Проектирование конечных автоматов по методу ONE

Андрей СТРОГОНОВ,
д. т. н.
andreis@hotmail.ru

В ряде случаев автоматная модель устройства позволяет получить б… струю и эффективную реализацию последовательностного устройств… Обычно рассматривают два типа автоматов — автомат Мили (Mea… и Мура (Moore) [1, 2].

Конечные автоматы широко используются в различных цифровых прикладных системах и устройствах, особенно в контроллерах. Выход автомата Мура является функцией только текущего состояния, выход автомата Мили — функция как текущего состояния, так и начального внешнего воздействия.

Обычно конечный автомат состоит из трех основных частей:

- Регистр текущего состояния. Этот регистр представляет собой набор тактируемых *D*-триггеров, синхронизируемых одним синхросигналом. Этот набор используется для хранения кода текущего состояния автомата. Для автомата с n состояниями требуется $\mathrm{Log}_2(n)$ триггеров.
- Логика переходов. Конечный автомат может находиться в каждый конкретный момент времени только в одном состоянии. Каждый тактовый импульс вызывает переход автомата из одного состояния в другое. Правила перехода определяются комбинационной схемой, называемой логикой переходов. Следующее состояние определяется как функция текущего состояния и входного воздействия.
- Логика формирования выхода. Выход цифрового автомата обычно определяется как функция текущего состояния и исходной установки (в случае автомата Мили). Формирование выходного сигнала автомата определяется с помощью логики формирования выхода.

В качестве примера рассмотрим проектирование простейшего синхронного автомата, который формирует два неперекрывающихся импульса Out1 и Out2 в ответ на появление сигнала Run на входе автомата (рис. 1а). Полностью синхронный конечный автомат использует регистры для фиксации всех выходных сигналов управления и состояний, а также для асинхронных входных сигналов. Следует заметить, что синхронные конечные автоматы по быстродействию уступают асинхронным.

Метка, расположенная в каждом круге выше линии, — это имя состояния, а метки ниже линий — это выходные сигналы, которые выдаются, когда данное состояние активно. «Дуги», которые возвращаются в то же самое состояние, — это переходы, которые работают по умолчанию. Эти дуги будут иметь истинные значения только в случае, когда не будет истинных значений других условий переходов. Каждое условие перехода из состояния в состояние имеет соответствующее логическое условие, которое должно выполняться, чтобы конечный автомат мог перейти в следующее состояние.

Автомат принимает четыре состояния: Idle, Delay, Next, Done (рис. 1а). Воспользуемся методом двоичного кодирования состояний, который обеспечивает высокую степень кодирования последовательности состояний. Для кодирования состояний потребуется два триггера. Запишем булевы логические уравнения:

$$Idle = \overline{S0} * \overline{S1};$$
$$Delay = \overline{S1} * S0;$$
$$Next = S1 * S0;$$
$$Done = S1 * \overline{S0};$$
$$S0 := (Idle, Run) + Delay;$$
$$S1 := Delay + Next + (Done, \overline{Run});$$
$$Out1 := Idle, Run;$$
$$Out2 := Next.$$

Символ = обозначает комбинацио… схему, ответственную за переход по сос… ниям, а символ := обозначает триггер… выход, необходимый для хранения ко… кущего состояния автомата и выходных… налов. Схема, построенная по булевым… нениям в САПР ПЛИС Quartus II комп… Altera, показана на рис. 2. На рис. 3 по… ны временные диаграммы работы авто… Уравнение для выходного сигнала … представляет собой функцию как состо… так и входного сигнала Run. Конечны… томат с таким видом стробирования в… дов называется автоматом Мили. Урав… для выходного сигнала Out2 записыв… как функция только состояния автомат… соответствует структуре автомата М… Для автоматов Мили удобно предста… диаграммы в другом виде (рис. 1б).

Метод one hot encoding (ONE — кодир… ние с одним активным, или горячим, с… янием; иначе — унитарное кодирование… лучил такое название потому, что в ка… конкретный момент времени активным… может быть только один триггер состо… Применение метода ONE для ПЛИС … хитектуре ППВМ (программируемые … зователем вентильные матрицы, зарубе… аббревиатура — FPGA) было предло… компанией High-Gate Design [3].

Построение конечного автомата с ис… зованием метода ONE осуществляется … дующей методике — вначале для отоб… ния каждого состояния автомата выде…

