**Синтез синхронных цифровых автоматов с использованием ПЛИС**

ПЛИС, относящиеся к зарубежному классу FPGA-микросхем являются в общем случае набором логических вентилей, из которых строится более укрупненная блочная структура, которая включает в себя:

* логические блоки, состоящие из триггера, комбинационной схемы (LUT) и мультиплексора
* матрицы межсоединений
* портов ввода-вывода (GPIO)

Как и с любой другой цифровой схеме все сигналы распространяются параллельно друг другу во времени, хотя текст программы на языке HDL (VHDL, Verilog, SystemC и т.п.) идет последовательно строка за строкой, но все эти строки выполняются параллельно. Большинство задач предусматривают ожидание наступления определенного события и переход в т, либо иное состояние в зависимости от текущего состояния и ряда условий (сигналов). Для решения таких задач используют синхронные цифровые автоматы, являющиеся отдельным классом последовательностных схем. Наиболее известными типами цифровых автоматов являются автоматы Милли и Мура:

* в автомате Мура выходы схемы являются функцией только текущего состояния
* в автомате Милли – функцией внутреннего состояния и по крайней мере одного из входов.

В общем случае цифровой автомат математически можно описать А = <A, B, C, δ, λ> [1-2], где:

множество A – множество значений на физических входах цифрового автомата

множество B – множество значений на физических выходах цифрового автомата

множество C – множество, которое представляет собой внутренне состояние автомата

δ – функции переходов автоматов.

λ – функции выходов автоматов

Автомат Милли описывается следующей системой уравнений:

c(t) = δ(a(t), c(t-1))

b(t) = λ(a(t), c(t-1))

Автомат Мура описывается следующей системой уравнений:

c(t) = δ(a(t), c(t-1))

b(t) = λ(a(t), c(t))

Автомат функционирует дискретно во времени, т.е. значения входов, выходов и внутреннее состояние изменяются в дискретные моменты времени. Примерами таких автоматов являются: триггер, регистр и т.п.

Есть два основных способа задания автомата:

1. с помощью графа
2. с помощью таблицы функций переходов и функций выходов.

Граф автомата – это ***ориентированный связанный граф***, вершины которого представляют внутреннее состояние, а ***дуги – переходы из одного состояния в другое***.

Пример задания цифрового автомата (автомата Милли) приведен на рис.1. ***Над дугами***, связывающими состояния пишутся буквы, ***входных и выходных состояний***, при этом ***выходное состояние зависит от состояния автомата в предыдущие моменты времени***. Автомат, приведенный на рис.1 можно задать с помощью таблиц переходов и выходов (ТПВ). В ТПВ строками являются внутренние состояния автомата, а столбцами – входы. ТПВ автомата, приведенного на рис.1. Трактовать значения приведенные на пересечении строк и столбцов следует следующим образом, рассмотрим выделенную оранжевым цветом ячейку автомат находится в состоянии С1, при наличии на его входах значения a2 он перейдет в состояние С2, а на выходах автомата будет значение b3.

Изображение выглядит как текст

Автоматически созданное описание

Рис.1. Пример задания автомата Милли.

Таблица 1 – ТПВ автомата, приведенного на рис.1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | a1 | a2 | a3 |
| C1 | C1/b1 | C2/b3 | C3/b4 |
| C2 | - | - | C1/b1 |
| C3 | C4/b1 | C2/b4 | C2/b3 |
| C4 | C4/b2 | - | C2/b2 |

При описании с помощью графа автомата Мура над дугами записываются только входные значения, выходные остаются у вершин, пример графа автомата Мура приведен на рис.2.

Изображение выглядит как карта, текст

Автоматически созданное описание

Рис.2. Пример задания автомата Мура.

При построении ТПВ для графа Мура строят отмеченную таблицу переходов для которой выделяют дополнительный столбец справа, в котором пишут выходное состояние. Для автомата изображенного на рис.2. ТПВ приведена в табл.2.

Таблица 2 – ТПВ автомата Мура, приведенного на рис.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a1 | a2 | a3 |  |
| С1 | C2 | C3 | C1 | b3 |
| С2 | - | C2 | C1 | b2 |
| С3 | C3 | - | C2 | b1 |

Таблицу ТПВ для автомата Мура следует понимать следующим образом: в какое состояние автомат переходит если он находится в определенном состоянии при наличии на входах одного из значений и при этом на выходе схему будет следующее. Для примера рассмотри выделенную оранжевым клетку таблицы 2: автомат находясь в состоянии С2 перейдет в состоянии С1 по поступлении на вход сигналов a3, при этом в состоянии C2 на его выходах значение b2.

