Программная реализация автомата

Писецкий В.В., Никитин В.А., - СарФТИ НИЯУ МИФИ
Николаев Д.Б., Гончаров С.А. – РФЯЦ-ВНИИЭФ

В статье рассматриваются способы программного синтеза сложных управляющих автоматов. Авторы предлагают достаточно простую методику синтеза программного автомата по графу автомата.

При разработке ряда устройств и интеллектуальных датчиков часто возникает проблема создания управляющего устройства. Такое устройство должно быть простым - не сложнее объекта управления, иначе теряется смысл разработки. В настоящее время в качестве такого устройства разработчики чаще всего используют программный автомат, сделанный на базе младших семейств микроконтроллеров, что вполне оправдано по соотношению функционал-стоимость. Однако такое решение требует от разработчика умения программировать микроконтроллеры и корректно верифицировать программы.

В то же время существует и достаточно полно описана технология, позволяющая конвертировать граф автомата в эквивалентную ему программу. Это – так называемая Switch-технология разработки систем управления на базе автоматов, охватывающая процесс спецификации, проектирования, реализации, отладки, верификации, документирования и сопровождения, предложенная А. А. Шалыто в 1991 году [1].

Существовавшие до нее подходы к алгоритмизации подобных систем базировались либо на понятии «событие» либо на понятии «точка алгоритма». Понятие «событие» предполагает, что вычислительная система находится в пассивном состоянии до наступления внешнего для системы события, после чего выполняются действия, определяемые алгоритмом в качестве реакции на это событие. Понятие «точка алгоритма» подразумевает, что в данной точке алгоритма выполняется заданный алгоритмом набор действий.

В Switch-технологии первичным является понятие «состояние», интуитивно близкое по смыслу к понятию «точка алгоритма». В качестве языка описания алгоритмов в этом случае применяются графы переходов. Другими словами, базой для разработки является поведенческая модель управляющего устройства, а не его схемотехника.

Эта технология достаточно хорошо описана в литературе [2] и статьях, например - [3]. Однако большинство из описаний четко ориентировано на реализацию технологии на базе оператора Switch языка С. Применение же языка высокого уровня при разработке прикладного программного обеспечения для микроконтроллеров в ряде случаев оказывается неоправданным, особенно в случае требования обеспечения достаточно жесткого реального времени.

Более того, примеры реализации этой технологии для программ на ассемблере, приведенные в [1], относятся к алгоритмам, которые могут быть преобразованы в более-менее линейную структуру.

Так, например, автор в [1] рассматривает синтез RS – триггера, как автомата, описываемого графом, показанным на рисунке 1.

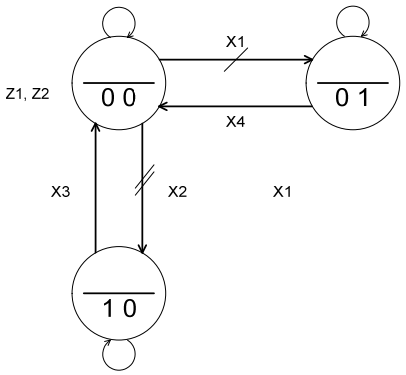

Рис. 1 Пример графа переходов (рис 14.12 из [1])

На основании этого графа автор строит 4 варианта программы на ассемблере, показанные на рисунке 2.

```
M5: CPL   z1
    MOV c, z1
    ANL c, /z2
    ANL c, x2
    JNC   M1
    SETB  z1
    CLR   z2
M1: CPL   z1
    MOV c, z1
    ANL c, /z2
    ANL c, x1
    JNC   M2
    CLR   z1
    SETB  z2
M2: MOV c, z2
    ANL c, /z1
    ANL c, x4
    JNC   M3
    CLR   z1
    CLR   z2
M3: MOV c, z1
    ANL c, /z2
    ANL c, x3
    JNC   M4
    CLR   z1
    CLR   z2
M4: SJMP  M5
```

```
M5: CPL   z1
    MOV c, z1
    ANL c, /z2
    ANL c, x2
    JNC   M1
    SETB  z1
M1: CPL   z1
    MOV c, z1
    ANL c, /z2
    ANL c, x1
    ANL c, /x2
    JNC   M2
    SETB  z2
M2: MOV c, z2
    ANL c, x4
    JNC   M3
    CLR   z2
M3: MOV c. z1
    ANL c, x3
    JNC   M4
    CLR   z1
M4: SJMP  M5
```

```
M5: CPL   z1
    MOV c, z1
    ANL c, /z2
    JNC   Ml
    MOV c, xl
    JNC   M2
    CPL   z1
    SETB  z2
M2: MOV c, x2
    JNC   Ml
    SETB  z1
    CPL   z2
M1: MOV c, z2
    ANL c, /z1
    ANL c, X4
    JNC   M3
    CLR   z1
    CLR   z2
M3: MOV c, z1
    ANL c, /z2
    ANL c, x3
    JNC   M4
    CLR   z1
    CLR   z2
M4: SJMP  M5
```

```
M5: CPL   z1
    MOV c, z1
    ANL c. /z2
    JNC   M1
    MOV c, x1
    ANL c, /x2
    JNC   M2
    SETB  z2
M2: MOV c, x2
    JNC   M1
    SETB  z1
M1: MOV C, z2
    ANL C, x4
    JNC   M3
    CLR   z2
M3: MOV c, z1
    ANL C, x3
    JNC   M4
    CLR   z1
M4 : SJMP  M5
```

