|  |  |
| --- | --- |
| ДИСЦИЛИНА | **Схемотехника устройств компьютерных систем Часть 2** |
|  |  |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **вычислительной техники** |
|  |  |
| ВИД УЧЕБНОГО | **Лекция** |
| МАТЕРИАЛА |  |
| ПРЕПОДАВАТЕЛЬ | **Тарасов И.Е.** |
|  |  |
| СЕМЕСТР | 3 |
|  |  |

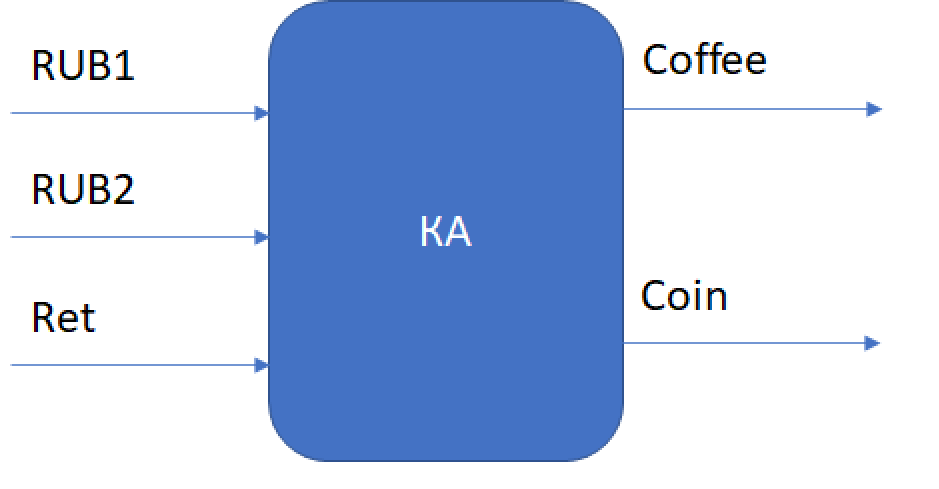
2

9. Конечные автоматы

9.1. Понятие конечного автомата

Конечный автомат (КА), также Finite State Machine (FSM) – цифровая схема, которая может находиться в конечном числе разных состояний. Каждое из состояний однозначно определяется комбинацией внутренних сигналов. Переход между состояниями происходит на основе внешних сигналов и текущего состояния конечного автомата. Именно с помощью конечного автомата можно реализовать цифровую схему, которая будет вести себя подобно компьютерной программе – последовательно выполняя запрограммированные действия. Для этого необходимо распределить требуемые действия по разным состояниям КА и задать правила перехода между этими состояниями в требуемой последовательности.

Выходные сигналы КА могут быть одинаковыми при нахождении в нескольких состояниях, однако обратное не гарантируется. С помощью конечных автоматов можно описывать работу устройств самого разного типа. Например, на рис. 9.1 показан модуль конечного автомата для управления продажей кофе. Модуль выбран в качестве примера и не соответствует реальному устройству. Он необходим для иллюстрации процесса разработки КА.



*Рисунок 9.1 Модуль конечного автомата по управлению продажей кофе*

На этом рисунке видно, что у кофейного аппарата есть датчики опускания монет Rub1 и Rub2, реагирующие на попадание соответствующих монет в монетоприемник. Также есть кнопка Ret, управляющая возвратом сдачи. У автомата есть возможность сварить кофе и выдать сдачу монетой в 1 рубль. Примем, что для этого достаточно установить в 1 соответствующие выходные сигналы.

