



*x*1*x*2

*Q*1*Q*2

*x*1*x*2

*Q*1*Q*2

***RS*-триггер.**

|  |  |  |  |
| --- | --- | --- | --- |
| *Q t* | *Q t+1* | *R* | *S* |
| 0 | 0 | X | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | X |

Таблица функции входов *RS-*триггера.

Таблица функций возбуждения.

*x*1*x*2

*Q*1*Q*2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
|  | *R*1 | *S*1 | *R*2 | *S*2 | *R*1 | *S*1 | *R*2 | *S*2 | *R*1 | *S*1 | *R*2 | *S*2 | *R*1 | *S*1 | *R*2 | *S*2 |
| 00 | X | 0 | X | 0 | 0 | 1 | 1 | 0 | 0 | X | 1 | 0 | – | – |
| 01 | – | – | 0 | 1 | 0 | X | 1 | 0 | 1 | 0 | – | – |
| 11 | X | 0 | 0 | 1 | – | – | 1 | 0 | 0 | X | 0 | X | 0 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2 |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | X | 0 | 0 | – |  | 00 | 0 | 1 | X | – |
|  | 01 | – | 0 | 1 | – |  | 01 | – | 1 | 0 | – |
|  | 11 | X | – | 1 | 0 |  | 11 | 0 | – | 0 | X |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | X | 1 | 1 | – |  | 00 | 0 | 0 | 0 | – |
|  | 01 | – | 0 | 1 | – |  | 01 | – | X | 0 | – |
|  | 11 | 0 | – | 0 | 0 |  | 11 | 1 | – | X | 1 |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |

 

***JK*-триггер.**

|  |  |  |  |
| --- | --- | --- | --- |
| *Q t* | *Q t+1* | *J* | *K* |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
|  | *J*1 | *K*1 | *J*2 | *K*2 | *J*1 | *K*1 | *J*2 | *K*2 | *J*1 | *K*1 | *J*2 | *K*2 | *J*1 | *K*1 | *J*2 | *K*2 |
| 00 | 0 | X | 0 | X | 1 | X | X | 1 | X | 0 | X | 1 | – | – |
| 01 | – | – | 1 | X | X | 0 | X | 1 | X | 1 | – | – |
| 11 | 0 | X | 1 | X | – | – | X | 1 | X | 0 | X | 0 | 1 | X |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | 0 | 1 | X | – |  | 00 | X | X | 0 | – |
| Таблица функции входов *JK-*триггера.Таблица функций возбуждения.*x*1*x*2*Q*1*Q*2**Карта Карно для *J*1.****Карта Карно для *K*1.***x*1*x*2*Q*1*Q*2*x*1*x*2*Q*1*Q*2 | 01 | – | 1 | X | – |  | 01 | – | X | 1 | – |
|  | 11 | 0 | – | X | X |  | 11 | X | – | 1 | 0 |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 00 | 01 | 11 | 10 |  |  | 00 | 01 | 11 | 10 |
|  | 00 | 0 | X | X | – |  | 00 | X | 1 | 1 | – |
|  | 01 | – | X | X | – |  | 01 | – | 0 | 1 | – |
|  | 11 | X | – | 1 | 0 |  | 11 | 0 | – | 0 | X |
|  | 10 | – | – | – | – |  | 10 | – | – | – | – |

**Карта Карно для *J*2.**

**Карта Карно для *K*2.**

*x*1*x*2

*Q*1*Q*2

*x*1*x*2

*Q*1*Q*2



Функциональные схемы автоматов с различными типами триггеров предлагается построить самостоятельно.

**Кодирование внутренних состояний ЦА.**

**Гонки в автомате.**

***Кодирование заключается в сопоставлении каждому состоянию автомата набора (кода) состояний элементов памяти.*** При этом наборы для всех состояний должны иметь одинаковую длину, а разным состояниям автомата должны соответствовать разные наборы. ***Если элементы памяти двоичные, то их число .***

Переход автомата из одного состояния в другое осуществляется за счет изменения состояний элементов памяти. Если автомат переходит из состояния с кодом 010 в состояние с кодом 100, то это означает, что триггер *V*1 переходит из состояния 0 в состояние 1, *V*2 – из 1 в 0, *V*3 – сохраняет свое состояние.

При функционировании автомата могут появиться так называемые состязания. Это явление возникает вследствие того, что элементы памяти имеют различные, хотя и достаточно близкие, времена срабатывания. Различны также задержки сигналов возбуждения, поступающих на входные каналы элементарных автоматов по логическим цепям неодинаковой длины.



Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько запоминающих элементов, то между ними начинаются состязания. Тот элемент, который выиграет эти состязания, т.е. изменит свое состояние ранее, чем другие элементы, может через цепь обратной связи изменить сигналы на входах некоторых запоминающих элементов до того, как другие, участвующие в состязаниях элементы, изменят свои состояния. Это может привести к переходу автомата в состояние, не предусмотренное его графом. Поэтому в процессе перехода из состояния *a*m в состояние *a*s под действием входного сигнала *Z*f автомат может оказаться в состоянии *a*k или *al*: (рис.36.).

Если затем при том же входном сигнале *Z*f автомат из *а*k и *аl* перейдет в *а*s, то такие состязания являются допустимыми или некритическими.

Если же в этом автомате есть переход, например, из *а*k в *аj* *а*s под действием того же сигнала *Z*f, то автомат может перейти в *аj*, а не в *а*s и правильность его работы будет нарушена (рис.37.).



Такие состязания называются критическими состязаниями или гонками и необходимо принимать меры для их устранения.

Устранить гонки можно аппаратными средствами либо используя специальные методы кодирования. Один из способов ликвидации гонок состоит ***в тактировании входных сигналов автомата импульсами определенной длительности.*** Предполагается, что кроме входных каналов *х*1, ..., *хl* имеется еще канал *С* от генератора синхроимпульсов, по которому поступает сигнал *С* = 1 в момент прихода импульса и *С* = 0 при его отсутствии. В связи с этим входным сигналом на переходе (*a*m, *a*s) будет не *Zf*, а *CZf*. Тогда, если длительность импульса *t*c меньше самого короткого пути прохождения тактированного сигнала обратной связи по комбинационной схеме, то к моменту перехода в промежуточное состояние *a*k сигнал *C* = 0, *CZf*=0, что исключает гонки. Канал *С* – это фактически синхровход триггера. Недостаток метода – в трудности подбора требуемой длительности импульса, т.к. она зависит от многих факторов, не поддающихся строгому учету.

Другой способ ликвидации гонок заключается во введении двойной памяти. В этом случае каждый элемент памяти дублируется, причем перепись из первого элемента памяти во второй происходит в момент *С* = 0(рис.38.).



Сигналы обратной связи для получения функций возбуждения и функций выходов автомата снимаются с выхода второго триггера. Таким образом состязания могут возникнуть только между первыми триггерами, сигналы ОС (выходы вторых триггеров) не могут измениться до тех пор, пока *С* не станет равным 0. Но тогда *CZf* = 0, первый триггер перестанет воспринимать информацию, и гонок не будет.

Для устранения гонок используются специальные методы противогоночного кодирования, среди которых чаще всего применяется так называемое соседнее кодирование состояний автомата, при котором условие отсутствия гонок всегда выполнено. При соседнем кодировании любые два, состояния связанные дугой на графе автомата кодируются наборами, отличающимися состояниями лишь одного элемента памяти.