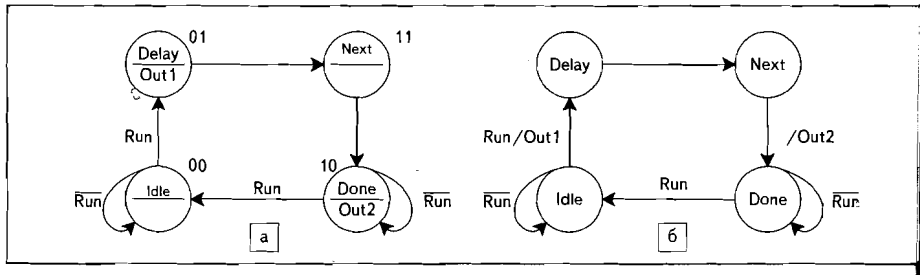

Рис. 1. Граф-автомат проектируемого устройства

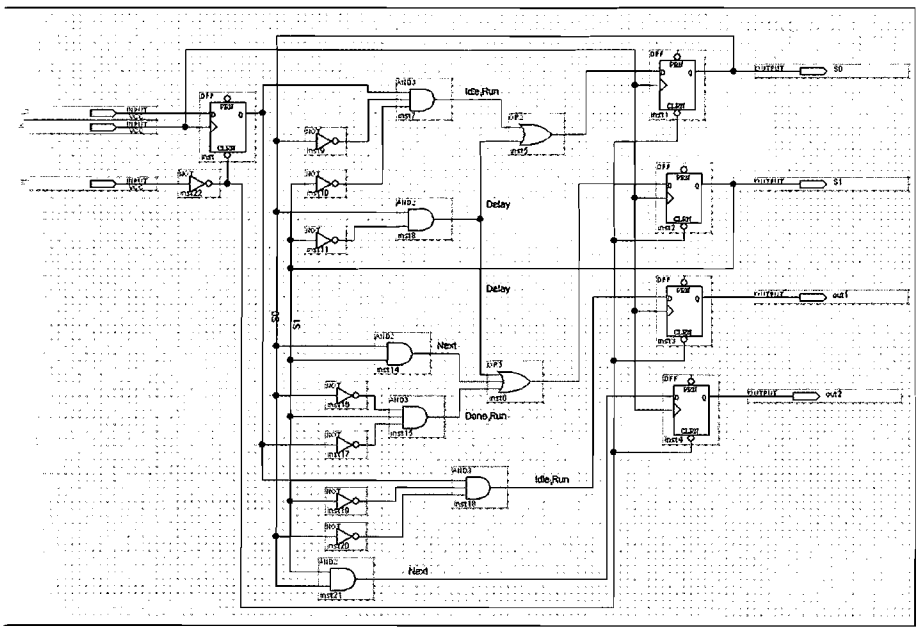

Рис. 2. Схема конечного автомата, построенного по логическим уравнениям

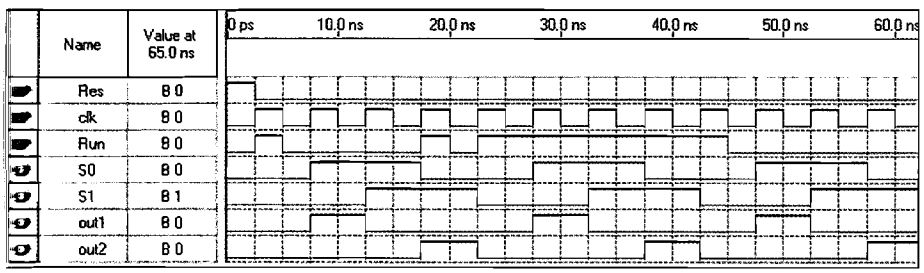

Рис. 3. Временные диаграммы работы конечного автомата

дивидуальный триггер, а затем организуется схема, позволяющая в каждый конкретный момент времени только одному состоянию быть активным [3, 4].

Рассмотрим конечный автомат Мура, просматривающий семь различных состояний [3, 4]. Построим граф-автомат проектируемого устройства (рис. 4). Автомат переходит из состояния в состояние по переднему фронту синхроимпульса, который отмечен «крестом». Иерархическая блок-схема автомата, состоящая из 7 блоков S1–S7 и логики формирования выхода, в САПР ПЛИС Quartus II компании Altera показана на рис. 5.

В примере имеется семь состояний (stage1–7), каждый блок ответственен за формирование своего состояния, например блок S1 отвечает за формирование состояния 1. Все логические входы помечаются как переменные от *A* до *E*. Выходы конечного автомата носят названия Multi, Contig и Single. В данном примере состояние 1, в котором должен находиться конечный автомат при включении питания, имеет структуру триггера с двумя инверторами (схема S1, рис. 6). Для того чтобы конечный автомат при включении питания всегда принимал известное начальное состояние, выход триггера состояния 1 инвертируется, а чтобы обеспечить логическую непротиворечивость, входной информационный сигнал этого триггера также инвертируется. Таким образом, состояние 1 в начальный момент времени принимает значение логической единицы. Для всех других состояний 2–7 используется *D*-триггер с асинхронным сбросом, тактируемый фронтом синхросигнала.

Автомат спроектирован так, что активный низкий уровень сигнала RSTG (глобальный сброс состояний всего автомата, кроме состояния 1) в начальный момент сбрасывает состояния 2–7 (S2–S7) в ноль, а состояние 1 будет находиться в единице. Далее сигнал RSTG должен всегда оставаться логической 1. В случае, если конечный автомат все же окажется в недопустимом состоянии, например, в состоянии 3, то с приходом следующего переднего фронта синхроимпульса будет установлено состояние 4. Состояние 4 сбросит состояние 3. Состояние 4 может сбросить и состояние 2. Состояние 5 сбрасывает состояние 4, состояние 6 сбрасывает состояние 5, состояние 7 сбрасывает состояние 6, а состояние 1 сбросит состояние 7. Таким образом, для правильной работы конечного автомата достаточно его один раз сбросить с помощью сигнала RSTG, а далее автомат, шагая по состояниям, способен сам их сбрасывать.