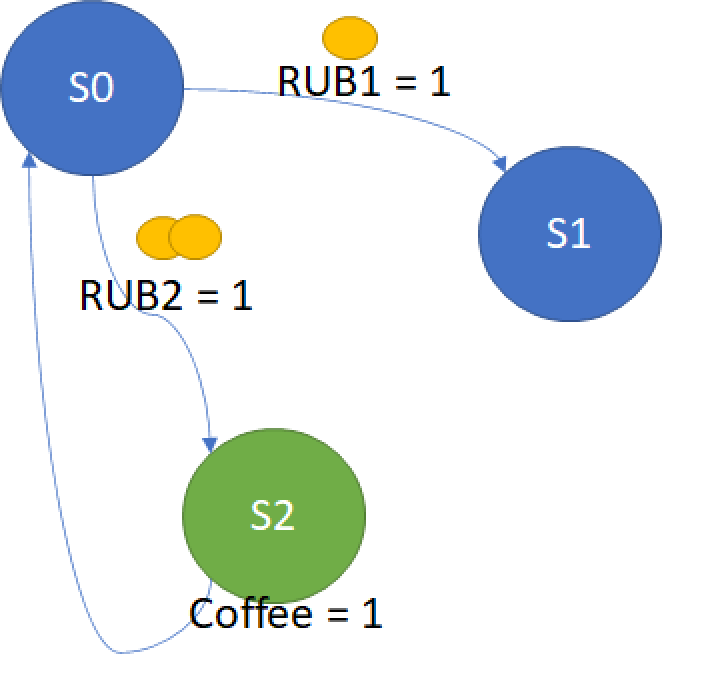
На основе данного описания можно показать процесс разработки конечного автомата. Описание КА можно представить в виде графа, представляющего собой «островки» - состояния конечного автомата, соединенные стрелками. Над стрелками записывается условие, по которому происходит переход от одного состояния в другое. Автомат является синхронным устройством, поэтому его переходы привязаны к тактовому сигналу. За одно действие автомат может перейти в какое-то одно состояние, но не пройти по длинной цепочке, даже если выполняются все условие. Если автомат попадает в какое-то состояние, ему однозначно соответствует набор выходных сигналов.

Работа автомата, выбранного в качестве примера, начинается с состояния S0. В этом состоянии у него нет активных выходов (т.е. он не готовит кофе и не выдает монету в виде сдачи). Если нет внешних сигналов, КА может находиться в состоянии S0 сколь угодно долго.

В этом состоянии можно нажимать кнопку сдачи, но в этом случае ничего не должно происходить. Могут произойти два интересующих нас события:

1. Опущена монета 1 рубль. Этого пока недостаточно для выдачи кофе, но внутреннее состояние автомата изменилось (хотя внешние сигналы остались теми же – нет выдачи кофе, нет сдачи). Поэтому необходимо завести новое состояние S1, которое будет соответствовать ситуации «уже есть 1 рубль».

2. Опущена монета 2 рубля. Этого достаточно для приготовления кофе. Автомат переходит в состояние S2, в котором есть выходной сигнал Coffee = 1. Из этого состояния автомат безусловно переходит в S0 и становится готов к приему оплаты за следующую порцию кофе. Детали приготовления кофе в данном случае неважны. Понятно, что это длительный процесс, из которого нельзя мгновенно перейти к S0, однако для упрощения на схеме нет ожидания завершения.

