С автором можно связаться по адресу: aurora@peterlink.ru, "для Шалыто".

ЖУРНАЛ "Промышленные АСУ и контроллеры". 2000. №4. С.45-50.

Реализация алгоритмов логического управления программами на языке функциональных блоков

© 2000 г. А. А. Шалыто

Санкт-Петербург, Федеральный научно-производственный
центр – ГУП "НПО "Аврора"",

С.-Петербургский гос. ин-т точной механики и оптики
(техн. ун-т)

Предлагаются методы непосредственного построения программ на языке функциональных блоков по графам переходов. Граф, описывающий алгоритм логического управления, предлагается использовать в качестве сертификационного теста для подтверждения того, что построенная "схема" (программа) реализует этот граф.

1. Постановка задачи

Одним из языков программирования программируемых логических контроллеров (ПЛК) по стандарту IEC 1131-3 [1] является язык функциональных блоков (Function Block Diagram). Этот язык в входит в состав программного обеспечения ряда программируемых управляющих систем [2] и многих ПЛК [3,4], выпускаемых ведущими в области автоматизации фирмами мира.

Этот язык входит также в ряд программных продуктов, предназначенных для программирования на нем IBM PC совместимых компьютеров [5,6].

Подход, используемый в этих разработках, обладает, по мнению автора, одним недостатком – язык функциональных блоков применяется в них как исполняемый язык спецификаций [7]. При этом исполняемой является функциональная схема, построенная из библиотечных блоков.

Однако при таком подходе открытыми остаются ответы на вопросы: откуда "берутся" функциональные схемы и тесты для их проверки?

Так как известные подходы не отвечают на эти вопросы, то говорить о построении программ на этом языке с гарантированным качеством при применении указанных подходов не приходится.

Перечисленные выше недостатки могут быть устранены при использовании **SWITCH-технологии** [8,9], в рамках которой предлагается использовать два уровня языков: алгоритмизации (спецификации) и программирования.

В качестве языка спецификаций в этой технологии предлагается применять язык графов переходов, позволяющий описывать **поведение**, а не реализацию управляющего автомата. При этом решаемая задача описывается графом переходов, соответствующим выбранному типу автомата (например, автомату Мура), по которому искомая функциональная схема из библиотечных элементов строится **формально**. Такой подход позволяет применять граф переходов в качестве сертификационного теста для подтверждения того, что построенная "схема" (программа) реализует этот граф.

В [8] изложены различные методы построения функциональных схем по графу переходов, которые основаны на формальном получении по этому графу системы булевых

формул (СБФ), по которой также формально строится искомая функциональная схема в выбранном базисе библиотечных элементов.

Однако и такой подход обладает недостатком, так как является двухэтапным. Но в рамках предложенной технологии может быть устранен, если функциональную схему строить формально непосредственно по графу переходов. Это можно осуществить, используя стандартные для логического синтеза элементы памяти – триггеры [8]. Соответствующий метод синтеза функциональных схем изложен в разд.2.

Однако более "понятные" схемы (программы) строятся одноэтапно при не только формальном, но и **изоморфном** переходе от графа переходов к функциональной схеме. При этом схема должна быть построена так, чтобы она была внешне похожа на используемый граф переходов, что, например, упрощает ее модификацию при необходимости.

Это может быть обеспечено, если отказаться от триггеров и наряду с традиционными логическими элементами (например, И, ИЛИ, НЕ) применять также логические и **цифровые** мультиплексоры, охваченные обратной связью, а также преобразователи "десятичный код – двоичный код".

Указанные типы элементов разрешены стандартом [1] и входят в состав библиотеки [2], но, к сожалению, отсутствуют в других, известных автору, библиотеках. В открытые библиотеки такие элементы всегда могут быть введены. Соответствующий метод синтеза функциональных схем изложен в разд.3. Идея этого метода впервые описана в [10].

2. Построение функциональной схемы непосредственно по графу переходов

Для определенности выберем в качестве элементов памяти R-триггеры, для каждого из которых характерно, что если одновременно на вход сброса (вход "R") и на вход установки (вход "S") поданы единичные значения управляющих переменных, то он устанавливается в состояние "0".

Предлагаемый метод основан на фиксации в триггерах **изменений** значений выходных и внутренних переменных, происходящих при переходах в графе переходов автомата без выходного преобразователя, вершины которого закодированы полными двоичными кодами [8].

На рис.1 в качестве примера приведен граф переходов счетного триггера, по которому функциональная схема (рис.2) построена непосредственно. Отметим, что в этом графе противогоночный код не применяется, так как программная реализация является синхронной, что обеспечивается расстановкой номеров рядом с элементами схемы, определяющими порядок их срабатывания (выполнения соответствующих фрагментов программы) [2].