Соседнее кодирование не всегда возможно. Граф автомата, допускающее соседнее кодирование, должен удовлетворять ряду требований, а именно:

1. в графе автомата не должно быть циклов с нечетным числом вершин;
2. два соседних состояния второго порядка не должны иметь более двух состояний, лежащих между ними.



Под состояниями второго порядка понимаются такие два состояния, путь между которыми по графу автомата состоит из двух ребер (независимо от ориентации). Примеры графов автоматов допускающих и не допускающих соседнее кодирование представлены на рис.39а. и 39б. соответственно.



При соседнем кодировании обычно пользуются картой Карно. В этом случае состояния, связанные дугой, располагают на соседних клетках карты (рис.40.).

Легко видеть, что при соседнем кодировании на каждом переходе переключается только один триггер, что принципиально устраняет гонки.

 **Кодирование состояний и сложность комбинационной схемы автомата.**

Анализ канонического метода структурного синтеза автомата показывает, что различные варианты кодирования состояний автомата приводят к различным выражениям функций возбуждения памяти и функций выходов, в результате чего сложность комбинационной схемы существенно зависит от выбранного кодирования. Среди множества существующих алгоритмов кодирования рассмотрим лишь два наиболее часто встречаемых:

**1)** алгоритм кодирования для *D*-триггеров;

**2)** эвристический алгоритм кодирования.

 **Алгоритм кодирования для *D*-триггеров.**

Согласно рассматриваемому алгоритму при кодировании необходимо выполнить следующее:

1. Каждому состоянию автомата *а*m (m = 1,2,...,*M*) ставится в соответствие целое число *N*m, равное числу переходов в состояние *а*m (*N*m равно числу появлений *а*m в поле таблицы переходов или числу дуг, входящих в *а*m при графическом способе задания автомата).
2. Числа *N*1, *N*2, ..., *N*m упорядочиваются по убыванию.
3. Состояние *а*s с наибольшим *N*s кодируется кодом:, где *R*-количество элементов памяти.
4. Следующие *R* состояний согласно списка пункта 2 кодируются кодами, содержащими только одну 1: 00 ... 01, 00 ... 10, ... , 01 ... 00, 10 ... 00.
5. Для оставшихся состояний опять в порядке списка п.2. используют коды с двумя единицами, затем с тремя и так далее пока не будут закодированы все состояния.

В результате получается такое кодирование, при котором чем больше имеется переходов в некоторое состояние, тем меньше единиц в его коде. Т.к. для *D*-триггеров функции возбуждения однозначно определяются кодом состояния перехода, то очевидно, что выражения для функций возбуждения будут проще. Этот метод особенно эффективен при отсутствии минимизации функций возбуждения, что имеет место в реальных автоматах с большим количеством внутренних состояний и входных переменных.

В частности, для автомата, заданного своими таблицами переходов и выходов (см. ниже) при кодировании на базе *D*-триггеров.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *a*1 | *a*2 | *a*3 | *a*4 | *a*5 |  |  | *a*1 | *a*2 | *a*3 | *a*4 | *a*5 |
| *Z*1 | *a*1 | *a*1 | *a*5 | *a*3 | *a*1 |  | *Z*1 | *w*1 | *w*2 | *w*1 | *w*1 | *w*1 |
| *Z*2 | *a*2 | *a*3 | *a*2 | *a*3 | *a*3 |  | *Z*2 | *w*1 | *w*3 | *w*4 | *w*2 | *w*2 |
| *Z*3 | *a*3 | *a*4 | *a*2 | *a*4 | *a*2 |  | *Z*3 | *w*2 | *w*2 | *w*2 | *w*1 | *w*3 |



 *a*1 ~ *N*1 = 3 *N*3 *a*3 = 000

 *a*2 ~ *N*2 = 4 *N*2 *a*2 = 001

 *a*3 ~ *N*3 = 5 *N*1 *a*1 = 010

 *a*4 ~ *N*4 = 5 *N*4 *a*4 = 100

 *a*5 ~ *N*5 = 1 *N*5 *a*5 = 011

Аналогично кодированию внутренних состояний для *D*-триггеров можно кодировать выходные сигналы для любого типа триггеров, т.е. чем чаще вырабатывается данный выходной сигнал *w*i, тем меньше единиц в его коде. Так для автомата (рис.41.) имеем:

 *w*1 ~ *N*1 = 6 *N*1 *w*1 = 00

 *w*2 ~ *N*2 = 5 *N*2 *w*2 = 01

 *w*3 ~ *N*3 = 2 *N*3 *w*3 = 10

 *w*4 ~ *N*4 = 2 *N*4 *w*4 = 11

Предполагается самостоятельно окончить синтез автомата при данном кодировании и при любом другом. Результаты сравнить.

**Эвристический алгоритм кодирования.**

Данный алгоритм минимизирует суммарное число переключений элементов памяти на всех переходах автомата и используется для кодирования состояний автомата при синтезе на базе *T*, *RS*, *JK*-триггеров. Для данных типов триггеров (в отличие от *D*-триггеров!) на каждом переходе, где триггер меняет свое значение на противоположное, одна из функций возбуждения обязательно равна 1. Уменьшение числа переключений триггеров приводит к уменьшению количества единиц соответствующих функций возбуждения, что при отсутствии минимизации однозначно приводит к упрощению комбинационной схемы автомата.

Введем некоторые определения.

Пусть *Г*(*S*) – неориентированный граф переходов автомата *S*. Вершины графа отождествляются с состояниями автомата. Вершины *i* и *j* соединены ребром, если есть переход из *аi* и *аj* или наоборот.

Обозначим *q*(*i, j*) число всевозможных переходов автомата из *аi* в *аj*. Каждому ребру (*i, j*) графа *Г(S)* поставим в соответствие вес ребра *р*(*i, j*) = *q*(*i, j*) *+ q*(*j, i*).

Введем функцию *w*(*i, j*) = *р*(*i, j*)⋅ *d*(*i, j*), где *d*(*i, j*) – число компонентов, которыми коды состояний *аi* в *аj* отличаются друг от друга (т.е. кодовое расстояние между кодами *аi* в *аj*).

 Функция *w*(*i ,j*) имеет простой физический смысл. Перход автомата из *аi* в *аj* (или наоборот) сопровождается переключением стольких триггеров, сколькими компонентами отличаются коды этих состояний, т.е. их число равно *w*(*i ,j*). Следовательно, при переходе автомата по всем ребрам, соединяющим состояниям *аi* и *аj* (их число *p*(*i, j*)!) всего переключится количество триггеров, равное *p*(*i, j*)⋅*d*(*i ,j*) =*w*(*i ,j*).

Но тогда функция показывает, сколько всего переключается триггеров при прохождении автомата по всем возможным переходам. Функция *w* показывает, сколько всего единиц в функции возбуждения, т.е. позволяет оценивать сложность комбинационной схемы автомата. *W* можно рассматривать как некую целевую функцию, минимум которой определит такое кодирование, при котором комбинационная схема наиболее простая. Кстати, миинмальное кодовое расстояние между различными состояниями равно 1 и если удается закодировать все состояния соседним кодированием, то очевидно, что *w* будет минимально возможным и равным , т.е. суммарному числу переходов для автомата.

Из выражения для *w* следует, что переход из *аi* в *аi*, для которого *d*(*i,i*)=0, не влияет на *w* (что вполне очевидно, если учесть, что на этом переходе ни один триггер не переключается).

Рассмотрим применение эвристического алгоритма на конкретном примере автомата, заданного таблицами переходов и выходов (рис.41. ). Для данного автомата можно построить ориентированный граф (без учета петель), представленный на рис.42. На каждом ребре указан его вес.