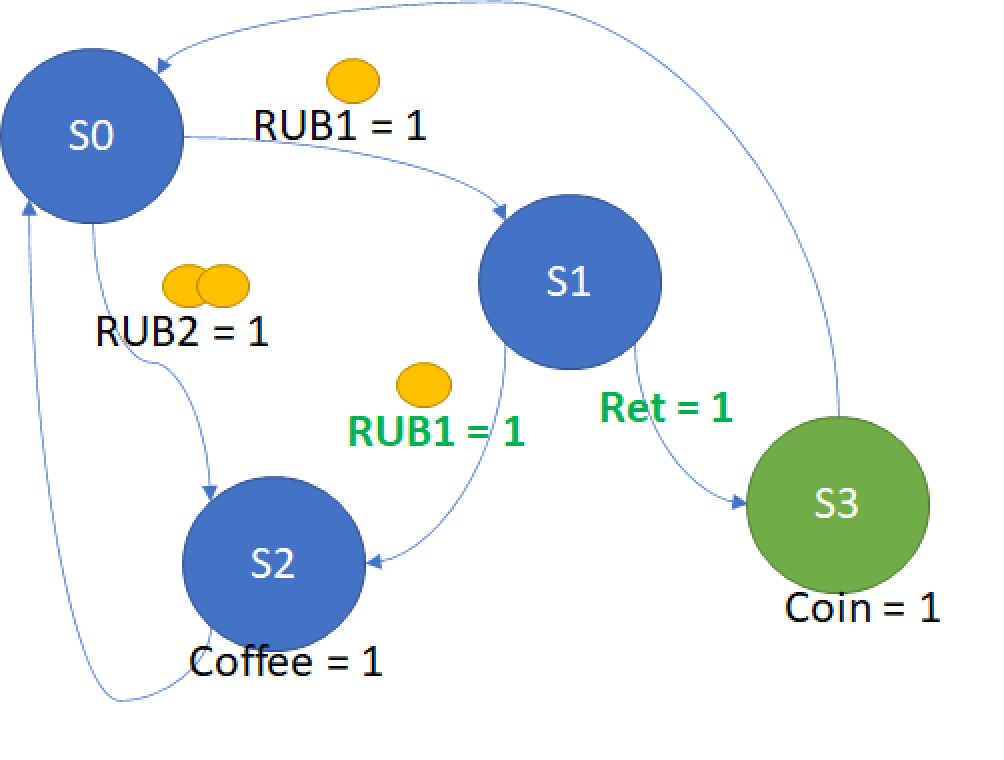


*Рисунок 9.2 Граф описания конечного автомата по управлению продажей кофе*

Поскольку цепочка S0 – S2 – S0 в целом понятна, осталось завершить описание состояния S1. Возможны два сценария:

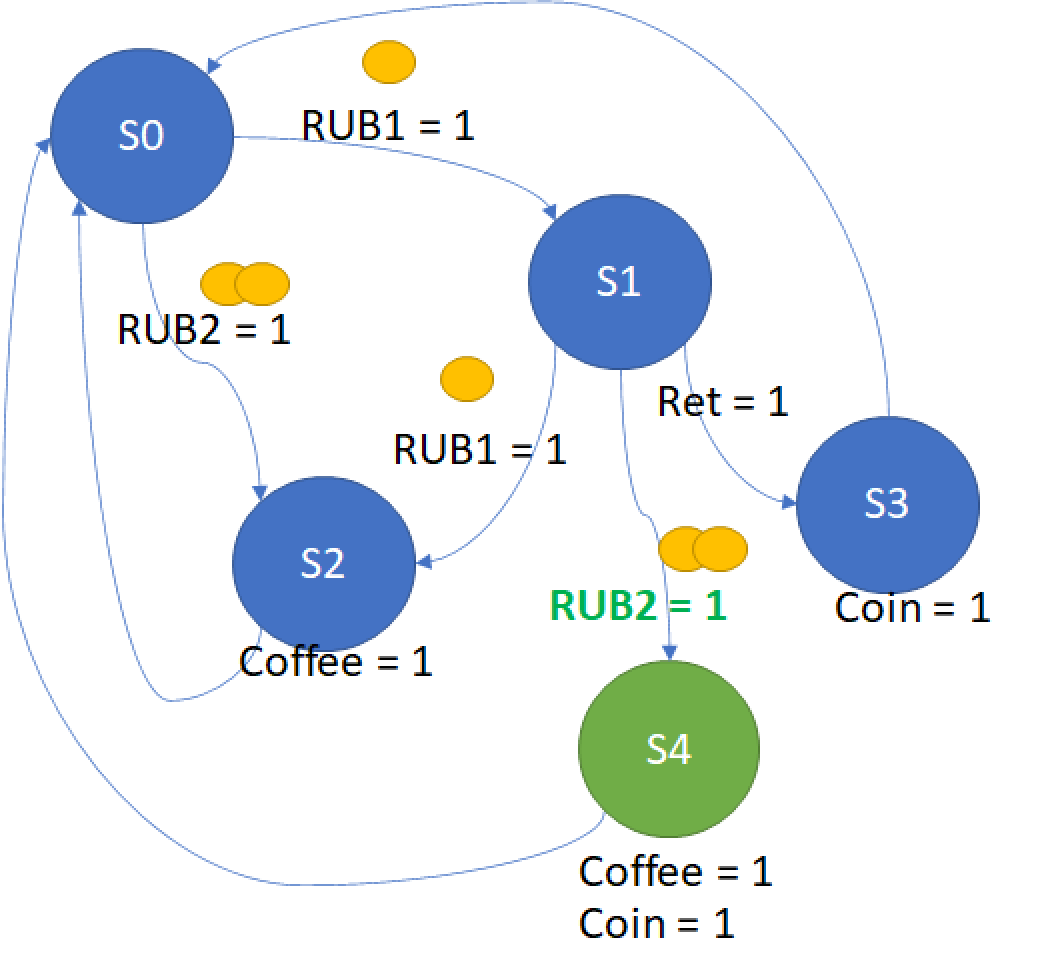
1. Опущен еще 1 рубль. Теперь автомат может готовить кофе, и для этого уже есть состояние S2. В него можно перейти, если КА уже был в состоянии S1, и сработал датчик RUB1. Таким образом, в S2 можно попасть, либо сразу опустив монету 2 рубля, либо последовательно опустив две монеты в 1 рубль.