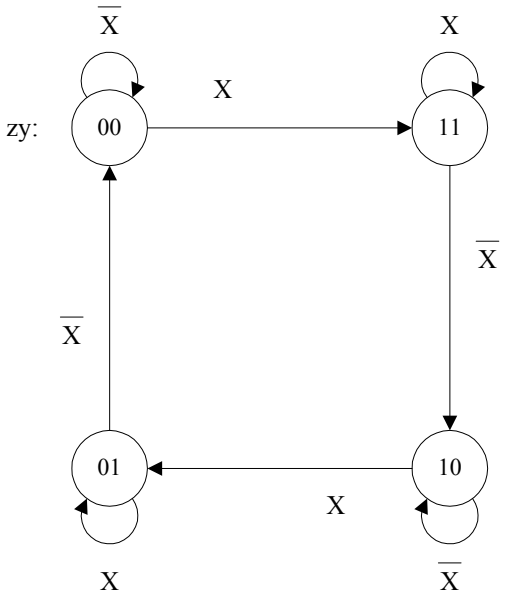

Рис. 1

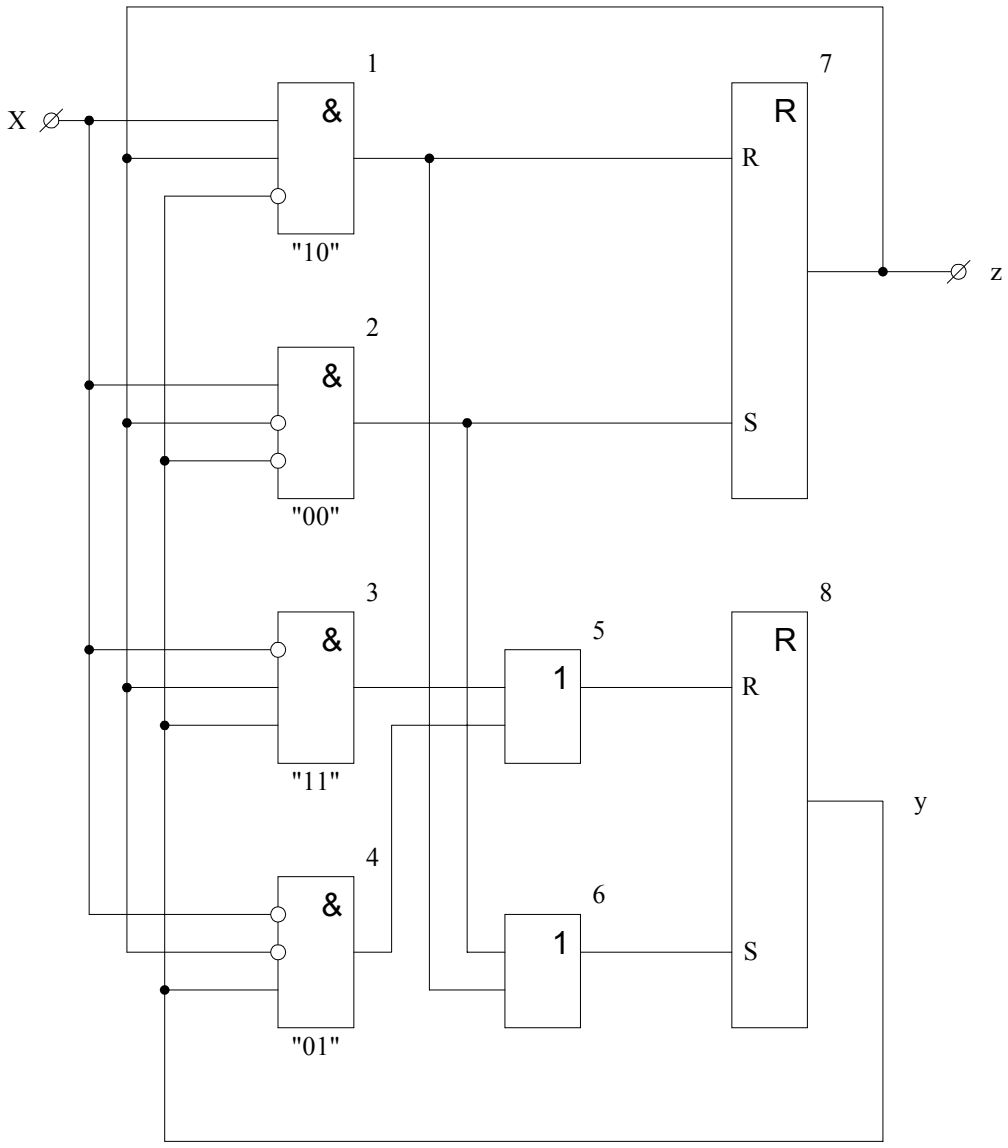

Рис. 2

Эта схема строится следующим образом. Из рассмотрения графа переходов следует, что для состояния "z=1, y=0" при x=1 формирование значений переменных, определяющих следующее состояние "z=0, y=1" связано с изменением значений каждой из этих переменных: первая должна сбрасываться (R_1 =1), а вторая устанавливаться (S_2=1).

При переходе из состояния "z=0, y=0" в состояние "z=1, y=1", также происходящем при x=1, должны измениться значения обеих переменных, определяющих состояние – S_1=1, S_2=1.

При переходе из состояния "z=1, y=1" в состояние "z=1, y=0", происходящем при x=0, должно измениться значение только одной из переменных, определяющих состояние – R_2=1.