После того как установлены начальные состояния, необходимо построить логику перехода в следующее состояние. Для определения индивидуального состояния воспользуемся алгоритмом, предложенным в работе [3]. Вначале подсчитывается число условий переходов, ведущих к данному состоянию, и добавляется еще один переход, если условие по умолчанию должно оставлять конечный автомат в том же самом состоянии. Далее строится логический вентиль ИЛИ с числом входов, равным числу условий переходов, определенных ранее.

Далее, для каждого входа вентиля ИЛИ строится логический вентиль И, входами которого служат предыдущие состояния и его логика условия. Если по умолчанию конечный автомат должен оставаться в том же самом состоянии, строится логический вентиль И, входами которого служат данное состояние и обратная величина всех возможных условий переходов, исходящих из данного состояния.

Чтобы определить число условий переходов для состояния 1, рассмотрим граф-автомат. Из рис. 6 видно, что состояние 1 имеет один переход от состояния 7, когда переменная E истинна. Другой переход — это условие

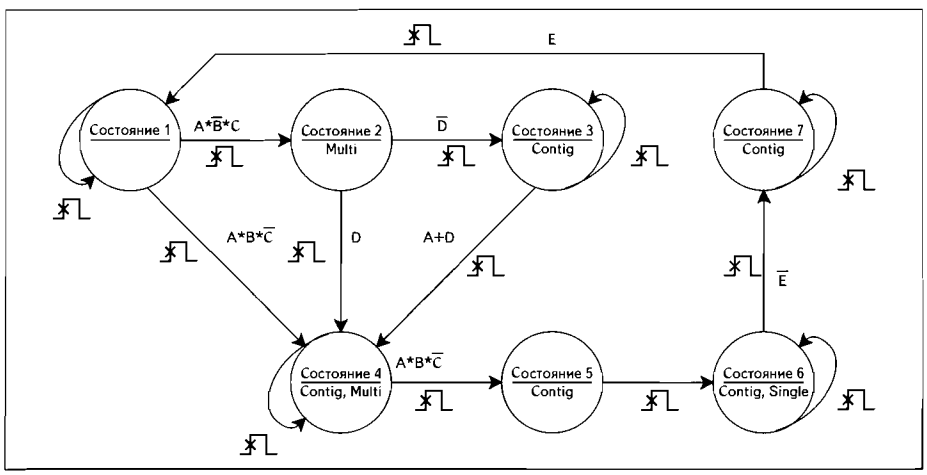

Рис. 4. Граф-автомат проектируемого устройства (КА Мура)