2. Человек, опустивший одну монету, обнаружил, что у него больше нет подходящих монет. Он хочет получить сдачу, для чего нажимает кнопку Ret («возврат»). Тогда КА должен перейти в состояние S3, в котором формируется выходной сигнал Coin = 1, управляющий устройством по выдаче монеты 1 рубль.



*Рисунок 9.3 Граф описания конечного автомата по управлению продажей кофе*

Наконец, возможен еще один сценарий. Если после монеты 1 рубль будет опущена монета 2 рубля, это вполне допустимо, поскольку монетоприемник рассчитан и на те, и на другие монеты. Однако тогда автомат получит 3 рубля, и должен будет вернуть сдачу. На первый взгляд, уже существует состояние S2, в котором готовится кофе, но по правилам описания КА состояние нельзя модифицировать в зависимости от того, по какому пути автомат попал в него. Если требуется другое поведение («одновременно и варка кофе, и выдача сдачи»), необходимо добавить еще одно состояние. На рис. 9.4 это состояние S4, в котором есть два выходных сигнала – Coffee = 1, Coin = 1.



*Рисунок 9.4 Граф описания конечного автомата по управлению продажей кофе*

Представление КА в виде графа достаточно наглядно, хотя для графов с большим количеством узлов и сложной структурой переходов анализ может стать затрудненным. Визуальное представление позволяет, в частности, быстро отследить две проблемы, которые препятствуют нормальной работе КА:

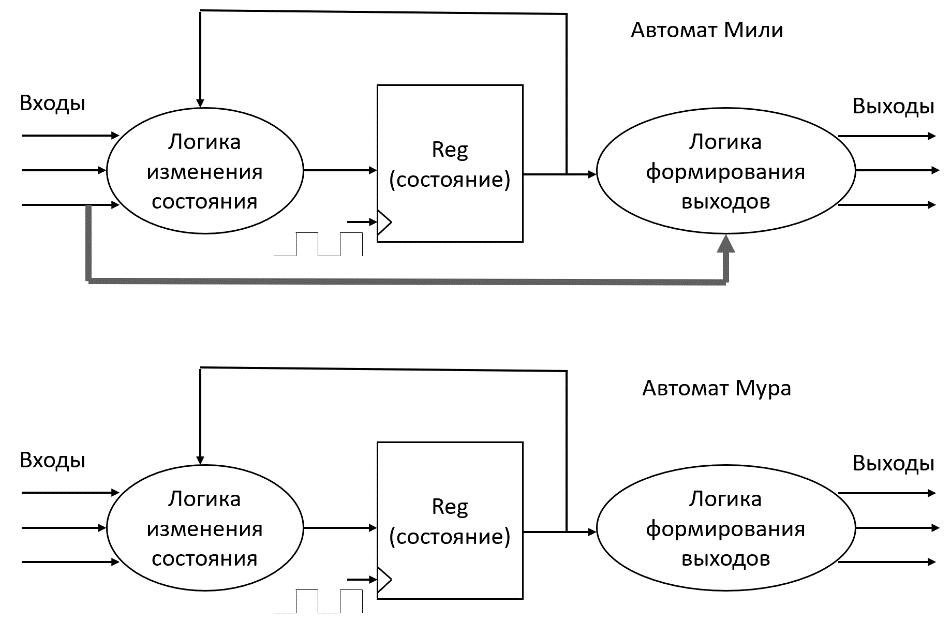
- тупиковые состояния – состояния, в которые можно попасть, но из которых не ведет ни одной выходной линии;

- недостижимые состояния – состояния, в которые невозможно попасть.

В обоих случаях реализованный КА будет работать хотя и в соответствии с разработанным графом, но скорее всего некорректно с точки зрения общей постановки задачи. В общем случае, тупиковых и недостижимых состояний быть не должно. Исключением может быть, например, состояние полной блокировки системы, в которое автомат может попасть один раз, а выход из которого производится путем полного перезапуска, включающего и инициализацию КА. Кроме того, в КА обычно предусмотрен сброс в начальное состояние (в приведенном примере это состояние S0, с которого начинается работа).