При переходе из состояния "z=0, y=1" в состояние "z=0, y=0" , также происходящем при x=0, должно измениться значение только одной из переменных, определяющих состояние – R_2=1.

Сертификация полученной функциональной схемы (программы) может выполняться с помощью графа переходов, по которому она построена.

3. Изоморфное построение функциональной схемы непосредственно по графу переходов

При использовании предлагаемого метода применяется граф переходов автомата Мура, вершины которого кодируются многозначно [8]. Этот граф переходов непосредственно и изоморфно реализуется одной из стандартных схем, рассмотренных ниже, которые строятся из

традиционных логических элементов, логических (Л) и цифровых (Ц) мультиплексоров, а также преобразователей "десятичный код – двоичный код" (Д/В).

Логический мультиплексор имеет m логических управляющих входов, 2^m цифровых информационных входов и один цифровой выход, а цифровой мультиплексор – один цифровой управляющий и S цифровых информационных входов (S – число вершин в графе переходов (число состояний в автомате)), а также один цифровой выход.

На логические входы этих элементов подаются константы 0 и 1, а на цифровые – константы 0,1,2,...,S-1. При этом на цифровых выходах также формируются константы 0,1,2,...,S-1.

Обратим внимание на отличие этих элементов от элементов, традиционно называемых мультиплексорами, все входы и выходы которых являются логическими (двоичными) [11].

Предлагается использовать три варианта стандартных схем, первый из которых предполагает наличие логических мультиплексоров при $m_i = 1$, второй – при $m_i = n_i$, а третий – при $m_i=k_i$, где n_i – число переменных в булевых формулах, помечающих дуги, исходящие из i-й вершины графа переходов; k_i – число дуг, исходящих из i-й вершины.

В **первом** случае стандартную схему можно описать следующим образом. Комбинационная схема строится из традиционных логических элементов и реализует формулы, указанные на всех дугах графа переходов, за исключением петель. Выходы этой схемы подключены к управляющим входам логических мультиплексоров, а выходы подсхемы, состоящей из этих мультиплексоров, – к информационным входам первого цифрового мультиплексора, выход которого **обратной** связью соединен (для реализации петель в вершинах графа переходов) со своим управляющим входом и первыми, помеченными символом "0", информационными входами логических мультиплексорами (входы указанной подсхемы), а **прямой** связью – с управляющим входом второго цифрового мультиплексора, на информационные входы которого подаются десятичные эквиваленты наборов двоичных значений выходных переменных в соответствующих вершинах графа переходов. Выход второго цифрового мультиплексора подключен ко входам преобразователя, осуществляющего преобразование десятичного кода в значения двоичных выходных переменных. На остальные информационные входы логических мультиплексоров подаются десятичные эквиваленты номеров вершин, в которые осуществляются соответствующие переходы в графе переходов.

Оценим число элементов в такой схеме. Если число различных входных переменных в пометках дуг графа переходов равно N, то число инверторов в схеме не превысит N. Если общее число букв во всех булевых формулах на дугах (за исключением петель) графа переходов равно H, то число логических элементов, реализующих эти формулы, не превышает H - L, где L – число дуг (без учета петель) в графе переходов. Это объясняется тем, что для реализации i-й (i = 1,...,L) формулы из h_i букв требуется не более $h_i - 1$ двухвходовых элементов без учета инверсий, реализуемых в первом слое схемы. Так как каждый переход (дуга) в графе переходов реализуется одним логическим мультиплексором, то их общее число равно L. Так как в схему входят два цифровых мультиплексора и один преобразователь, общее число элементов в ней определяется соотношением:

$$Э_1 \le N + (H - L) + L + 3 = N + H + 3.$$

Использование изложенного подхода позволяет реализовать граф переходов (рис.3) функциональной схемой (рис.4) из 11 элементов ($Э_1 \le 3 + 6 + 3 = 12$).