**Эвристический алгоритм состоит из следующих шагов.**

**1.** Строим матрицу , состоящую из всех пар номеров (*i, j*), для которых *р*(*i, j*) ≠ 0 (т.е. в автомате есть переход из *а*i в *а*j или наоборот) и *i*<*j*. Для каждой пары в матрице  указываем ее вес *р*(*i, j*), совпадающий с весом ребра соединяющего *а*i и *а*j.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *i* | *j* | *p(i,j)* |
|  |  | 1 | 2 | 2 |
|  |  | 1 | 3 | 1 |
| T  | = | 1 | 5 | 1 |
|  |  | 2 | 3 | 3 |
|  |  | 2 | 4 | 1 |
|  |  | 2 | 5 | 1 |
|  |  | 3 | 4 | 2 |
|  |  | 3 | 5 | 2 |

**2.** Упорядочим строки матрицы , для чего построим матрицу  следующим образом. В первую строку матрицы  поместим пару (*α,β*) с наибольшим весом *р*(*α,β*). В нашем случае (*α,β*) = (2,3), *р*(2,3) = 3. Из всех пар, имеющих общий компонент с парой (*α,β*) выбирается пара (*γ,δ*) с наибольшим весом и заносится во вторую строку матрицы . Ясно, что {*α,β*}⋅{*γ,δ*}≠0. Затем из всех пар, имеющих общий компонент хотя бы с одной из внесенных уже в матрицу  пар выбирается пара с наибольшим весом и заносится в матрицу  и т.д.. В случае равенства весов пар вычисляются суммы весов компонентов пар (весом *р*(*α*) компонента *α* называется число появлений α в матрице ) и в матрицу  заносится пара с наибольшей суммой весов. В рассматриваемом автомате на второе место вслед за парой (2,3) претендуют пары: (1,2) с *р*(1,2) = 2; (3,4) с *р*(3,4) = 2, (3,5) с *р*(3,5) = 2.

Для определения того, какая пара займет второе место в матрице  находим веса компонентов пар:

 *р*(1) = 3 *р*(2) = 3 *р*(1) + *р*(2) = 6

 *р*(3) = 4 *р*(4) = 2 *р*(3) + *р*(4) = 6

 *р*(3) = 4 *р*(5) = 2 *р*(3) + *р*(5) = 6

В данном случае для всех пар совпадают и их веса и веса их компонентов. Поэтому на второе место матрицы  может быть поставлена любая из пар (1,2), (3,4), (3,5). Но тогда на 3-м и 4-м будут остальные две. Выполнив упорядочивание всех пар, получим матрицу  в виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *i* | *j* | *p(i,j)* |
|  |  | 2 | 3 | 3 |
|  |  | 1 | 2 | 2 |
| M  | = | 3 | 4 | 2 |
|  |  | 3 | 5 | 2 |
|  |  | 1 | 3 | 1 |
|  |  | 1 | 5 | 1 |
|  |  | 2 | 4 | 1 |
|  |  | 2 | 5 | 1 |

**3.** Определяем разрядность кода для кодирования состояний автомата (количество элементов памяти – триггеров). Всего состояний M=5. Тогда

 R = ]log2M[ = ]log25[ =3.

Закодируем состояния из первой строки матрицы следующим образом: *K*2 = *K*(*а*2) = 000; *K*3 = *K*(*а*3) = 001.

 Для удобства кодирования будем иллюстрировать этот процесс картой Карно:



1. Вычеркнем из матрицы  первую строку, соответствующую закодированным состояниям *а*2 и *а*3. Получим матрицу .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *i* | *j* | *p(i,j)* |
|  |  | 1 | 2 | 2 |
|  |  | 3 | 4 | 2 |
| M’  | = | 3 | 5 | 2 |
|  |  | 1 | 3 | 1 |
|  |  | 1 | 5 | 1 |
|  |  | 2 | 4 | 1 |
|  |  | 2 | 5 | 1 |

**5.** В силу упорядочивания п.2 в первой строке закодирован ровно один элемент. Выберем из первой строки незакодированный элемент и обозначим его *γ*. (В нашем случае *γ* = 1).

**6.** Строим матрицу , выбрав из  строчки, содержащие *γ*.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | *i* | *j* | *p(i,j)* |
|  |  |  |  | 1 | 2 | 2 |
| M*γ*  | = | M’  | = | 1 | 3 | 1 |
|  |  |  |  | 1 | 5 | 1 |

Пусть *Bγ* = {*γ*1,...,*γF*} – множество элементов из матрицы , которые уже закодированы. Их коды *Кγ*1,..., *KγF* соответственно. В нашем случае:

 *Bγ* = *B*3 = {2,3} *K*2 = 000 *K*3 = 001.

**7.** Для каждого *Kγf* (*f*=1, ..., *F*) найдем – множество кодов, соседних с  и еще не занятых для кодирования состояний автомата. (Для соседних кодов кодовое расстояние d=1).

 *K*2 = 000  = {100, 010}

 *K*3 = 001  = {011, 101}.

Построим множество 



Если оказывается, что , то строим новое множество , где – множество кодов, у которых кодовое расстояние до кода равно 2 и т.д..

**8.** Для каждого кода из множества *Dγ* находим кодовое расстояние до кода .

 *K*2 = 000 *K*3 = 001

 *d*(100, 000) = 1 *d*(100, 001) = 2

 *d*(010, 000) = 1 *d*(010, 001) = 2

 *d*(011, 000) = 2 *d*(011, 001) = 1

 *d*(101, 000) = 2 *d*(100, 001) = 1

**9.** Находим значение функции *w* для каждого кода из множества *Dγ*.



**10.** Из множества *Dγ* выбираем код *Kγ*, у которого получилось минимальное значение *w* в п.9. Выбираем код для состояния *a*1 *К*1 =100.



**11.** Из матрицы  вычеркиваем строки, в которых оба элемента уже закодированы, в результате чего получим новую матрицу . Если в новой матрице  не осталось ни одной строки, то кодирование закончено. В противном случае возвращаемся к п.5. В нашем случае имеем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *i* | *j* | *p(i,j)* |
|  |  | 3 | 4 | 2 |
|  |  | 3 | 5 | 2 |
| M’  | = | 1 | 5 | 1 |
|  |  | 2 | 4 | 1 |
|  |  | 2 | 5 | 1 |

 

*К*2 = 000  = {010}

*K*3 = 001  = {011, 101}

 

 *K*2 = 000 *K*3 = 001

 *d*(010, 000) = 1 *d*(010, 001) = 2

 *d*(011, 000) = 2 *d*(011, 001) = 1

 *d*(101, 000) = 2 *d*(101, 001) = 1

 

Выбираем *К*4 = 101.





 *К*1 = 100  = {110}

 *K*2 = 000  = {010}

 *К*3 = 001  = {011}



 *К*1 = 100 *K*2 = 000 *K*3 = 001

 *d*(110, 100) = 1 *d*(110, 000) = 2 *d*(110, 001) = 3

 *d*(010, 100) = 2 *d*(010, 000) = 1 *d*(010, 001) = 2

 *d*(011, 100) = 3 *d*(011, 000) = 2 *d*(011, 001) = 1



Выбираем *К*5 = 011.