9.2 Автоматы Мили и Мура

Существуют две основные разновидности конечных автоматов, называемые автоматами Мили (Mealy) и Мура (Moore). Их основное отличие заключается в том, что значения выходных сигналов для автомата Мили зависят и от состояния автомата, и от значений входных сигналов, а для автомата Мура – только от состояния автомата. Эту разницу иллюстрирует рис. 9.5.



*Рисунок 9.5 Структуры автоматов Мили и Мура*

На первый взгляд, разница между этими автоматами несущественна и заключается в одной дополнительной связи – между входными и выходными сигналами (на рис. 9.5 она показана жирной линией). Автомат Мили превращается в автомат Мура при отсутствии прямой связи между входами и выходами. Однако недостаток автомата Мили с точки зрения цифровой схемотехники заключается именно в этой прямой связи. Сквозное прохождение сигналов через ПЛИС имеет большую задержку распространения сигнала, а в автомате Мура входные сигналы приходят в конечном итоге в группу регистров, хранящую состояние автомата. Поэтому в целом автомат Мура может рассчитывать на более высокую производительность.

9.3 Кодирование состояний конечного автомата.

Конечные автоматы могут использовать несколько схем кодирования состояний. Очевидной является двоичное кодирование, при котором каждому состоянию приписывается номер, представляемый в двоичном коде. Для N-разрядного регистра возможно представление 2^N состояний автомата, поэтому такой способ не только очевиден, но и наиболее эффективно использует триггеры ПЛИС.

С другой стороны, для проверки того, что КА находится в определенном состоянии, потребуется проверить N-разрядное число. Такая проверка заложена в операторе case, для каждой ветки необходимо проверить регистр состояния на равенство некоторой константе.

Одним из альтернативных вариантов является т.н. one-hot кодирование. Из названия one-hot видно, что при таком кодировании только один разряд регистра состояния равен 1. Таким образом, в N разрядах можно закодировать только N состояний. По сравнению с двоичным кодированием это приведет к расточительному использованию триггеров ПЛИС, поскольку при двоичном кодировании для 256 состояний потребуется 8 триггеров, а для one-hot – 256 триггеров. Однако для проверки каждой из веток оператора case достаточно проверить только один триггер.

Преимущество one-hot кодирования проявляются при небольшом количестве состояний КА. В этом случае рост количества триггеров еще не так существенен, но уже имеется эффект от простой проверки состояния.

Другими вариантами кодирования являются LFSR, счетчик Джонсона, счетчик Грэя или сдвиговый регистр с определенным кодом, не повторяющимся при сдвиге. Такие способы кодирования могут быть эффективны, например, при наличии в КА большого количества последовательных переключений.

Синтезаторы языков описания аппаратуры обычно имеют режим автоматического выбора кодирования конечных автоматов. Достаточно установить соответствующую настройку в Auto, чтобы был подобран оптимальный способ кодирования состояний КА (также можно задать этот способ принудительно).

Часто полезно применять для описания состояний КА перечислимый тип (enumerated type). В перечислимом типе можно указать символические обозначения состояний, которые должен принимать сигнал, как показано в примере.

**enum reg**[2:0] {S0, S1, S2, S3, S4} state;

При описании в виде перечислимого типа синтезатор может автоматически подобрать тип кодирования состояний, наиболее подходящий для описанного КА. На практике автоматы с большим количеством состояний кодируются в двоичном виде, а при небольшом количестве используются варианты one-hot, Johnson, LFSR.

9.4 Однопроцессное и трехпроцессное описание конечного автомата

Самым простой способ разработки КА – описать все переходы между состояниями графа внутри одного оператора case.

always @(posedge clk)  
begin  
case (state)   
 3’b000 : state <= …; output\_reg1 <= …  
 3’b001 :   
 3’b010 :   
 3’b011 :   
 3’b100 :   
 default : q = 1’bx;  
endcase   
end

В каждой ветке оператора case необходимо установить следующее состояние и выходные сигналы. Для автомата, показанного на рис. 9.4, описание модуля на Verilog может выглядеть следующим образом.

module coffee\_fsm(

input clk,

input rub1,

input rub2,

input ret,

output reg coffee,

output reg coin

);

reg [2:0] state;

always @ (posedge clk)

begin

case (state)