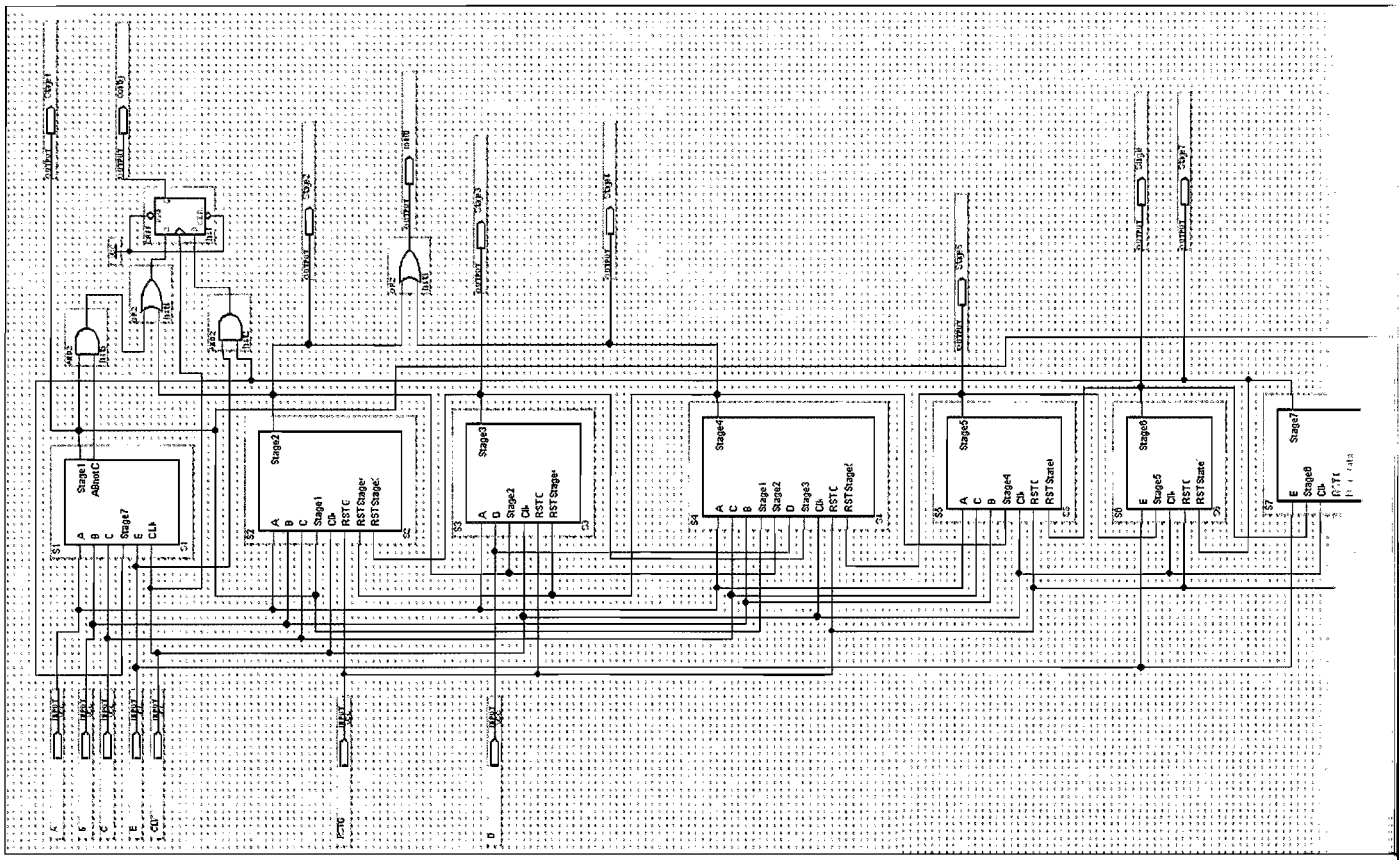

Рис. 5. Иерархическая блок-схема автомата с кодированием по методу ОНЕ в САПР ПЛИС Quartus II

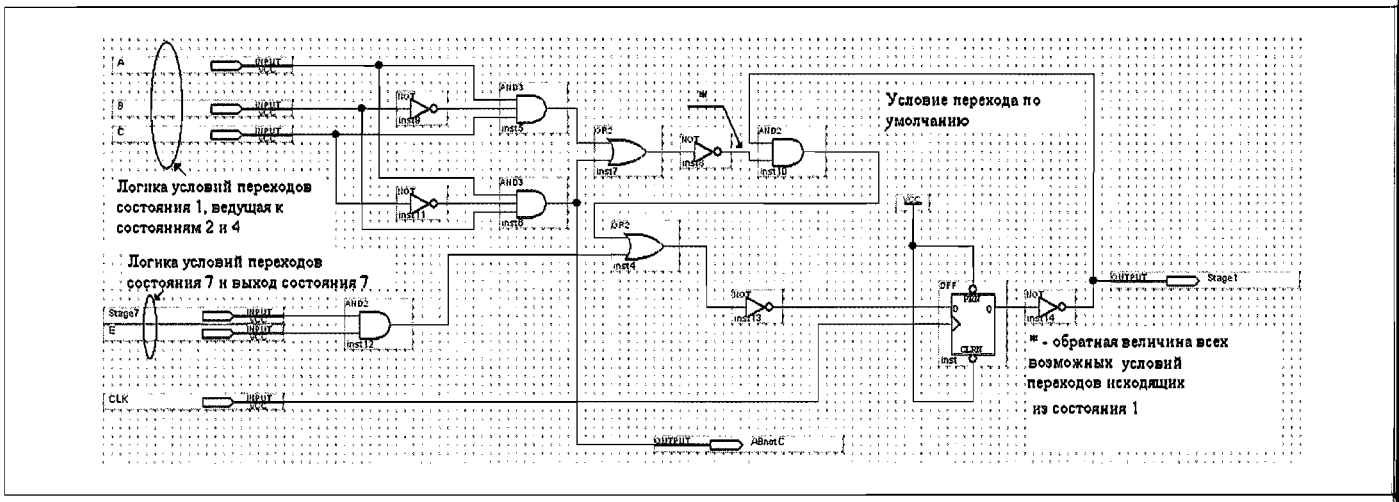

Рис. 6. Схема для состояния 1

по умолчанию, ведущее в состояние 1. Таким образом, состояние 1 имеет два условия переходов. После этого можно построить двухвходовой логический вентиль 2ИЛИ с одним входом для условия перехода от состояния 7, а другим — для перехода по умолчанию, чтобы оставаться в состоянии 1 (рис. 6).

Следующий шаг — это построение логики переходов для данного вентиля 2ИЛИ. Каждый вход вентиля 2ИЛИ есть логическая функция И предыдущего состояния и логики переходов состояния 1. Например, состояние 7 поступает на вход состояния 1, когда E имеет истинное значение. Это обеспечивается при помощи логического вентиля 2И (рис. 6). Второй вход вентиля ИЛИ — переход по умолчанию, когда конечный автомат должен оставаться в состоянии 1. Если текущее состояние есть состояние 1 и нет условий переходов, выходящих из состояния 1, которые истинны, то конечный автомат должен оставаться в состоянии 1. Состояние 1 на диаграмме состояний имеет два исходящих условия переходов (рис. 6).

Первый переход является действительным, когда истинно условие $(A\overline{B}C)$, и ведет в состояние 2. Второй переход, ведущий в с... яние 4, является действительным при и... ном значении условия $(AB\overline{C})$. Логи... умолчанию — это функция И для со... ния 1 обратной величины всех условий ... ходов, исходящих из состояния 1. Эта ... ческая функция реализуется с использо... ем вентиля 2И с инвертором на одно... входов и логических элементов, форми... щих сигнал для инвертирующего входа ... тиля 2И (рис. 6). Комбинационная ло... обеспечивает декодирование с учетом ... ных сигналов и сигнала обратной связи

Состояние 4 не является начальным состоянием, поэтому для его представления используется *D*-триггер без инверторов, с входом асинхронного сброса RSTG. Триггер может быть сброшен и выходом состояния 5 (сигнал RSTState5). Имеется три входящих условия перехода и условие по умолчанию, чтобы конечный автомат мог оставаться в состоянии 4. Поэтому на входе триггера используется вентиль 4ИЛИ (схема S4, рис. 7).