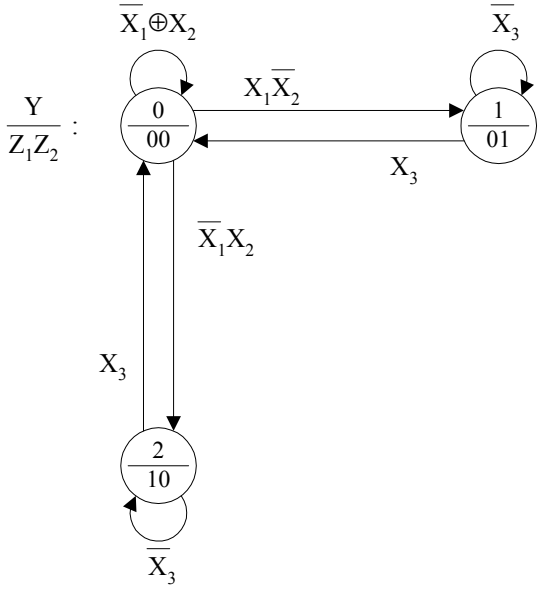

Рис. 3

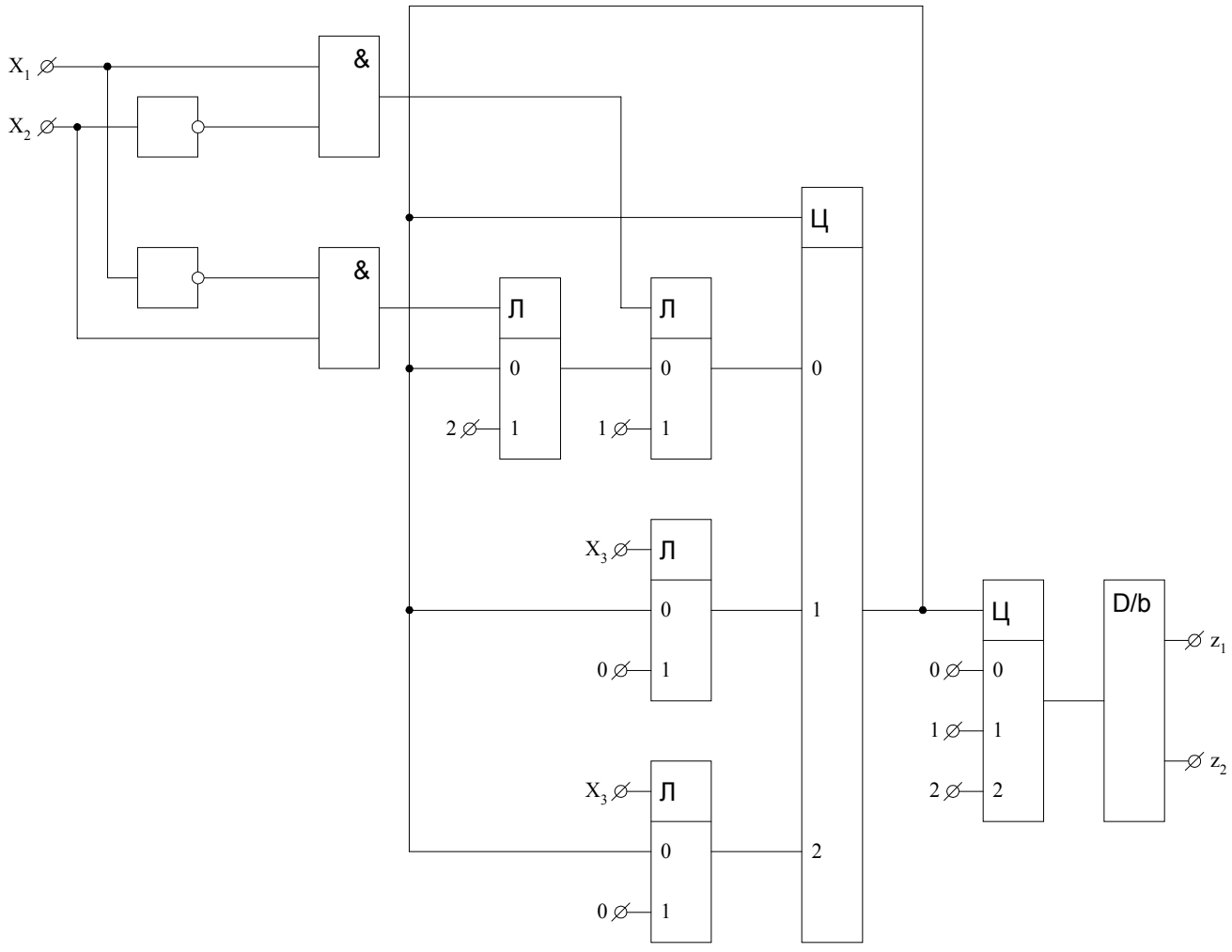

Рис. 4

Часть схемы этого типа, построенная из традиционных логических элементов, может быть упрощена, если противоречивость в графе переходов устранять не ортогонализацией как в рассмотренном примере, а расстановкой приоритетов [8]. При этом чем выше приоритет дуги, исходящей из j-й вершины графа переходов, тем ближе логический мультиплексор,

соответствующий этой дуге, располагается к j-му информационному входу первого цифрового мультиплексора.

Вторая разновидность стандартных схем отличается от первой тем, что комбинационная схема из традиционных логических элементов не строится, а j-ый (j = 1,...,S) логический мультиплексор реализует все булевы формулы, помечающие дуги, исходящие из j-ой вершины графа переходов. Это удается осуществить в том случае, когда все указанные формулы взаимно ортогональны. Таким образом, число логических мультиплексоров в схеме равно S, а общее число элементов в схеме в этом случае

$$Э_2 = S + 3.$$

На рис.5 приведена схема из шести элементов, реализующая граф переходов, приведенный на рис.3.