3'b000 : if (rub1)

begin

state <= 3'b001;

coffee <= 1'b0;

coin <= 1'b0;

end else

if (rub2)

begin

state <= 3'b010;

coffee <= 1'b1;

coin <= 1'b0;

end

3'b001 : if (rub1)

begin

state <= 3'b010;

coffee <= 1'b1;

coin <= 1'b0;

end else

if (rub2)

begin

state <= 3'b100;

coffee <= 1'b1;

coin <= 1'b1;

end else

if (ret)

begin

state <= 3'b011;

coffee <= 1'b0;

coin <= 1'b1;

end

3'b010 : begin

state <= 3'b000;

coffee <= 1'b0;

coin <= 1'b0;

end

3'b011 : begin

state <= 3'b000;

coffee <= 1'b0;

coin <= 1'b0;

end

3'b100 : begin

state <= 3'b000;

coffee <= 1'b0;

coin <= 1'b0;

end

endcase

end

endmodule

Можно воспользоваться полезным приемом под названием «мультипроцессное описание КА». В примере, показанном выше, все описание, по сути, было сведено к одному оператору case, в отдельных ветках которого описывались все действия, выполняемые в конкретном состоянии. При возрастании сложности автомата его работу можно разделить на три процесса:

– процесс назначения состояния;

– процесс декодирования входных сигналов и определения нового состояния;

– процесс назначения выходных сигналов.

Процесс назначения состояния является наиболее простым. Его буквальный вид показан в следующем листинге.

always @ (posedge clk)

begin

if (reset) state <= initial\_state

else state <= new\_state;

end

В данном примере появились новые сигналы initial\_state (это, по сути, константа, которая требуется для перевода КА в начальное состояние) и new\_state. Сигнал new\_state должен формироваться комбинационной логикой в отдельном процессе. Процесс может быть описан с помощью оператора case или с помощью цепочки вложенных операторов if.

always @ \*

begin

case (state)

3'b000 : …

endcase

always @ \*

begin

if (state = 3’b000) new\_state <=

…

При небольшой сложности автомата можно использовать оператор assign.

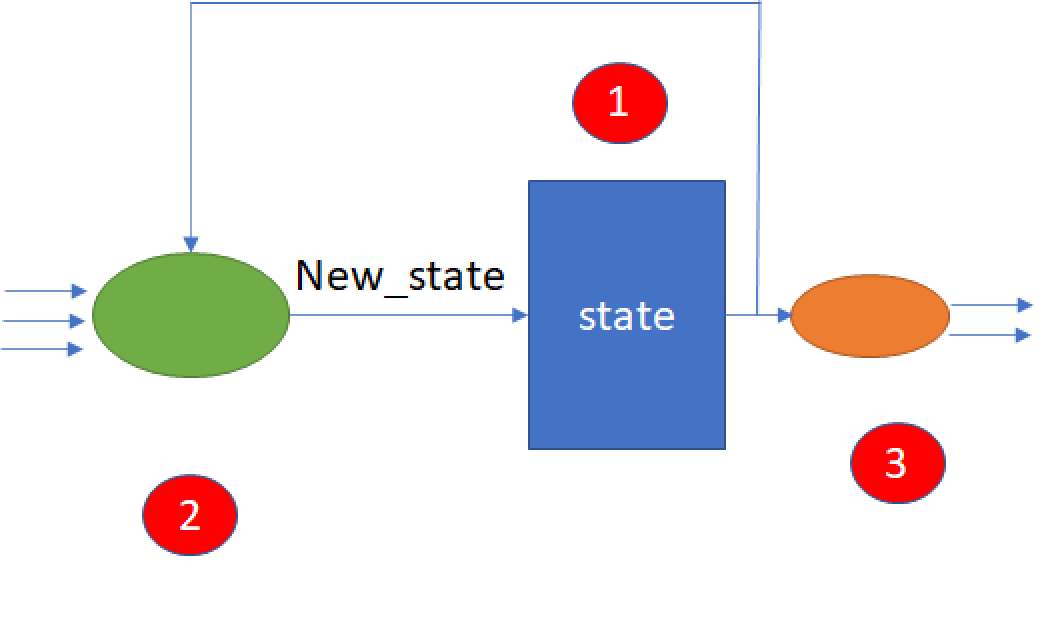
assign new\_state = …

При использовании процессов необходимо отслеживать формирование защелок. Если в какой-то ветке оператора case новое состояние new\_state будет не определено, для него будет синтезирована защелка (latch) на основе специального режима триггера. Защелка не рекомендуется к применению в синхронном стиле проектирования.