Т.к. все состояния автомата закодированы, то работа алгоритма заканчивается. Общее количество переключений триггеров:



Минимально возможное количество переключений (если бы состояния были закодированы соседним кодированием)

 

Коэффициент эффективности кодирования:

 

Рассмотренный алгоритм кодирования является машино-ориентированным, существуют программы, реализующие этот алгоритм.

Необходимо отметить в заключении, что использование алгоритма кодирования для *D-*триггеров или эвристического алгоритма для других типов триггеров обеспечивает наиболее простую с точки зрения реализации схему, но при этом возможны гонки. Для радикального устранения последних используют аппаратные методы – триггеры с двойной памятью: триггеры, управляемые фронтом и т.д..

**Управляющие и операторные автоматы.**

**Принцип микропрограммного управления.**

ЭВМ перерабатывает информацию, выполняя над ней какие-то операции. Для выполнения операций над информацией используются операционные устройства – процессоры, каналы ввода-вывода, устройства управления внешними устройствами и т.д. Функцией операционного устройства является выполнение заданного множества операций *F*={*f*1,...,*f*G} над входными словами *D*={*d*1,...,*d*H} c целью вычисления слов *R*={*r*1,...,*r*Q}, которые представляют результаты операций *R*=*f*g(*D*), где *g*=1,2,...,*G*.

Функциональная и структурная организация операционных устройств базируется на принципе микропрограммного управления, который состоит в следующем:

**1.** Любая операция *f*g(*g*=1,...,*G*), реализуемая устройством, рассматривается как сложное действие, которое разделяется на последовательность элементарных действий над словами информации. Эти элементарные действия называются ***микрооперациями***.

**2.** Для управления порядком следования микроопераций используются ***логические условия***, которые в зависимости от значений слов, преобразуемых микрооперациями, принимают значения "*ложь*" или "*истина*" (1 или 0).

**3.** Процесс выполнения операций в устройстве описывается в форме алгоритма, который представляется в терминах микроопераций и логических условий и называется ***микропрограммой***. Микропрограмма определяет порядок проверки значений логических условий и следования микроопераций, необходимый для получения требуемых результатов.

**4.** Микропрограмма используется как форма представления функции устройства, на основе которой определяется структура и порядок функционирования устройства во времени.

 Т.о. из принципа микропрограммного управления следует, что структура и порядок функционирования операционных устройств предопределяется алгоритмом выполнения операции *F*={*f*1,...,*f*G}.

К элементарным действиям над словами информации микрооперациям относятся: *передача информации из одного регистра в другой, взятие обратного кода, сдвиг и т.д.*

**Понятие операционного и управляющих автоматов.**

Как показал академик В.М. Глушков в любом устройстве обработки цифровой информации можно выделить два основных блока – операционный автомат (ОА) и управляющий автомат (УА).



Операционный автомат (ОА) служит для хранения слов информации, выполнения набора микроопераций и вычисления значений логических условий, т.е. операционный автомат является структурой, организованной для выполнения действий над информацией. Микрооперации, выполняемые ОА, задаются множеством управляющих сигналов  *Y*={*y*1,....,*y*M}, с каждым из которых отождествляется определенная микрооперация.

Значения логических условий, вычисляемые в операционном автомате, отображаются множеством осведомительных сигналов *X*={*x*1,...,*x*L}, каждый из которых отождествляется с определенным логическим условием.

Управляющий автомат (УА) генерирует последовательность управляющих сигналов, предписанную микропрограммой и соответствующую значениям логическим условий. Иначе говоря, управляющий автомат задает порядок выполнения действий в ОА, вытекающий из алгоритма выполнения операций. Наименование операции, которую необходимо выполнить в устройстве, определяется кодом *g* операции, поступающим в УА извне. По отношению к УА сигналы *g*1,...,*g*h, посредством которых кодируется наименование операции и осведомительные сигналы *x*1,...,*x*L, формируемые в операционном автомате, играют одинаковую роль: они влияют на порядок выработки управляющих сигналов Y. Поэтому сигналы *g*1,...,*gh* и *x*1,...,*x*L относятся к одному классу – к классу осведомительных сигналов, поступающих на вход УА.

Т.о. любое операционное устройство – процессор, канал ввода-вывода и т.д. – является композицией операционного и управляющего автоматов. Операционный автомат, реализуя действия над словами информации, является исполнительной частью устройства, работой которого управляет управляющий автомат, генерирующий необходимые последовательности управляющих сигналов.

Операционный и управляющий автоматы могут быть определены своими функциями – перечнем выполняемых ими действий.

 ***Функция ОА определяется следующей совокупностью сведений:***

**1)** множеством входных слов *D*={*d*1,...,*d*H}, вводимых в автомат в качестве операндов;

**2)** множеством выходных слов *R*={*r*1,...,*r*Q}, представляющих результаты операций;

**3)** множеством внутренних слов *S*={*s*1,...,*s*N}, используемых для представления информации в процессе выполнения операций. Можно считать, что входные и выходные слова совпадают с определенными внутренними *D*⊆S, R⊆S.

**4)** множеством микроопераций *Y*={*y*m}, реализующих преобразование *S*=*ϕ*m(*s*) над словами информации, где *ϕ*m – вычисляемая функция;

**5)** множеством логических условий *X*={*xi*}, где *xi*=*ψi*(*si*) и *ψi* – булева функция;

T.o. функция ОА задана, если заданы (определены) множества *D*, *R*, *S*, *Y*, *X*. Время не является аргументом функции ОА. Функция устанавливает список действий-микроопераций и логических условий, которые может выполнять автомат, но никак не определяет порядок следования этих действий во времени. Т.е. функция ОА характеризует средства, которые могут быть использованы для вычислений, но не сам вычислительный процесс.

Порядок выполнения действий во времени определяется в форме функций управляющего автомата.

Функция управляющего автомата – это операторная схема алгоритма ( микропрограммы), функциональными операторами которой являются символы *у*1,...,*у*m, отождествляемые с микрооперациями, и в качестве логических условий используются булевы переменные *х*1,...,*х*L. Операторная схема алгоритма наиболее часто представляется в виде граф-схемы алгоритма (ГСА). ГСА определяет вычислительный процесс последовательно во времени, устанавливая порядок проверки логических условий *х*1-*х*L и порядок следования микроопераций *у*1-*у*m.

**СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ И МИКРОПРОГРАММ**

Наиболее наглядно изображать микропрограммы и алгоритмы в виде ориентированного графа, т.н. граф схемы алгоритма (ГСА). Кроме наглядности это дает возможность использовать для анализа и преобразования микропрограмм эффективные методы теории графов. При графическом описании отдельные функции алгоритмов (микрооперации) отображаются в виде условных графических изображений, т.н. вершин. В ГСА обычно используют вершины следующих типов:



- вершина «начало» имеет один выход, входов не имеет. Обозначает начало микропрограммы.

- вершина «конец» имеет любое число входов, выходов не имеет. Обозначает конец микропрограммы.

- операторная вершина имеет любое число входов, один выход. Внутри операторной вершины записывается одна микрокоманда - совокупность микроопераций, допускающих совместное (т.е. одновременное) выполнение.



- условная вершина имеет любое число входов и 2 выхода. Внутри условной вершины записывается булевое выражение, в зависимости от значения которого осуществляется выбор направления дальнейшего выполнения микропрограммы.

- особый вид условной вершины - ждущая - имеет множество входов, 2 выхода, 1 из которых заведен на вход. При попадании в ждущую вершину выход из нее возможен только при выполнении условия Х.