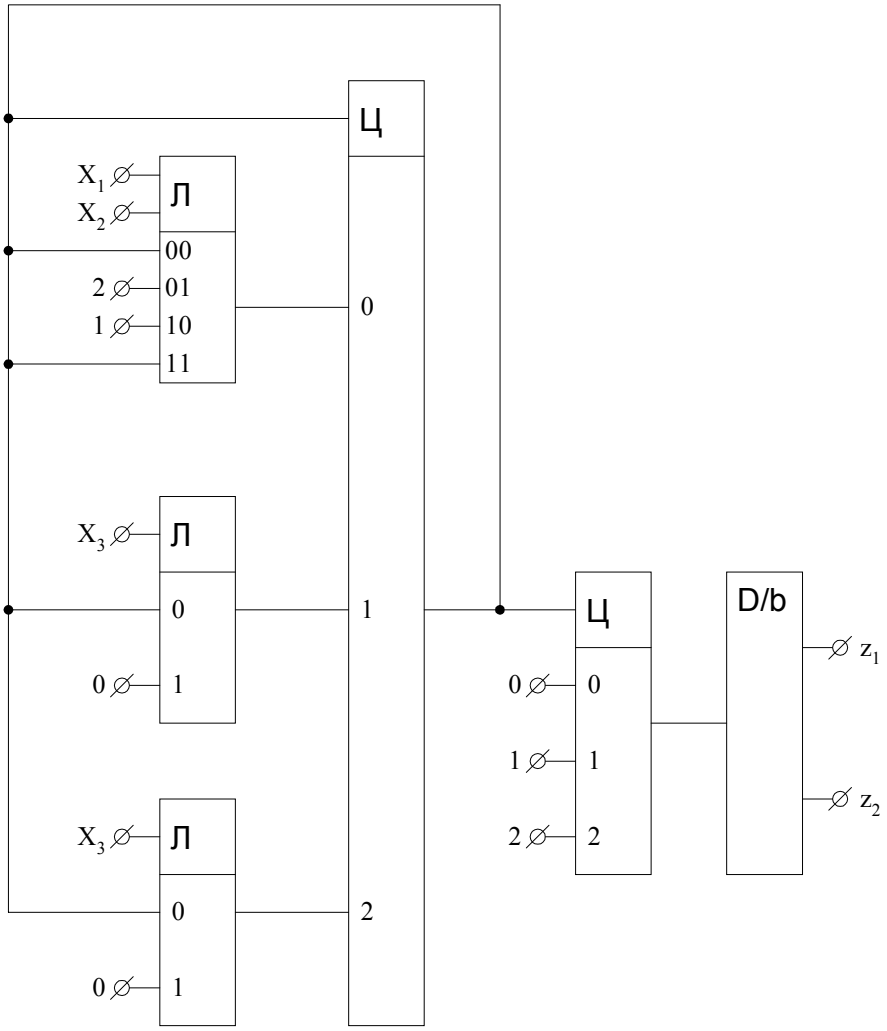

Рис. 5

Третий вариант стандартных схем является промежуточным между рассмотренными. Он содержит, как и вторая разновидность схем, S логических мультиплексоров, но в силу того что в этом случае число информационных входов i-го логического мультиплексора ограничено и равно $M = 2^a$, где $a =]\log_2 k_i [$, то приходится строить комбинационную схему из традиционных логических элементов так же, как в первом случае. Свободные информационные входы логических мультиплексоров подключаются к выходу первого цифрового мультиплексора, что обеспечивает сохранение состояний автомата. Число элементов в этом случае оценивается соотношением:

$$Э_3 \le N + H - L + S + 3.$$

Граф переходов, представленный на рис.6, реализуется схемой третьего типа (рис.7) из 10 элементов ($Э_3 \le 4 + 7 - 4 + 3 + 3 = 13$).