Наконец, в третьем процессе необходимо назначить состояния выходным сигналам. В зависимости от типа автомата (Мура или Мили) потребуется анализ только текущего состояния, или состояния и входных сигналов. Выделение отдельного процесса может привести к более простому описанию, которое проще анализировать и сопровождать. Например, для кофейного КА можно заметить, что включение варки кофе происходит в состояниях S2 и S4, а выдача сдачи – в S3 и S4.

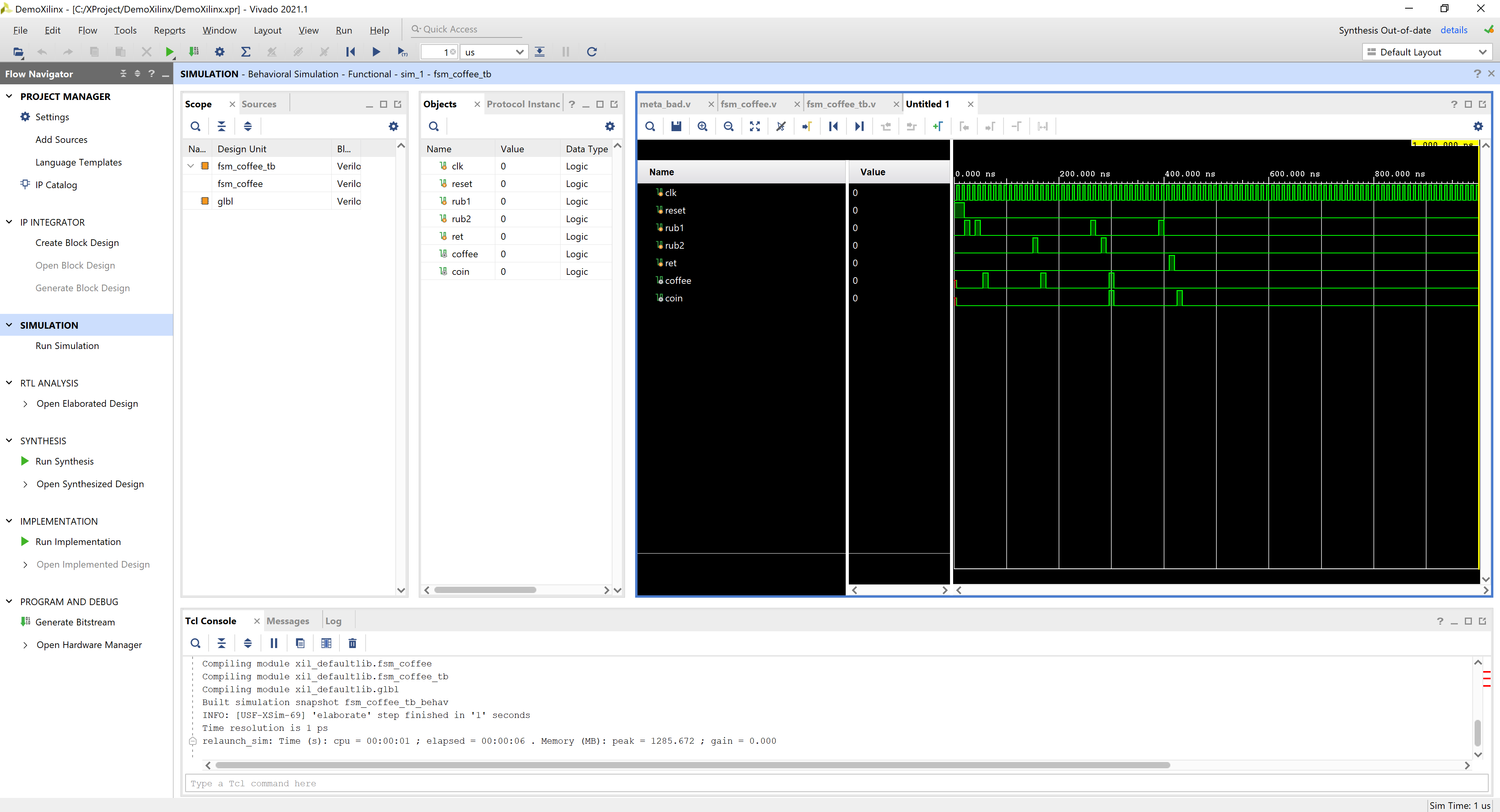
Латентность (latency), т.е. дополнительная задержка в тактах на прохождение сигнала по цепи обработки данных может оказаться неприемлемой для многих устройств. Однако можно указать и множество вариантов, когда дополнительные регистры на пути входных или выходных данных не влияют на принципиальную работоспособность схемы. Например, если речь идет о механических переключателях или кнопках, то момент распознавания автоматом их срабатывания совершенно неважен. Поскольку реакция человека оценивается в доли секунды, лишние десятки наносекунд, обусловленные прохождением сигнала от кнопки через один или несколько регистров, очевидно не будут замечены при наблюдении за работой устройства. Аналогично, сигнальные светодиоды могут срабатывать на несколько тактов позже. Электрические моторы также обладают инерцией, на порядки больше периода тактовой частоты проекта на ПЛИС, поэтому управляющие сигналы для них также могут быть конвейеризованы.