Граф микропрограммы состоит из совокупности перечисленных вершин и дуг, соединяющих выходы одних вершин с входами других. Соединение вершин и направление дуг графа определяют исходя из алгоритма операции, описываемого графом, и структуры операционного автомата.

Сама микропрограмма и ее граф должны быть корректны, т.е. отвечать следующим условиям:

1. В графе должна быть только одна начальная и одна конечная вершина.

2. В любую вершину графа должен вести по крайней мере один путь из начальной вершины.

3. Из каждого выхода любой вершины графа должен существовать по

крайней мере один путь в конечную вершину.

4. При всех возможных значениях логических условий и используемых слов должен существовать путь из начальной вершины в конечную.



Пример ГСА представлен на рисунке:

ГСА на рис.43 называется содержательной, т.к. внутри вершин записаны в явном виде микрооперации и логические условия. Если же каждую микрооперацию обозначить символами Yi, a логические условия через Xi, то получится так называемая кодированная ГСА (рис.44 ). Для правильного восприятия микропрограммы, заданной в виде кодированной ГСА, необходимо знать соответствия между Yi, Xi и содержанием соответствующих микроопераций и логических условий.

 Для записи микроопераций внутри вершин используется так называемый Ф-язык. Подробно с зтим языком можно ознакомиться в последующих курсах «Схемотехника ЭВМ», «Теория и проектирование ЭВМ». Здесь же мы рассмотрим только основные положения этого языка.

В этом языке существуют двоичные константы и переменные: 0010 - константа, A(1:4) - четырехразрядное слово - четырехразрядная двоичная переменная. Например, A(1:4)=1010 означает, что в первом разряде слова A будет 1, во втором - 0 и т.д. A(2:3) - часть слова A, размещенная во втором и третьем разрядах.

Наиболее употребительные операции Ф-языка:

присваивание - A( 0:3 ): = 1000, B( 1:4 ): = A( 5:8 ) и т.д.

инвертирование - A( 0:3 ): = ^ B( 1:4 )

конкатенации - С( 0:6 ): = A( 0:3 ). B( 1:3 )

Пример 1. A( 0:3 ): = 1100 B( 1:4 ): = A( 0:3 ) → B( 1:4 ): = 1100

 2. B( 1:4 ): = 0101 A( 0:3 ): = ^B( 1:4 ) → A( 0:3 ): = 1010

 3. A( 0:3 ): = 1101 B( 1:3 ): = 110 C( 0:6 ): = A( 0:3 ). B( 1:3 ) → C(0:6):=1101110

Запись вида A(2) означает, что здесь рассматривается второй разряд слова A, т.е. это бит, записанный во втором разряде слова A.

 При выполнении операций присваивания необходимо соблюдать равенство разрядов в словах слева и справа от знака присваивания. Операции сложения, логического сложения и т.д. в Ф-языке записываются обычным способом через оператор присваивания:

C(0:n):=A(0:n)+B(0:n)

D(0:n):=A(0:n)vB(0:n) и т.д.

## ОПЕРАЦИОННЫЕ ЭЛЕМЕНТЫ

Согласно принципа микропрограммного управления, любая сложная операция распадается на ряд микроопераций, которые выполняются ОА. Различные микрооперации выполняются элементарными ОА - так называемыми операционными элементами (ОЭ), которые являются составными частями основного ОА.

Под операционным элементом понимают устройство, реализующее одну из следующих функций или их произвольную комбинацию:

1. хранение слова информации С;
2. выполнение некоторых микроопераций, в результате которых вычисляется новое значение слова С;
3. вычисления логического условия, зависящего от слова С;

**Т.о. функция ОЭ определена, если заданы:**

1. описание хранимого или вычисляемого слова;
2. описание множества микроопераций, выполняемых этим элементом;
3. описание вычисляемых этим элементом логических условий.

Для построения ОА ОЭ соединяются между собой с помощью цепей передачи слов информации от выходов одних элементов к входам других.

В зависимости от выполняемых микроопераций ОЭ делятся на разновидности: шина, регистр, счетчик, сумматор, схема сравнения, дешифратор, шифратор и т.д.

***Шина*** - это совокупность цепей, предназначенных для передачи слова информации. Условное обозначение шины представлено на рис.45.



Шины, изображенные на рис.45 называются неуправляемыми шинами.

Управляемые шины представлены на рис.46.



Под действием управляющего сигнала у управляемая шина выполняет микрооперации: у=0 : B(0:3):=0 , y=1 : B(0:3):=A(0:3)

Т.е. при y=1 осуществляется операция передачи. Разрядность шины может быть произвольная, но обычно это 8, 12, 16, 24, 32 и т.д.

***Регистр*** - это операционный элемент, служащий для запоминания слов и обеспечивающий в общем случае выполнение следующих микроопераций:

1. установка регистра в 0 (сброс);
2. прием слова из другого регистра, шины и т.д.;
3. передача слова на другой регистр, шину и т.д.;
4. преобразование кодов хранимых слов в инверсные коды;
5. сдвиг хранимого слова влево или вправо на требуемое число разрядов.



Регистр, выполняющий такие микрооперации, называется многофункциональным. Т.к. регистр предназначен для хранения информации, то он содержит элементы памяти, в качестве которых используются триггеры. Количество триггеров определяет разрядность регистра. Будем обозначать регистр в виде прямоугольника с указанием разрядности (рис.47 ).

 Регистр может состоять из отдельных подрегистров, имеющих самостоятельное значение (рис.48.). На рис.48 представлен регистр, хранящий число в форме с плавающей запятой. В этом регистре три подрегистра: для хранения знака Рг(0), характеристики Рг(1:7), мантиссы Рг(8:31) числа.

Регистр может выполнять ряд микроопераций, например:



Регистр, который выполняет микрооперацию у4 (сдвиг вправо) или у5 (сдвиг влево) называются регистром сдвига.

***Сумматор*** - операционный элемент, выполняющий суммирование кодов чисел. В зависимости от кодов чисел различают сумматоры прямого, обратного, дополнительного кодов. Кроме того, сумматоры бывают комбинационными и накапливающими.

Комбинационный сумматор вырабатывает выходные сигналы суммы и переноса, определяемые комбинацией цифр слагаемых, одновременно поданных на входы сумматора. Данный сумматор не обладает памятью и после снятия сигналов с входов выходные сигналы также исчезают.

Условное обозначение комбинационного сумматора представлено на рис.50.



Накапливающим называется сумматор, который осуществляет сложение слов A и B при подаче их на сумматор одного за другим. В накапливающем сумматоре имеется дополнительный регистр для хранения результата.



Структура и условное обозначение накапливающего сумматора представлены на рис. 51.

***Счетчик*** - операционный элемент, который реализует микрооперацию счета. Микрооперация счета состоит в изменении состояния счетчика (значения хранимого слова) на 1. Кроме того счетчик может выполнить и такие микрооперации: установка в 0 и прием произвольного числа.

Т.е. полный набор возможных микроопераций для счетчика:





Счетчик, выполняющий микрооперацию у1 называется суммирующим, микрооперацию у2 - вычитающим, обе микрооперации - реверсивный.

Дешифратор - операционный элемент, выполняющий функцию преобразования некоторого n-разрядного двоичного кода в унитарный код «один из N». Если N=2n - то такой дешифратор называется полным, если N<2n - то частичным. Таблица истинности простейшего полного дешифратора (n=2) и его условное обозначение приведены в табл. 25. и на рис. 53.

Промышленность может выпускать дешифраторы с инверсными выходами. Для такого дешифратора таблица истинности и условное обозначение имеют вид (табл. 26., рис. 54.)



**СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ ПО ГРАФ-СХЕМЕ АЛГОРИТМА**

Граф-схема алгоритма есть форма представления микропрограммы, которую должно выполнить операционное устройство (ОУ). При построении операционного устройства, как состоящего из операционного (ОА) и управляющего (УА) автоматов, необходимо уметь выделить функции ОА и УА из ГСА. Обычно микропрограмма представляется в виде содержательной ГСА. В этом случае для задания функций ОА необходимо перечислить все выполняемые микрооперации и все проверяемые логические условия данной микропрограммы, а также описать разрядность слов, обрабатываемых операционным устройством. На основании этих данных можно построить ОА методами, которые будут изложены в курсе «Схемотехника ЭВМ». Для инициализации выполнения той или иной микрооперации на ОА должны поступать в нужный согласно ГСА момент времени управляющие сигналы Yi. Обычно при проектировании ОУ принимают определенный способ кодирования микроопераций (чаще всего кодом, содержащим столько разрядов, сколько всего различных микроопераций) и для разработки ОА считают, что УА выдает код микроопераций, которые должны выполниться в данный момент времени.

Для УА важна последовательность выдачи соответствующих кодов микроопераций в зависимости от логических условий, вырабатываемых ОА и анализируемых УА в нужные моменты времени. Если принят способ кодирования микроопераций, то функции УА задаются кодированной ГСА. Поэтому для различных содержательных ГСА , имеющих одинаковую кодированную ГСА, ОА будут различны, но УА будет одним и тем же.

В дальнейшем будем рассматривать синтез только УА и только кодированной ГСА.

***Конечный автомат, интерпретирующий микропрограмму работы дискретного устройства, называется микропрограммным автоматом.*** Одну и ту же ГСА можно интерпретировать как автоматом Мили, так и автоматом Мура.

Абстрактный синтез микропрограммного автомата по ГСА осуществляется в два этапа:

1. Получение отмеченной ГСА.

2. Построение графа автомата или таблиц переходов и выходов.

***СИНТЕЗ АВТОМАТА МИЛИ***

На этапе получения отмеченной ГСА входы вершин, следующих за операторными, отмечают символами a1, a2,.. по следующим правилам:

1) символом а1 отмечают вход вершины, следующей за начальной, а также вход конечной вершины;

2) входы всех вершин следующих за операторными, должны быть отмечены;

3) входы различных вершин, за исключением конечной, отмечаются различными символами;

4) если вход вершины отмечается, то только одним символом.

Ясно, что для проведения отметок потребуется конечное число символов а1,...,am. Результатом первого этапа является отмеченная ГСА, которая служит основой для второго этапа - перехода к графу или таблицам переходов-выходов. Пример ГСА, отмеченной для автомата Мили, представлен на рис. 55.



На втором этапе, из отмеченной ГСА, строят граф автомата или таблицы переходов-выходов. Для этого полагают, что в автомате будет столько состояний сколько символов ai понадобилось при отметке ГСА.

На плоскости рисунка отмечаем все состояния автомата ai. Для каждого из состояний ai определяем по отмеченной ГСА все пути, ведущие в другие состояния ***и проходящие обязательно только через одну операторную вершину****.* Например, из состояния а1(рис.55.) есть переход в состояние a2 (путь проходит через операторную вершину y1 y2) и в состояние a4 (путь проходит через вершину y3 y4). Перехода из a1 в a3 нет, так как на этом пути нет ни одной операторной вершины. Будем считать, что автомат осуществляет переход, например, из a1 в a2 при условии x1 = 0 или x1 (см.ГСА, рис.55. ) и вырабатывает на этом переходе выходные сигналы у1 у2 (то, что записано в проходимой операторной вершине ГСА, рис.55.). Значение условий х2, х3, х4 на этом переходе не оказывает влияния на автомат.

Исключение составляет только путь, ведущий в конечную вершину, он может не содержать ни одной операторной вершины (например, переход из а6 в а1), т.е. не сопровождается выработкой выходных сигналов.

Отмечаем на графе все указанные пути для всех состояний в виде дуг, которым приписываем условия перехода и выходной сигнал, вырабатываемый на этом переходе. Получим граф автомата (рис.55. ).



На этом графе переходам типа а3 →a4, a5 → a1 приписывается условие перехода 1, т.к. эти переходы являются безусловными и выполняются всегда, когда автомат попадает в состояние а3 (или а5). На основании отмеченной ГСА или графа автомата можно построить таблицу переходов-выходов. Для микропрограммных автоматов таблица переходов-выходов строится в виде списка и различаются прямая и обратная таблицы. Для данного автомата прямая таблица представлена в табл. 27., обратная - в табл. 28.

Табл. 27.Прямая таблица переходов- Табл. 28.Обратная таблица перехо-

 выходов автомата Мили дов - выходов автомата Мили

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| am | as | X | Y |  | am | as | X | Y |
| a1 | a2 | x1 | y1y2 |  | a4 | a1 | x2 | y2 |
|  | a4 | x1 | y3y4 |  | a5 |  | 1 | y2 |
| a2 | a2 | x3x2 | y1y2 |  | a6 |  | x4 | - |
|  | a5 | x3 | y2y3 |  | a1 | a2 | x1 | y1y2 |
|  | a6 | x3x2 | y4 |  | a2 |  | x3x2 | y1y2 |
| a3 | a4 | 1 | y3y4 |  | a6 |  | x4 | y1y2 |
| a4 | a1 | x2 | y2 |  | a4 | a3 | x2 | y1y4 |
|  | a3 | x2 | y1y4 |  | a1 | a4 | x1 | y3y4 |
| a5 | a1 | 1 | y2 |  | a3 |  | 1 | y3y4 |
| a6 | a1 | x4 | - |  | a2 | a5 | x3 | y2y3 |
|  | a2 | x4 | y1y2 |  | a2 | a6 | x3x2 | y4 |

В приведенных таблицах am - исходное состояние, aS - состояние перехода, Х - условие (входной сигнал), обеспечивающий переход из состояния am в состояние as, Y - выходной сигнал, вырабатываемый автоматом при переходе из am в aS.

***СИНТЕЗ АВТОМАТА МУРА.***

Для автомата Мура на этапе получения отмеченной ГСА разметка производится согласно следующим правилам:

1) символом а1 отмечается начальная и конечная вершины;

2) различные операторные вершины отмечаются различными символами;

3) все операторные вершины должны быть отмечены;

Пример ГСА, отмеченной для автомата Мура, представлен на рис. 56.



Граф автомата Мура, соответствующий отмеченной ГСА (рис. ), представлен на рис. . Построение его аналогично построению графа для автомата Мили.

#

Таблицы переходов-выходов автомата Мура представлены в табл. 29 (прямая) и табл. 30 (обратная). Обычно для автомата Мура в таблице переходов-выходов дополнительный столбец для выходных сигналов не используется и выходной сигнал записывается в столбце, где указывается исходное состояние am или состояния перехода aS.