Первое условие перехода исходит из состояния 3. В соответствии с изложенными выше правилами необходимо построить функцию И для состояния 3 и логику условия, которая имеет вид $A+D$ (рис. 7).

Следующее условие перехода исходит из состояния 2, оно требует логической функции И для состояния 2 и переменной *D*. Последнее условие перехода для состояния 4 — от состояния 1. Выход состояния 1 должен пройти через схему 2И с логикой его условия перехода — логическим произведением $AB\overline{C}$ (рис. 7).

Далее нужно построить логику, обеспечивающую сохранение состояния 4, когда ни одно из условий переходов, исходящих из состояния 4, не имеет истинного значения. Переход, исходящий из состояния 4, является действительным, когда логическое произведение $AB\overline{C}$ истинно. Следовательно, необходимо пропустить состояние 4 через вентиль И с обратной величиной произведения $AB\overline{C}$. Это необходимо для поддержания триггера в высоком уровне, пока не произойдет действительный переход в следующее состояние. В логике перехода по умолчанию используется вентиль 2И и выход вентиля 3И с инвертором на входе *C*.

Состояние 2 имеет только одно условие перехода, которое приходит от состояния 1, когда произведение $A\overline{B}C$ истинно. Конечный автомат будет немедленно переходить по одному из двух переходов из состояния 2 в зависимости от значения сигнала *D*. Состояние 3, подобно состояниям 1 и 4, имеет переход по умолчанию, и для управления входом D-триггера используется комбинация сигналов *A* и *D*, состояния 2 и состояния 3. Состояние 5 управляет состоянием 6 без всяких условий. Конечный автомат ждет в состоянии 6, пока переменная *E* не переключится на низкий уровень, прежде чем перейти в состояние 7. В состоянии 7 конечный автомат ждет переключения переменной *E* в истинное значение, после чего переходит в состояние 1.

После описания всей логики переходов по состояниям описывается выходная логика. В примере используются три выходных сигнала — Multi, Contig и Single, — каждый из которых относится к одной из трех основных категорий выходных сигналов:

Выходные сигналы, формируемые в одном состоянии. Примером может служить выходной сигнал Single, формируемый только в состоянии 6, то есть выходным сигналом является выход триггера.

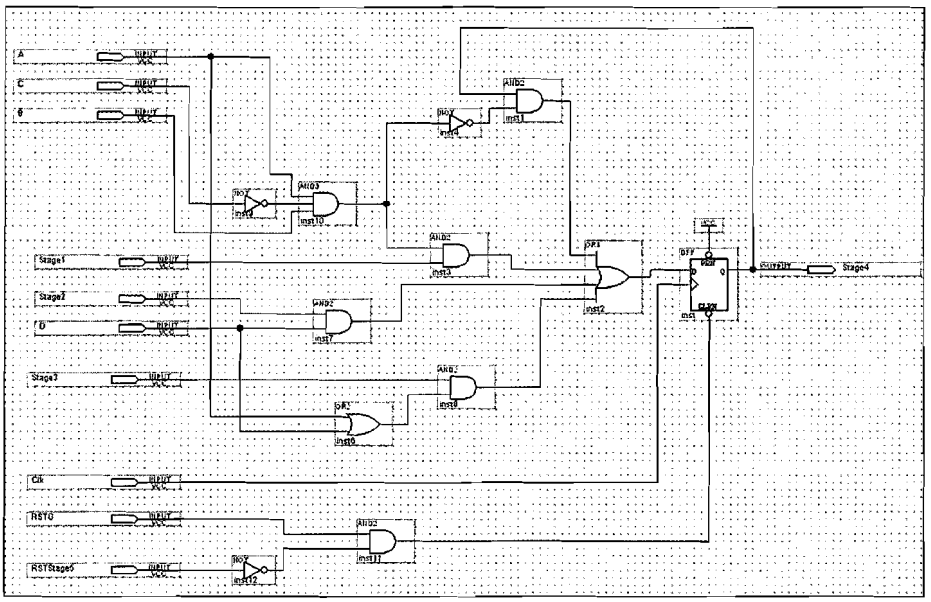

Рис. 7. Схема S4 для состояния 4

2. Выходные сигналы, формируемые во многих смежных состояниях. Например, выходной сигнал Contig, который формируется в состояниях 3–7, хотя есть ветвь для состояния 2.
3. Выходные сигналы, формируемые по многим несмежным состояниям. Здесь обычно оптимальное решение — это простое декодирование активных состояний. Например, сигнал Multi, который формируется для состояний 2 и 4.

Для формирования логики выходного сигнала Multi используется декодирование состояний 2 и 4 при помощи вентиля 2ИЛИ. Каждый раз, когда конечный автомат окажется в одном из этих состояний, будет сформирован активный сигнал Multi. Для декодирования выходных сигналов для смежных состояний используется синхронный RS-триггер. RS-триггер устанавливается при входе в смежное состояние и сбрасывается при выходе (рис. 5). Временная диаграмма проектируемого автомата представлена на рис. 8.