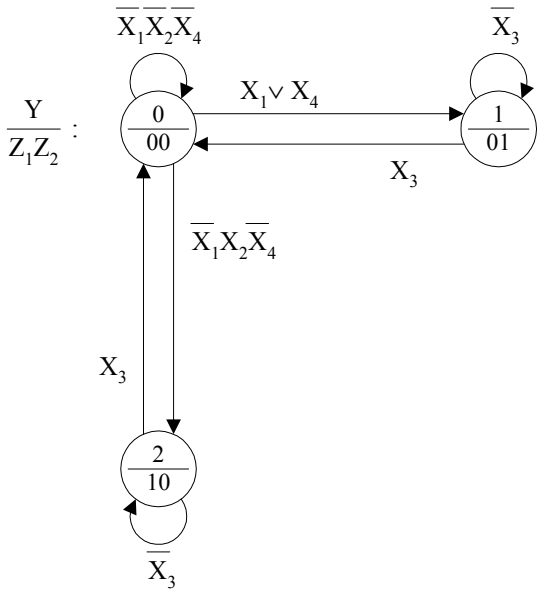

Рис. 6

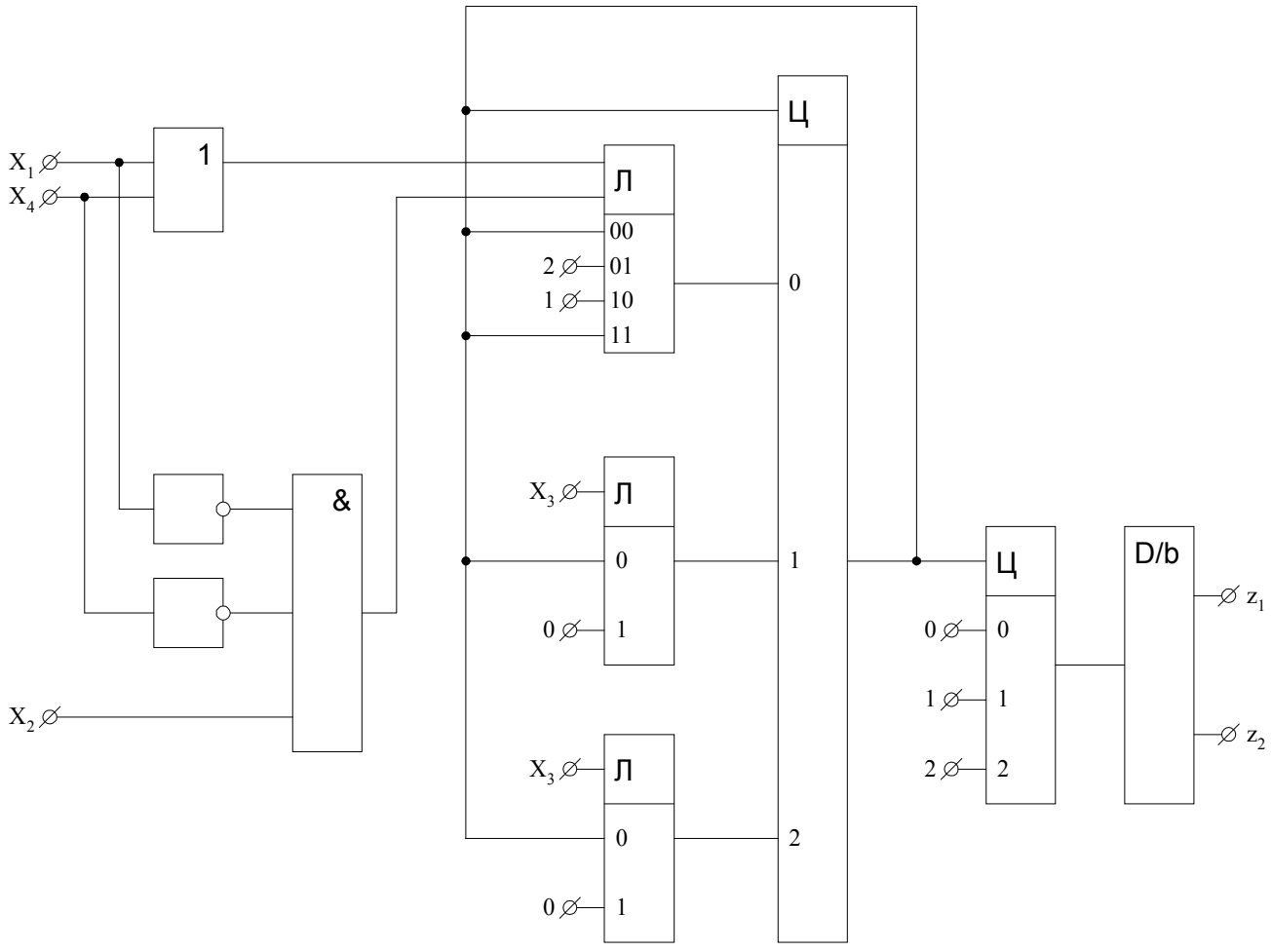

Рис. 7

Сравнивая рассмотренные типы схем, можно утверждать, что второй тип схем является наиболее однородным, а первый – наиболее просто моделируется программой на языке Си при

использовании алгоритмической конструкции, соответствующей автомату Мура второго рода и построенной с помощью двух конструкций **switch** [8]. При этом каждой конструкции switch соответствует цифровой мультиплексор в схеме, а каждой конструкции if – соответствующий логический мультиплексор. Это позволяет моделировать программу, написанную на языке функциональных блоков, изоморфной программой на широко распространенном алгоритмическом языке высокого уровня.

Изложенный метод построения функциональных схем позволяет предложить **новый подход** к их сертификационному тестированию, которое предлагается проводить в два этапа. На первом этапе наблюдая за значениями **одной** внутренней переменной Y, кодирующей состояния автомата, в динамике по графу переходов проверяется правильность реализации программой каждого из переходов в графе, а на втором - в статике по графу переходов анализируется правильность формирования программой значений выходных переменных, указанных в каждой вершине графа.

4. Реализация системы графов переходов

Рассмотрим вопрос о реализации алгоритмов, заданных несколькими взаимосвязанными графами переходов, например двумя. В этом случае для упрощения схемы обмен между графами предлагается осуществлять не с помощью значений многозначных переменных, каждая из которых кодирует вершины соответствующего графа переходов, как это было предложено в [8] для программной реализации на алгоритмических языках, а путем введения дополнительных двоичных выходных переменных.

Число таких переменных равно числу состояний (вершин), используемых во взаимных блокировках. Каждая из этих переменных принимает значение "1" в одном состоянии, применяемом для блокировки, и "0" во всех остальных состояниях.