Табл. 29.Прямая таблица переходов Табл. 30.Обратная таблица переходов

автомата Мура. автомата Мура.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| am(Y) | as­ | X |  | am | as­(Y) | X |
| a1(--) | a2 | x1 |  | a6 | a1(-) | x4 |
|  | a3 | x1 |  | a7 |  | 1 |
| a2(y1y2) | a2 | x3 x2 |  | a1 | a2(y1y2) | x1 |
|  | a5 | x3 |  | a2 |  | x3 x2 |
|  | a6 | x3 x2 |  | a6 |  | x4 |
| a3(y3y4) | a4 | x2 |  | a1 | a3(y3y4) | x1 |
|  | a7 | x2 |  | a4 |  | 1 |
| a4(y1y4) | a3 | 1 |  | a3 | a4(y1y4) | x2 |
| a5(y2y3) | a7 | 1 |  | a2 | a5(y2y3) | x3 |
| a6(y4) | a1 | x4 |  | a2 | a6(y4) | x3 x2 |
|  | a2 | x4 |  | a3 | a7(y2) | x2 |
| a7(y2) | a1 | 1 |  | a5 |  | 1 |

Получением графа или таблиц переходов-выходов заканчивается этап абстрактного синтеза микропрограммного автомата. Как и для конечных автоматов, на этапе абстрактного синтеза можно выполнить минимизацию количества внутренних состояний автомата.

***СТРУКТУРНЫЙ СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ***

Структурный синтез микропрограммных автоматов после получения графа или таблицы переходов-выходов в принципе аналогичен каноническому методу синтеза цифровых автоматов, рассмотренному ранее. Однако существуют и определенные особенности в первую очередь связанные с тем, что для реальных автоматов количество элементов памяти и входных сигналов может достигать десяти и более. Функции возбуждения и выходных сигналов трудно поддаются минимизации да и практически минимизация не дает существенного упрощения этих функций при большом количестве переменных. Поэтому минимизация практически не используется при синтезе микропрограммных автоматов.

 При выполнении структурного синтеза строят так называемые структурные таблицы переходов и выходов, которые также могут быть как прямыми так и обратными.

 Рассмотрим этапы структурного синтеза на конкретных примерах.

***СТРУКТУРНЫЙ СИНТЕЗ АВТОМАТА МИЛИ***

 Выполним структурный синтез микропрограммного автомата Мили, заданного своей таблицей переходов-выходов (табл. 27 или табл. 28). В качестве примера синтез будем выполнять по прямой таблице (табл. 27).

1. В исходном автомате количество состояний М=6, следовательно, число элементов памяти

 m = ] log 2 M [ = ] log 2 6 [ = 3.

 Пусть для синтеза используются JK триггеры.

2. Кодируем внутренние состояния автомата, используя для этого карту Карно (рис.57.) и по возможности метод соседнего кодирования. Рекомендуется самостоятельно закодировать состояние с помощью эвристического алгоритма.



3. Строим прямую структурную таблицу переходов-выходов автомата Мили (табл. 31). В данной таблице в столбцах k(am) и k(as) указывается код исходного состояния и состояния перехода соответственно. В столбце функций возбуждения ФВ указывается те значения функций возбуждения, которые на данном переходе обязательно равны 1. Остальные (т.е. равные 0 или принимающие неопределенные значения) не указываются. Это эквивалентно тому, что всем неопределенным значениям функций возбуждения приписывается значение 0, что в общем случае не дает минимальной функции, однако в реальных автоматах минимизация обычно не делается в виду ее неэффективности. Предлагается самостоятельно построить структурную таблицу переходов с указанием всех значений функций возбуждения (в том числе и неопределенных), выполнить минимизацию и сравнить результаты с приведенными ниже.

Табл. 31. Структурная таблица переходов-выходов автомата Мили.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Am | K(am) | as | K(as) | X | Y | ФВ |
| a1 | 000 | a2 | 010 | x1 | y1y2 | J2 |
|  |  | a4 | 001 | x1 | y3y4 | J3 |
| a2 | 010 | a2 | 010 | x3x2 | y1y2 | - |
|  |  | a5 | 110 | x3 | y2y3 | J1 |
|  |  | a6 | 011 | x3x2 | y4 | J3 |
| a3 | 101 | a4 | 001 | 1 | y3y4 | K1 |
| a4 | 001 | a1 | 000 | x2 | y2 | K3 |
|  |  | a3 | 101 | x2 | y1y4 | J1 |
| a5 | 110 | a1 | 000 | 1 | y2 | K1K2 |
| a6 | 011 | a1 | 000 | x4 | - | K2K3 |
|  |  | a2 | 010 | x4 | y1y2 | K3 |

4. Для получения функций возбуждения поступаем следующим образом. Выражение для каждой функции получается в виде логической суммы произведений вида aiX, где ai - исходное состояние, X-условие перехода. Для упрощения полученных выражений выполняем все возможные операции склеивания и поглощения:

 J1 = a2x3 + a4x2 K1 = a3 + a5

 J2 = a1x1 K2 = a5 + a6x4

 J3 = a1x1 + a2x3x2 K3 = a4x2 + a6x4 + a6x4 = a6 + a4x2

5. Для получения функций выходов поступаем аналогично:

 y1 = a1x1 + a2x3x2 + a4x2 + a6x4

 y2 = a1x1 + a2x3x2 + a2x3 + a4x2 + a5 + a6x4

 y3 = a2x3 + a3 + a1x1

 y4 = a1x1 + a2x3x2 + a3 + a4x2

6. Для построения функциональной схемы автомата по полученным выражениям необходимо либо заменить ai его значениями через Q1Q2Q3 либо получить сигнал, соответствующий ai. Обычно используют второй способ и для получения сигнала ai применяют так называемый дешифратор состояний, на вход которого поступают сигналы с выходов элементов памяти Q1Q2Q3. Кроме того, при построении схемы стараются выделить общие части, встречающиеся в функциях возбуждения или выходных сигналах. В этом случае окончательная система уравнений, по которым строится схема, будет иметь вид:

 A = a2x3x2+J2 ; J1 = D + B ; y1 = A + B + E ;

 B = a4x2 ; K1 = a3 + a5; y2 = A + D + C + a5 + E ;

 C = a4x2 ; J2 = a1x1 ; y3 = D + F + a3 ;

 D = a2x3 ; K2 = a5 + a6x4 ; y4 = a3 + B + J3;

 E = a1x1  ; K3 = a6 + C ;

 F = a1x1  J3 = F+a2x3x2

Функциональная схема автомата, построенная на основании полученных уравнений, представлена на рис. 58.



***СТРУКТУРНЫЙ СИНТЕЗ АВТОМАТА МУРА***

Выполним структурный синтез микропрограммного автомата Мура, заданного своей таблицей переходов-выходов (табл.29 или табл. 30). В качестве примера синтез будем выполнять по обратной таблице (табл. 32).

# 1. В исходном автомате количество состояний М=7, следовательно число элементов памяти

m = ] log 2 M [ = ] log 2 7 [ = 3

Пусть для синтеза используется D-триггеры.

2. Кодируем внутренние состояния автомата, используя алгоритм кодирования для D-триггеров. Количество переходов в данное состояние легко определяется из обратной таблицы: a1 ~ 2, a2 ~ 3, a3 ~ 2, a4 ~ 1, a5 ~ 1, a6 ~ 1, a7 ~ 2.

Поэтому коды состояний следующие:

a2-000, a1-001, a3-010, a7-100, a4-011, a5-101, a6-110.

3. Строим структурную таблицу переходов - выходов автомата Мура.