Для размещения автомата выберем ПЛИС по архитектуре ППВМ APEX20K (EP20K30ETC144-1). Архитектура ПЛИС семейства APEX20K сочетает в себе достоинства ППВМ ПЛИС с их таблицами перекодировок (LUT). После компиляции проекта оказалось задействовано 20 логических элементов, 8 триггеров. Моделирование проводилось без учета реальных задержек распространения сигналов в ПЛИС. С учетом реальных задержек период тактового сигнала CLK для стабильной работы автомата должен быть не менее 15 нс. Уменьшить число триггеров на один позволяет декодирование состояний 3–7 при помощи 5-входового вентиля ИЛИ. Каждый раз, когда конечный автомат окажется в одном из этих состояний, будет сформирован сигнал Conting. В этом случае и сокращается число логических элементов. Максимальная тактовая частота в обоих схемных решениях составляет f_{MAX} = 290,02 МГц.

Опишем функционирование данного автомата на языке описания аппаратных средств VHDL. Описание проектируемого конечного автомата с использованием двухпроцессного шаблона и перечисляемого типа данных (Enumerated type) на языке VHDL приведено

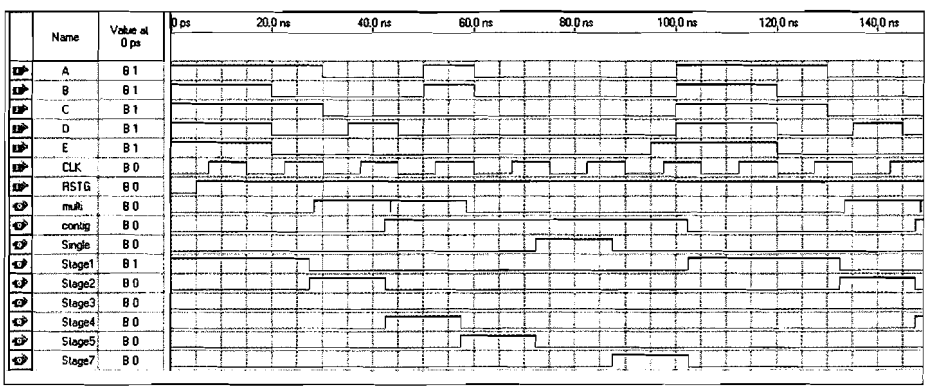

Рис. 8. Результаты моделирования работы конечного автомата с принудительным сбросом состояний. Показаны переходы по состояниям 1–7

далее. Перечисляемый тип — это такой тип данных, при котором количество всех возможных состояний конечно. Такой тип наиболее часто используется для обозначений состояний конечных автоматов. Любой перечисляемый тип имеет внутреннюю нумерацию: первый элемент всегда имеет номер 0, второй — 1 и т. д.

Для этой модели триггеры синтезируются только для сигнала state, что позволяет избежать лишних триггеров в схеме. Для обеспечения стабильной и безотказной работы используется сброс автомата в начальное состояние (активный высокий уровень сигнала TRST). Таким образом, всегда обеспечивается инициализация автомата в начальное состояние.

В данном примере стиль кодирования (например, метод двоичного кодирования или кодирование по методу OHE) не определен в коде языка VHDL. Xilinx рекомендует использовать кодирование цифровых автоматов с использованием перечисляемого типа, так как в этом случае САПР предоставляется возможность использовать модуль логического синтеза и в зависимости от архитектуры ПЛИС самостоятельно выбирать метод кодирования [5]. В САПР Quartus II предлагается использовать три метода кодирования: Auto, OHE, Minimal Bits. В САПР Xilinx используется семь методов кодирования: Auto, OHE, Gray, Compact, Johnson, Sequential, User [5].

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY avtOHE IS
   PORT(
      A,B,C,D,E,TRST,TCK   : IN     STD_LOGIC;
      Multi, Contig,Single : OUT    STD_LOGIC;
      STATE1_7             : OUT    STD_LOGIC_VECTOR
                                    (3 DOWNTO 0));
END avtOHE;
ARCHITECTURE a OF avtOHE IS
TYPE state_values IS (Stage1, Stage2, Stage3, Stage4, Stage5, Stage6,
Stage7);
signal state,next_state: state_values;