На рис.8 приведены два взаимосвязанных графа переходов, реализующие алгоритм совместного управления двумя клапанами с памятью, с дополнительно введенными выходными переменными z_5 и z_6 (для первого и второго графов переходов соответственно). Эти переменные обеспечивают взаимодействие (по данным) между графами переходов. На рис.9 приведена функциональная схема, реализующая указанные графы.

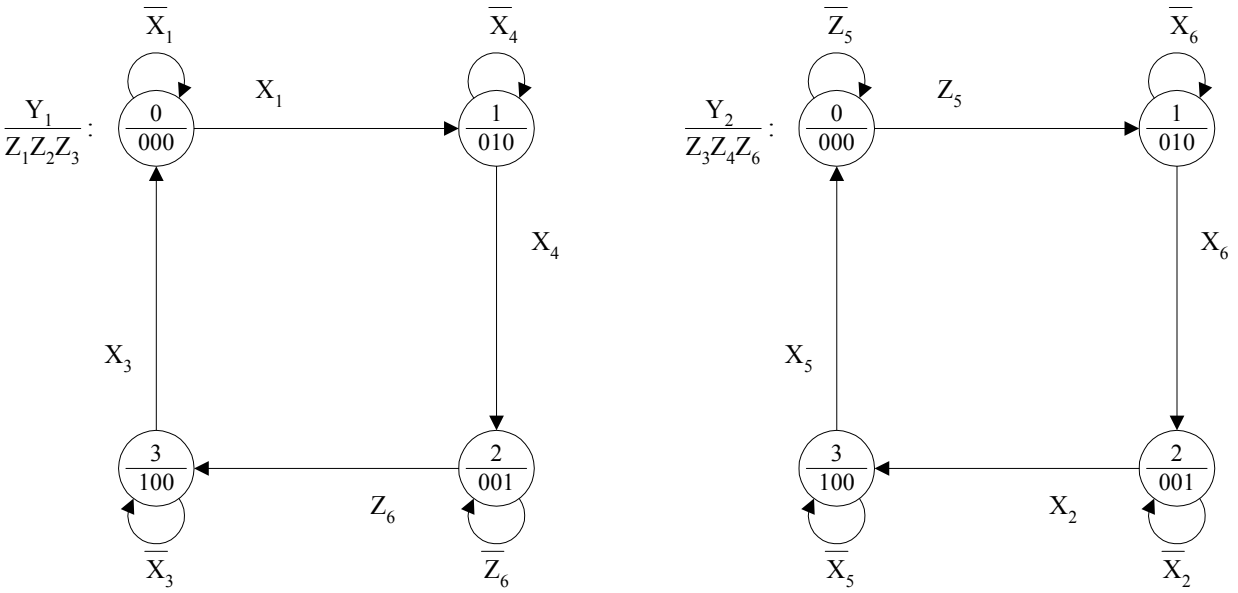

Рис. 8

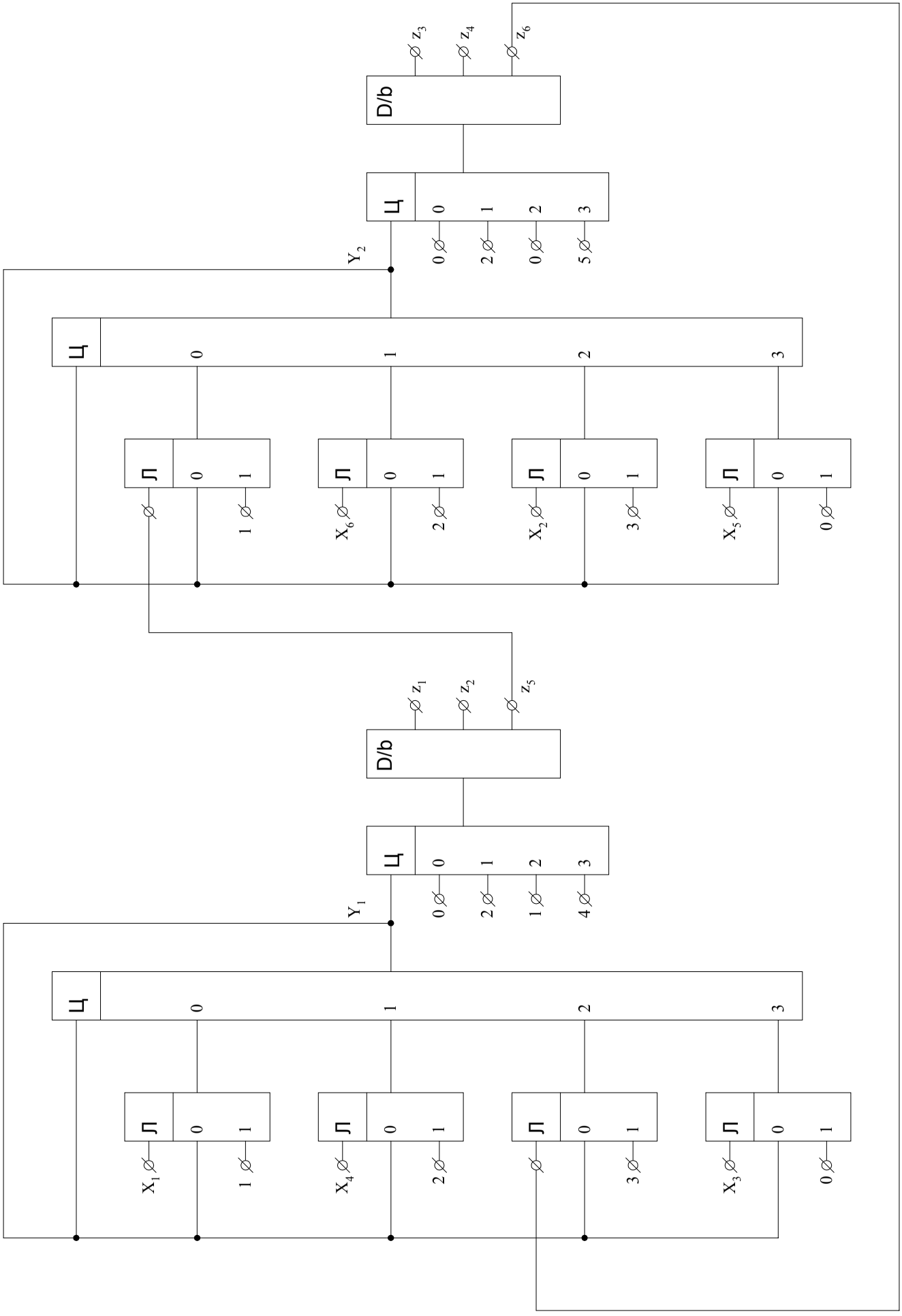

Рис. 9

Для тестирования программы, реализующей систему взаимосвязанных графов переходов, для последней первоначально должен быть построен граф достижимых маркировок [8], который в дальнейшем предлагается использовать в качестве сертификационного теста.

5. Заключение

В завершение работы отметим, что подход, рассмотренный в разд.3,4 обеспечивает построение более компактных схем по сравнению со схемами, построенными из программно реализованных двоичных мультиплексоров, при использовании которых многозначное кодирование не может быть применено.

Этот подход следует использовать во всех случаях, когда параметры транслируемых по этим стандартным схемам программ не превосходят ограничений (если они имеются) на эти параметры. В противном случае должны применяться стандартные схемы, построенные с помощью метода, изложенного в разд.2. При еще более жестких ограничениях следует отказаться от использования стандартных схем и перейти к построению нерегулярных функциональных схем с помощью других методов, изложенных в [8].

С материалом, изложенным в настоящей работе, можно ознакомиться также в главе 17 работы [12].

Подход, изложенный в настоящей работе, был впервые использован в 1991 году при создании системы управления дизель-генератором ДГР-2А 500*500 судна проекта 15640 на базе аппаратуры "Selma-2" [2,13].

"Человеческие" методы программирования на других языках по стандарту IEC 1131-3, отличающиеся от предлагаемых в документациях ведущих фирм мира, например [3], изложены в [8].