Рис. 2 Программы, основанные на графе рис.1 (стр. 432-433 из [1])

Как видно из приведенного выше примера, методика дает неоднозначные варианты построения программ. Кроме того, восстановить исходный граф по любому из вариантов программы – задача хоть и решаемая, но весьма нетривиальная. К тому же использование автором нестандартных команд (например – команда «ANL c, /x2», в которой, судя по смыслу, символ «/» означает инверсию), наводит на мысль о том, что пример дан в условно-упрощенном варианте, не реализуемом на практике.

Кроме того, для разработчика управляющего автомата одной из самых важных задач является синхронизация автомата с объектом управления. Привязка любого из приведенных на рисунке 2 алгоритмов к внешней временной сетке является сама по себе достаточно сложной задачей.

Судя по всему, перед автором технологии такая задача и не стояла, поскольку (судя по тексту книги [1]) использовать Switch – технологию автор предполагал на промышленных контроллерах, где проблема синхронизации решается аппаратно – путем выделения задаче тайм-слота. Кроме того, такие контроллеры не предназначены для управления быстродействующим оборудованием и реализуют квант реального времени порядка сотни миллисекунд.

Для устранения перечисленных выше проблем авторы настоящей статьи предлагают собственную технологию построения программы по графу автомата. Предлагаемая технология позволяет достаточно быстро реализовать на языке ассемблера практически любого микроконтроллера автомат с заданным (или разработанным) графом, не предъявляя высоких требований к квалификации программиста. Кроме того, предлагаемая технология априори исключает некоторые ошибки, которые могут возникнуть при применении Switch-технологии на языке С – появление в переменной, отвечающей за номер состояния автомата, значения, выходящего за пределы, и других, см. [3].

Предлагаемая технология сводится к следующему упорядоченному набору действий:

1. Все вершины графа нумеруются. Порядок нумерации – не критичен;
2. Граф переносится в программу. Для этого создается структура, состоящая из:

- метки, связанной с номером состояния графа. Обычно это структура, состоящая из как минимум одного символа и цифры, совпадающей с номером вершины;
- тела вершины, в котором формируются выходные сигналы, соответствующие данной вершине графа;

- блока переходов, в котором проверяются условия перехода на другую вершину и, при обнаружении выполнения условия, выполняется переход. Анализ условий расставляется в соответствии с приоритетами переходов. В случае, если ни одно из условий не выполнено, осуществляется переход на начало блока.

3. Порядок размещения таких структур – произвольный.

4. На стартовом адресе размещается программа инициализации необходимых устройств, заканчивающаяся командой перехода на ту вершину, с которой предполагается старт автомата.

Реализацию предложенной технологии проще всего показать на конкретном примере. В качестве примера возьмем граф, показанный на рисунке 1 и пронумеруем его вершины. Результат показан на рисунке 3.