BEGIN
-- регистровый блок
statereg: process(TCK,TRST)
begin
    if (TRST = '1') then state<=Stage1;
        elsif (TCK'event and TCK='1') then
            state<=next_state;
    end if;
end process statereg;
-- комбинаторный блок (логика переходов)
process(state, A,B,C,D,E)
begin
case state is
when Stage1=>
STATE1_7 <= «0001»;
IF (A='1' and B='0' and C='1') THEN next_state<=Stage2;
ELSIF (A='1' and B='1' and C='0') THEN next_state<=Stage4;
ELSE next_state<=Stage1; END IF;
when Stage2=>
STATE1_7 <= «0010»;
IF (D='0') THEN next_state<=Stage3;
ELSIF (D='1') THEN next_state<=Stage4;
END IF;
when Stage3=>
STATE1_7 <= «0011»;
IF (A='1' or D='1') THEN next_state<=Stage4;
ELSE next_state<=Stage3; END IF;
when Stage4=>
STATE1_7 <= «0100»;
IF (A='1' and B='1' and C='0') THEN next_state<=Stage5;
ELSE next_state<=Stage4;END IF;
when Stage5=>
STATE1_7 <= «0101»;
next_state<=Stage6;
when Stage6=>
STATE1_7 <= «0110»;
IF (E='0') THEN next_state<=Stage7;
ELSE next_state<=Stage6; END IF;
when Stage7=>
STATE1_7 <= «0111»;
IF (E='1')THEN next_state<=Stage1;
ELSE next_state<=Stage7; END IF;
end case;
end process;
-- логика формирования выхода
process (state)
begin
Multi <= '0';
Contig <= '0';
Single <= '0';
case state is
when Stage1=>
Multi <= '0';
Contig <= '0';
Single <= '0';
when Stage2=>
Multi <= '1';
Contig <= '0';
Single <= '0';
when Stage3=>
Multi <= '0';
Contig <= '1';
Single <= '0';
when Stage4=>
Multi <= '1';
Contig <= '1';
Single <= '0';
when Stage5=>
Multi <= '0';
Contig <= '1';
Single <= '0';
when Stage6=>
Multi <= '0';
Contig <= '1';
Single <= '1';
when Stage7=>
Multi <= '0';
Contig <= '1';
Single <= '0';
end case;
end process;
    END a;
```

Для ПЛИС по архитектуре ППВМ САПР Quartus II автоматически предлагается использовать метод кодирования OHE. В меню Project/Option&Parameter Settings также можно выбрать установку Auto, которая позволяет средствам синтеза автоматически выбрать для каждого конечного автомата наилучший алгоритм кодирования. В случае выбора установки User-Encoded средства синтеза будет использоваться алгоритм кодирования, представленный в файле исходного описания.

Выберем в меню Project/Option&Parameter Settings метод кодирования OHE с помощью установки State Machine Processing. Реализуем комбинационную логику на таблицах перекодировок LUT, меню Option&Parameter Settings, установка технологии маппирования Technology Mapper — LUT. Проект после синтеза автоматически был размещен в ПЛИС EP20K30ETC144-1. Оказалось задействовано 16 логических элементов, 7 триггеров. Максимальная тактовая частота составляет f_{MAX} = 290,02 МГц. Таким образом, число триггеров уменьшилось на один.

На рис. 9 показана тестовая схема конечного автомата, а на рис. 10 — временные диаграммы его работы. Показаны переходы по состояниям 1, 4, 5, 6 и 7. По первому такту синхроимпульса и по условию $A\bar{B}C$ автомат переходит в состояние 4. В этом состоянии формируются выходные сигналы Contig и Multi. По второму такту синхроимпульса и по условию $A\bar{B}\bar{C}$ автомат переходит в состояние 5, формируя на выходе сигнал Contig. По третьему такту синхроимпульса автомат без всяких условий переходит в состояние 6 с формированием выходных сигналов Contig и Single.

Выберем в меню Project/Option&Parameter Settings метод двоичного кодирования Minimal Bits с помощью установки State Machine Processing. При этом в процессе синтеза конечного автомата будет минимизиро-

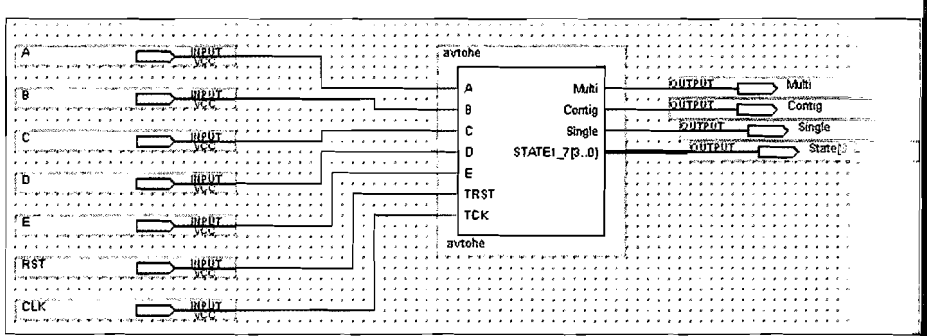

Рис. 9. Блок-схема конечного автомата

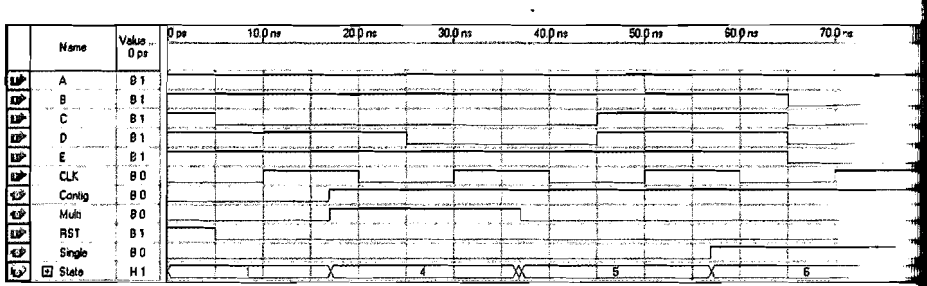

Рис. 10. Временная диаграмма конечного автомата, описанного на VHDL. Показаны переходы по состояниям 1, 4, 5, 6 и 7

Число используемых триггеров. Реализуем комбинационную логику на таблицах перекодировок LUT, меню Option & Parameter Settings, установка технологии маппирования Technology Mapper — LUT.

Проект автоматически был размещен в ПЛИС EP20K30ETC144-1. Задействовано 18 логических элементов, 3 триггера. Максимальная тактовая частота составляет f_{max} = 262,81 МГц. Таким образом, число триггеров уменьшилось на четыре за счет иного способа кодирования, но значительно увеличилось число логических элементов и несколько снизилась тактовая частота. Сравнительные результаты представлены в таблице.

Таблица. Сравнительные результаты методов ONE и двоичного кодирования

Метод	Число логических элементов	Число триггеров	Максимальная тактовая частота, f_{max}, МГц
Ручной способ кодирования по методу ONE с RS-триггером	20	8	290,02
ONE	16	7	290,02
Двоичное кодирование	18	3	262,81

Проекты с использованием цифровых автоматов на языках описания аппаратных средств, в том числе и низкоуровневых, типа VHDL, часто допускают значения битов состояний, которые не присваиваются правильным состояниям. Эти значения с неприсвоенными битами состояний называются неправильными (избыточными) состояниями. Проект, который переходит в неправильное состояние, например, в результате нарушений временных требований к установке или задержке, может реализовать ошибочные выходы. Пользователь может сделать восстановление цифрового автомата после неправильного состояния путем принудительного преобразования неправильного состояния к известному состоянию, допустимому в рамках оператора CASE [6].

Для восстановления после неправильных состояний следует поименовать их все для данного автомата. Предложение WHEN OTHERS в операторе CASE, которое принудительно преобразует состояния, применяется только к состояниям, которые были объявлены, а не упомянуты в предложении WHEN. Данный метод работает, только если все неправильные состояния определены в объявлении цифрового автомата. Для n-битового цифрового автомата существует 2^n возможных состояний. Поэтому нужно добавить воображаемые имена состояний, чтобы получилось нужное число состояний. Ниже приведен фрагмент файла, в котором реализован цифровой автомат, который может восстанавливаться из неправильных состояний [6]:

```
SUBDESIGN recover
    (clk,go : INPUT; ok : OUTPUT;)
VARIABLE sequence : MACHINE OF BITS (q[2..0]) WITH STATES
( idle, one, two, three, four, illegal1, illegal2, illegal3 );
BEGIN
    sequence.clk = clk;
CASE sequence IS
    WHEN idle => IF go THEN sequence = one; END IF;
    WHEN one => sequence = two;
    WHEN two => sequence = three;
    WHEN three => sequence = four;
    WHEN OTHERS => sequence = idle;
END CASE;
    ok = (sequence = = four);
    END;
```

В данном примере цифровой автомат имеет три бита или 8 состояний. В объявлении заданы только 5 состояний. Следовательно, в объявление нужно добавить еще три воображаемых состояния illegal1, illegal2, illegal3. Однако если число состояний значительно, то их доопределение может приводить к сокращению быстродействия автомата.

Выводы

Таким образом, метод ONE применительно к ПЛИС с архитектурой ППВМ дает возможность строить конечные автоматы, которые в общем случае требуют меньших ресурсов и отличаются более высокими скоростными показателями, чем аналогичные конечные автоматы с двоичным кодированием состояний. Если число состояний не более 8, то двоичное кодирование в этом случае может быть более эффективным.

Повышенное быстродействие по методу ONE обеспечивается меньшим числом уровней логики между рабочими фронтами синхросигналов, чем в случае двоичного кодирования. Логические схемы при этом упрощаются, поскольку метод ONE практически не требует логики декодирования состояний. Получающийся в результате построения конечного автомата набор триггеров похож на структуру типа сдвигового регистра.

Быстродействие конечного автомата типа ONE остается постоянным с увеличением числа состояний. И напротив, быстродействие конечного автомата с высокой степенью кодирования состояний снижается с увеличением количества состояний, поскольку в этом случае для декодирования требуется большее число уровней логики с большим числом линий.

При проектировании цифровых автоматов на языках описания аппаратных средств возможно появление недопустимых состояний. В этом случае необходимо доопределять состояния автомата, которые, в свою очередь, могут снижать его быстродействие. □

Литература

1. Actel Digital Library. Q3 2001. Designing State Machines for FPGAs. September 1997. 97s05d18.pdf.
2. Стешенко В. Примеры проектирования цифровых устройств с использованием языков описания аппаратуры // Схемотехника. 2001. № 7–9.
3. Knapp S. K. Accelerate FPGA macros with one-hot approach // ED. 1990. № 17.
4. **http://toolbox.xilinx.com/docsan/xilinx4/data/docs/sim/vtex9.html**
5. **http://toolbox.xilinx.com/docsan/xilinx4**. Xilinx Synthesis Technology (XST). User Guide. xst.pdf.
6. Стешенко В. Б. ПЛИС фирмы «Altera»: элементная база, система проектирования и языки описания аппаратуры. М.: Издательский дом «Додэка-XXI», 2002.