По нашему мнению, наиболее целесообразно программирование логических контроллеров выполнять на языке "Структурированный текст" (ST), который обязательно должен содержать конструкцию, аналогичную конструкции switch языка Си, а не на языке функциональных блоков, который рассмотрен в настоящей работе и который широко используется при программировании ПЛК [5,6].

Список литературы

1. I*nternational* Standard IEC 1131-3. Programmable Controllers. Part 3. Programming languages //International Electrotechnical Commission. 1993.
2. *Selma-2*. Описание функциональных блоков. АББ Стромберг Драйвс, 1989.
3. *SIMATIC*. Simatic S7/M7/C7. Programmable Controllers. SIEMENS. Catalog ST 70. 1996.
4. Серия программируемых контроллеров Modicon TST Quantum. GROUPE SCHNEIDER. 1998.
5. *ISaGRAF*. Standard IEC 1131-3. Computer aided software engineering workbenh for open PLCs and industrial computers. User's Guide. CJ International. 1994.
6. *Шакиров С., Биюсов Р., Якубович В*. и др. ULTRALOGIC – система подготовки программ для промышленных контроллеров //Современные технологии автоматизации. 1997. №3.
7. *Боуэн Д.П., Хинчи М.Д*. Десять заповедей формальных методов //Мир ПК. 1997. №10.
8. *Шалыто А.А*. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
9. *Шалыто А.А*. SWITCH-технология. Алгоритмизация и программирование задач логического управления //Промышленные АСУ и контроллеры. 1999. №9.
10. *Кондратьев В.Н., Шалыто А.А*. Использование функциональных схем при программной реализации автоматов //Судостроит. пром-сть. Сер. Автоматика и телемеханика. 1991. Вып.13.
11. *Хоуп Г*. Проектирование цифровых вычислительных устройств на интегральных схемах. М.: Мир, 1984.
12. *Шалыто А.А*. Логическое управление. Методы аппаратной и программной реализации. СПб.: Наука, 2000.
13. *Project 15640.* AS 21.DG1. CONTROL. АМИЕ. 95564.12М. St.Petersburg. ASS "Avrora", 1991.