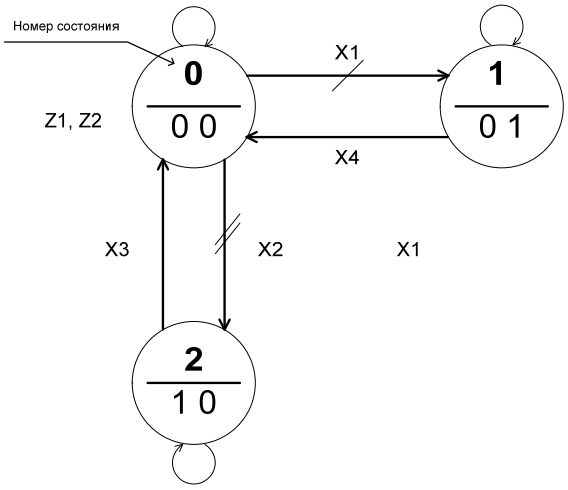

Рис. 3 Граф рис.1 с пронумерованными вершинами.

Применив к графу рис. 3 описанную выше технологию, получим следующую программу.

```
                .org    0
                jmp     start
                ; таблица векторов прерываний
                .org    100h
                ; описание входных и выходных
                сигналов
start:          ; инициализация устройств
                                …
                SJMP    s00
; Вершина 0
s00:            CLR     z1
                CLR     z2
j00:            JB      x1,             s01
                JB      x2,             s02
                SJMP    j00
;Вершина 1
s01:            CLR     z1
                SETB    z2
j01:            JB      x4,             s00
                SJMP    j01
;Вершина 2
s02:            SETB    z1
                CLR     z2
j02:            JB      x3,             s00
                SJMP    j02
; Конец программы
                .end    start
```

Начальная инициализация

Выполняемые действия

Выполнение переходов

Выполняемые действия

Выполнение переходов

Выполняемые действия

Выполнение переходов

Рис. 4 Программа, синтезированная по графу рис. 3.

По функционалу синтезированная программа ничем не уступает программам, полученным с помощью Switch-технологии (рис. 2), а по объему кода и читаемости – очевидно выигрывает.

Однако тактовая частота автомата, синтезированного по предлагаемой технологии является плавающей – время нахождения в каждом из состояний есть величина переменная,

зависящая от количества действий, выполняемых в каждом из состояний и количества анализируемых условий переходов. Для большинства применений это вполне допустимо. Однако, если техническими требованиями на разработку регламентирована конкретная тактовая частота, то проблема может быть решена с помощью встроенного в микроконтроллер таймера. При этом изменения в программе незначительны и сводятся к проверке факта срабатывания таймера к начале блока выполнения переходов. Пример такого варианта реализации показан на рисунке 5.

```
                ...
;Вершина 2
s02:    SETB    z1                      ) Выполняемые
        CLR     z2                      ) действия
j02:    JNB     TF0,    j02             ) Проверка
        CLR     TF0                     ) таймера
        JB      x3,     s00             ) Выполнение
        SJMP    j02                     ) переходов
                ...
```

Рис. 5 Фрагмент программы, синтезированной по графу рис. 3 с синхронизацией по таймеру T0.

В ряде случаев техническое задание требует синхронизовать автомат с внешним источником тактовой частоты. В этом случае проверка таймера может быть заменена на простой алгоритм обнаружения фронта внешнего сигнала – как показано на рисунке 6.

```
                ...
;Вершина 2
s02:    SETB    z1                      ) Выполняемые
        CLR     z2                      ) действия
j02:    JNB     P2.3,   j02             ) Проверка
d02:    JB      P2.3,   d02             ) синхросигнала
        JB      x3,     s00             ) Выполнение
        SJMP    j02                     ) переходов
                ...
```

Рис. 6 Фрагмент программы, синтезированной по графу рис. 3 с синхронизацией по внешнему сигналу, подключенному к 3 биту порта P2.

Вторая команда в блоке проверки синхросигнала (метка d02) необходима для исключения ложных срабатываний. Фактически, в данном примере, для перехода в следующее состояние автомату необходимо засечь положительный фронт внешнего синхросигнала, а собственно переход происходит в момент регистрации отрицательного фронта.

Таким образом, разработанная авторами технология синтеза программных автоматов не только не уступает классической Switch-технологии, но и превосходит ее по простоте применения и удобству синхронизации. Программа, синтезированная по данной методике, получается более читабельной, и простой в отладке. Кроме того, размер графа реализуемого автомата практически не зависит от квалификации программиста и определяется, по большей части, размером памяти программ используемого микроконтроллера.

Литература

1. Шалыто А. А. Программная реализация управляющих автоматов //Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып.13, с.41,42
2. Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПБ: Наука, 1998
3. Татарчевский В. Применение Switch-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Серия статей //Компоненты и технологии. №11 2006 г - №8 2007 г.