На рис. 9.6 показана иллюстрация, облегчающая понимание трехпроцессного описания конечного автомата. Три показанные компонента КА распределяются по трем процессам в языке описания аппаратуры, что упрощает анализ каждого из этих процессов. Например, процесс назначения нового состояния предельно прост, и заключается в том, что в текущее состояние записывается «новое состояние». Тогда можно сосредоточиться на анализе того, по каким правилам определяется это новое состояние, однако текст описания не перемешан с назначением сигнала state и выходных сигналов.



*Рисунок 9.6 Схема конечного автомата с компонентами, соответствующими процессам в RTL-описании*

При моделировании конечного автомата следует проверять основные сценарии его работы. Временные диаграммы работы КА показаны на рис. 9.7.



*Рисунок 9.7 Временные диаграммы работы конечного автомата по управлению продажей кофе*

В примере можно видеть, что проверены все основные маршруты графа:

1. Последовательное опускание двух монет по 1 рублю – можно наблюдать появление сигнала coffee.

2. Опускание монеты 2 рубля также приводит к появлению сигнала coffee.

3. Последовательное опускание монет 1 рубль и 2 рубля приводит к одновременному появлению сигналов coffee и coin.

4. Опускание монеты 1 рубль и последующее нажатие кнопки «возврат» приводит к появлению сигнала coin.

Таким образом, моделирование позволяет проверить корректность описания КА и его соответствие тем соображениям, которые имелись у разработчика. Для КА с большим числом состояний и возможных переходов исчерпывающее моделирование может быть затруднено.

9.5 Выводы по разделу

Конечный автомат – эффективное средств реализации схем, последовательно переключающихся в зависимости от внешних управляющих сигналов. Он может быть построен на основе графа, описывающего условия перехода между состояниями. При проектировании автомата можно на ранних стадиях убедиться в его формальной корректности – например, отсутствии противоречивых условий перехода, отсутствии недостижимых и тупиковых состояний.

Можно для краткости представить, что строки программы для процессора соответствуют состояниям конечного автомата. Тогда можно реализовать автомат, который будет, последовательно переходя между состояниями, выполнять в каждом состоянии отдельный оператор.

Существует множество инструментов проектирования и анализа конечных автоматов, в том числе алгоритмы синтеза цифровых схем по RTL-описаниям. В целом конечный автомат соответствует правилам синхронного проектирования и хорошо поддерживается ячейками FPGA.

В зависимости от сложности автомата может использоваться однопроцессное описание, в котором все правила работы находятся внутри одного процедурного блока. Альтернативой является разделение автомата на три процесса (трехпроцессное описание), в котором отдельно описывается запись нового состояния, вычисление нового состояния и назначение выходов

Контрольные вопросы:

1. Могут ли два состояния конечного автомата иметь одинаковые комбинации выходных сигналов? Можно ли заменить их в таком случае одним состоянием?

2. Сколько разрядов требуется для представления 8 состояний КА при двоичном кодировании? При one-hot кодировании?

3. Как с помощью конечного автомата реализовать управление мигающим светодиодом? Тактовая частота внутри схемы существенно больше, чем требуемая частота мигания.

4. Что такое тупиковые и недостижимые состояния КА и почему их следует избегать?

5. Что описывается в отдельных процессах при трехпроцессном описании конечного автомата?