Табл. 32. Структурная таблица переходов - выходов автомата Мура.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| am | K(am) | as(Y) | K(as) | X | ФВ |
| a6 | 110 | a1(-) | 001 | x4 | D3 |
| a7 | 100 |  |  | 1 | D3 |
| a1 | 001 | a2(y1y2) | 000 | x1 | - |
| a2 | 000 |  |  | x3x2 |  |
| a6 | 110 |  |  | x4 |  |
| a1 | 001 | a3(y3y4) | 010 | x1 | D2 |
| a4 | 011 |  |  | 1 | D2 |
| a3 | 010 | a4(y1y4) | 011 | x2 | D2D3 |
| a2 | 000 | a5(y2y3) | 101 | x3 | D1D3 |
| a2 | 000 | a6(y4) | 110 | x3x2 | D1D2 |
| a3 | 010 | a7(y2) | 100 | x2 | D1 |
| a5 | 101 |  |  | 1 | D1 |

Построение таблицы выполняется аналогично автомату Мили.

4. Выражения для функций возбуждения получаются в виде суммы произведений aiх, где ai-исходное состояние, х - условие перехода.

D1 = a2x3 + a2x3x2 + a3x2 + a5

D2 = a1x1 + a4 + a3x2 + a2x3x2

D3 = a6x4 + a7 + a3x2 + a2x3

 или

A = a3x2

B = a2x3x2

D1 = a2x3 + B + a3x2 + a5

D2 = a1x1 + a4 + A + B

D3 = a6x4 + a7 + A + a2x3

5. Выражения для выходных сигналов автомата Мура получаем, исходя из того, что эти сигналы определяются только внутренним состоянием автомата.

y1 = a2 + a4

y2 = a2 + a5 + a7

y3 = a3 + a5

y4 = a3 + a4 + a6

6. Для построения функциональной схемы автомата как и в предыдущем случае используем дешифратор состояний. Схема представлена на рис. 61 .

### ***ЗАМЕЧАНИЯ.***

1. При синтезе микропрограммных автоматов изложенным методом получаемые выражения для функций возбуждения не всегда являются минимальными и в некоторых случаях могут быть упрощены. В частности, можно доопределить функции возбуждения на некоторых наборах единичным значением (а не нулевым, как было ранее) и выполнить все операции склеивания. Например, в синтезированном ранее автомате Мили таким образом можно получить значение *k*1=1. Рекомендуется в этом убедиться самостоятельно.

Для упрощения схем автоматов можно также предварительно упростить ГСА, уменьшив количество вершин или узлов методами, изложенными в / /.

Автомат Мили в течении такта сохраняет свое состояние и лишь в конце его переходит в новое. Пока автомат находится в данном состоянии Ai он вырабатывает выходные сигналы y=f(Ai,x), где *х* - сигналы, подаваемые на вход автомата в течение данного такта. В связи с этим автомат Мили может интерпретировать микропрограмму корректно только в том случае, если для любого перехода выполняется условие независимости логических условий от результатов выполнения микроопераций на данном переходе.

Условие независимости нарушается, если на некотором переходе из *а*m в *а*s под действием сигнала *х* с выработкой *у*i наблюдается функциональная зависимость *х* = *f*(*у*i). В таком случае выполнение микрооперации *у*i может привести к изменению значения *х* и автомат, находясь в состоянии *а*m, и, реагируя на набор входных сигналов, сформирует набор выходных сигналов, не соответствующий микропрограмме. Чтобы избежать этого, можно использовать следующие способы:

 **1)** запомнить значение сигналов *х* на промежуток времени, равный длительности такта;

 **2)** ввести в автомат дополнительные состояния;

 **3)** реализовать автомат по схеме Мура.

Запоминание значений сигналов *х* в течение такта осуществляется операционным автоматом с помощью дополнительных элементов памяти – триггеров. Интерпретация микропрограммы автоматом Мура рассматривалась ранее. Введение дополнительных состояний иллюстрируется рис. 59 .

В исходном автомате (рис. 59.а) есть переходы из *аi* в *аj* под действием сигналов *х* и *х* с выработкой *y*1 и *y*2 соответственно и имеет место *х* = *f*(*y*1, *y*2). Действительно, введение вспомогательных состояний *а*k и *а*l позволяет устранить возможную ошибку в выдаче выходных сигналов. На переходах *аi аk* или *аiаl* выходные сигналы не вырабатываются. Переходы *аi аk* или *аiаl* являются безусловными с выработкой *y*1 или *y*2 соответственно. В таком случае изменение значения *х,* в результате выполнения микроопераций *y*1 или *y*2, не повлияет на выходной сигнал, вырабатываемый автоматом, так как на переходах *аi аk* или *аiаl* выходной сигнал уже не зависит от *х*.

****

**Синтез управляющего автомата Мура
на базе регистра сдвига.**

Кроме рассмотренного ранее канонического метода, существуют и другие методы синтеза управляющих автоматов, среди которых наиболее широко используется синтез на базе регистра сдвига. Этот метод позволяет при построении схемы отказаться от дешифратора, т.к. состояния кодируются унитарным кодом. В автомате количество элементов памяти выбирается равным количеству внутренних состояний. В каждый момент времени только один триггер находится в 1, остальные в 0. Обычно при синтезе на базе регистра сдвига используются *D*-триггеры. Очень эффективен данный метод для так называемых линейных микропрограмм, т.е. микропрограмм без ветвлений (отсутствует логические условия). Рассмотрим пример синтеза управляющего автомата Мура данным методом. Пусть закодированная ГСА микропрограммы имеет вид рис. 60. Разметив данную ГСА для автомата Мура, получаем семь состояний. Следовательно число триггеров *m*=7. Выполним синтез с использованием *D*-триггеров. Закодируем состояния унитарным кодом: *a*1=1000000, *a*2=0100000,..., *a*7=0000001. Обратная структурная таблица переходов-выходов для данного автомата представлена в таблице.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***a*m** | ***Ka*m** | ***a*s(*y)*** | ***Ka*s** | ***x*** | ***ФВ*** |
| *а*6 | 0000010 | *а*1(-) | 1000000 | 1 | *D*1 |
| *а*7 | 0000001 |  |  | 1 | *D*1 |
| *а*1 | 1000000 | *а*2(*y*1 *y*2) | 0100000 | 1 | *D*2 |
| *а*2 | 0100000 | *а*3(*y*2) | 0010000 | 1 | *D*3 |
| *а*3 | 0010000 | *а*4(*y*3 *y*4) | 0001000 | 1 | *D*4 |
| *а*4 | 0001000 | *а*5(*y*2) | 0000100 |  | *D*5 |
| *а*5 | 0000100 | *а*6(*y*3) | 0000010 | 1 | *D*6 |
| *а*4 | 0001000 | *а*7(*y*4) | 0000001 | *x* | *D*7 |



На основании структурной таблицы записываем выражения для выходных сигналов *yi* и функций *Di* :

*D*1 = *a*6 + *a*7 *y*1 = *a*2

*D*2 = *a*1 *y*2 = *a*2 + *a*3 + *a*5

*D*3 = *a*2 *y*3 = *a*4 + *a*6

*D*4 = *a*3 *y*4 = *a*4 + *a*7

*D*5 = *a*4

*D*6 = *a*5

*D*7 = *a*4⋅*x*

Т.к. состояния автомата закодированы унитарным кодом, то можно отождествить каждое состояние с выходом соответствующего триггера, т.е. принять *аi*=*Qi*. Для принятого способа кодирования переход из одного состояния в другое как бы сопровождается сдвигом кода, за-

писанного в семиразрядном регистре. Этим и объясняется название метода. Функциональная схема автомата Мура, построенная по полученным уравнениям, приведена на рисунке 62. При определенных навыках синтез автомата Мура на базе регистра сдвига выполняется непосредственно по отмеченной ГСА без построения структурной таблицы переходов-выходов.



