УДК 681.3

САМОСИНХРОНИЗИРУЕМЫЙ КОНЕЧНЫЙ АВТОМАТ: ОТ ПРИМЕРА К СИНТЕЗУ[1]

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

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

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

Введение

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

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

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

Самосинхронная память типа «стек»

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

1) регистровые структуры с реверсивным сдвигом информации;

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

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

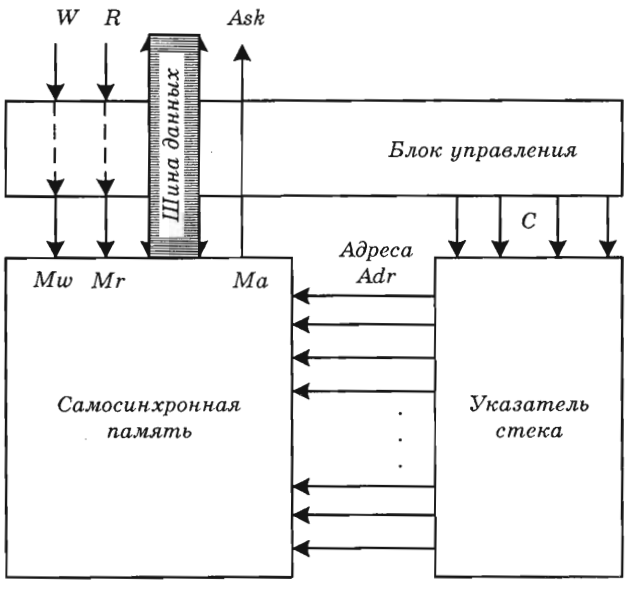
[1] STFSM – Self-Timed Finite State Machine: from Example to Synthesis, 3-d International Design Automation Workshop (Russian Workshop'93), July 19–20, Moscow, 1993.

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

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

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

W – сигнал записи в стек (PUSH);



■ *Рис. 1. Структура стековой памяти*

R – сигнал чтения из стека (POP);

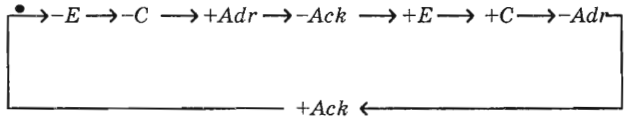
сигналы W и R поступают на управляющий автомат, определяя режим работы указателя стека, и на массив самосинхронной памяти, определяя режим работы памяти;

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

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

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

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



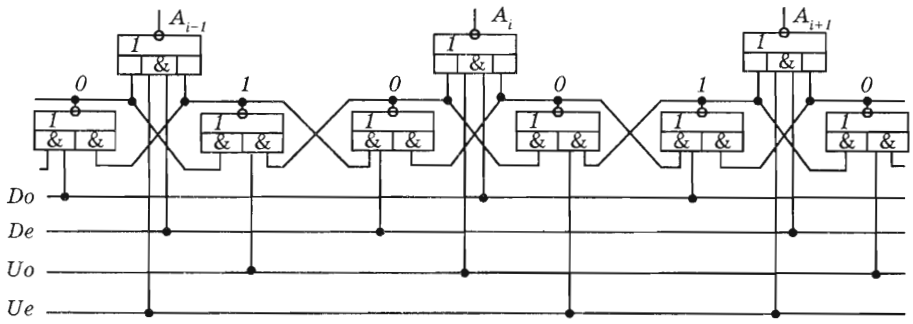
Указатель стека. При выборе или проектировании логической схемы указателя стека должно быть выполнено одно основное требование – схема должна быть максимально простой, быть может, за счет увеличения числа внешних управляющих сигналов и усложнения схем управления этими сигналами. Примером такого указателя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

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

...0101010101010010101010101010...

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

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



■ ***Рис. 2.*** *Указатель стека*

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

Содержательно поведение стека определяется поведением указателя стека следующим образом:

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

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

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

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

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

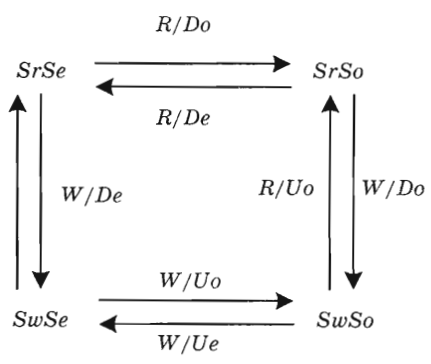
SwSo – запись на нечетной позиции;

SwSe – запись на четной позиции;

SrSo – чтение на нечетной позиции;

SrSe – чтение на четной позиции.

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



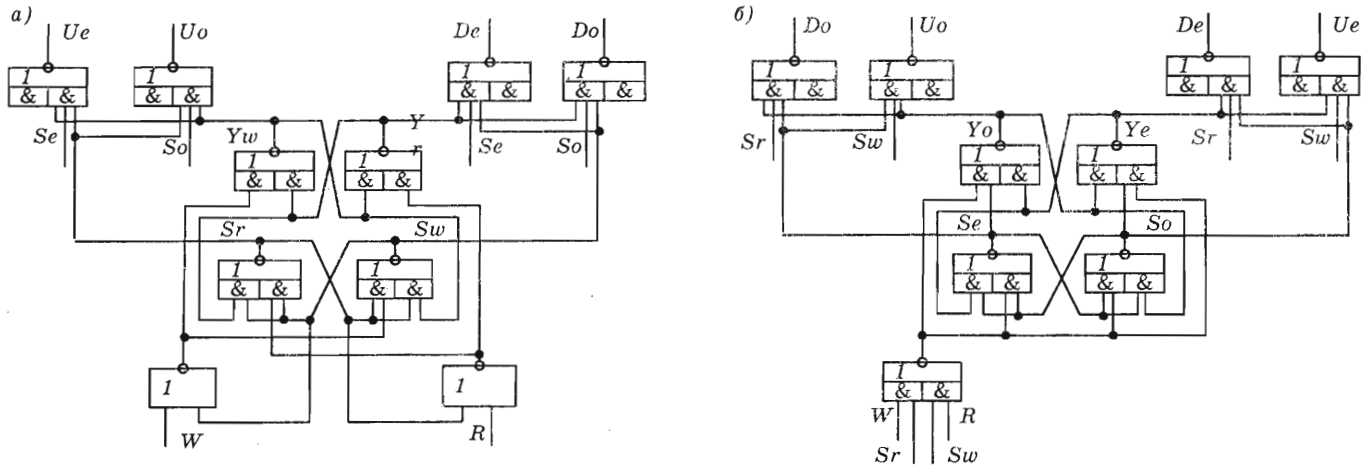
■ ***Рис. 3.*** *Граф управляющего автомата*

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

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

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

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



■ *Рис. 4. Схема автомата управления*

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

Объединяя выходные функции на переходах, получаем:

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

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

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

Обсуждение результатов

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

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

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

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

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

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

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

Литература

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