Оба приведенных автомата являются частичными, т.е. не для всего набора входных значений (a) определены переходы в состояние, для того, чтобы автомат можно было реализовать необходимо доопределить состояние автомата, простейший вариант – оставить автомат в том же состоянии, в котором он находился, хотя это не всегда будет соответствовать логике работы устройства.

Для выполнения операций синхронно (по наступлению события) используются always-блоки. В этой работе для синтеза цифрового автомата будет использоваться описание язык описания цифровых схем Verilog [3-5] Существует несколько подходов для описание цифровых автоматов с использованием одного или двух и более always блоков [6,7]. Авторы этих работ заявляют, что при использовании двухчастных автоматов их работа является более стабильной, однако, ***в рамках этой лабораторной работы*** мы будем использовать ***задание автомата с использованием одного always блока***. Реализуем, приведенный на рис.1. и в табл.1. автомат, пример (Quartus и ModelSim проекты) расположены здесь:

*\FpgaCourse\Basics\verilog\Labs\SyncAutomates\example\_automates\milley\_automate*

Verilog код модуля, описывающий работу автомата Милли:

**module milley\_automate(**

**input wire reset,**

**input wire clk,**

**input wire [1:0] a,**

**output reg [1:0] b**

**);**

**// a1 - 1, a2 - 2, a3 - 3**

**// b1 - 0, b2 - 1, b3 - 2, b4 -3**

**localparam reg [1:0] A1 = 2'b01;**

**localparam reg [1:0] A2 = 2'b10;**

**localparam reg [1:0] A3 = 2'b11;**

**localparam reg [1:0] B1 = 2'b00;**

**localparam reg [1:0] B2 = 2'b01;**

**localparam reg [1:0] B3 = 2'b10;**

**localparam reg [1:0] B4 = 2'b11;**

**localparam reg [1:0] C1 = 2'b00;**

**localparam reg [1:0] C2 = 2'b01;**

**localparam reg [1:0] C3 = 2'b10;**

**localparam reg [1:0] C4 = 2'b11;**

**reg[1:0] c;**

**always @(posedge clk)**

**begin**

**if (reset)**

**begin**

**c <= C1;**

**b <= 0;**

**end**

**else**

**begin**

**case (c)**

**C1:**

**begin**

**if (a == A2)**

**begin**

**c <= C2;**

**b <= B3;**

**end**

**else if (a == A3)**

**begin**

**c <= C3;**

**b <= B4;**

**end**

**else // доопределение состояния автомата**

**begin**

**c <= C1;**

**b <= B2;**

**end**

**end**

**C2:**

**begin**

**if (a == A3)**

**begin**

**c <= C1;**

**b <= B1;**

**end**

**else // доопределение состояния автомата**

**begin**

**c <= C2;**

**// мы не изменяем значение b**

**end**

**end**

**C3:**

**begin**

**if (a == A1)**

**begin**

**c <= C4;**

**b <= B1;**

**end**

**else if (a == A2)**

**begin**

**c <= C1;**

**b <= B3;**

**end**

**else if (a == A3)**

**begin**

**c <= C2;**

**b <= B4;**

**end**

**else**

**begin**

**c <= C3;**

**end**

**end**

**C4:**

**begin**

**if (a == A1)**

**begin**

**c <= C4;**

**b <= B2;**

**end**

**else if (a == A3)**

**begin**

**c <= C2;**

**b <= B2;**

**end**

**else**

**begin // доопределение автомата, мы не изменяем значение b**

**c <= C4;**

**end**

**end**

**default:**

**begin**

**b <= B1;**

**c <= C1;**

**end**

**endcase**

**end**

**end**

**endmodule**

Данный цифровой автомат был доопределен следующим образом:

1. Добавлено еще одно состояние для инициализации всех reg переменных (состояние сброса), сброс выполнен синхронно, что означает. Что в процессе сброса необходимо тактирование модуля.
2. Для состояний С1 и С4 возможен переход в то же самое состояние с установкой выхода в значение B2 по значению A1 на входе, однако, для других входных комбинаций a этот автомат доопределен таким образом, что состояние выходов в B не изменяется

Для этого модуля создан ModelSim проект и написан пример testbench’а (пример, поскольку он ***не выполняет тестирование всех переходов*** и всего набора входных параметров, что ***должно быть реализовано при проектировании*** промышленных синхронных цифровых автоматов.

Verilog код testbench модуля:

module milley\_automate\_testbench();

reg [1:0] a;

**wire [1:0] b;**

**reg clk;**

**reg reset;**

**reg [31:0] counter;**

**milley\_automate uut(.clk(clk), .reset(reset), .a(a), .b(b));**

**initial**

**begin**

**clk <= 0;**

**reset <= 1;**

**a <= 0;**

**counter <= 0;**

**# 400**

**reset <= 0;**

**end**

**always**

**begin**

**#50 clk <= ~clk;**

**#100 counter <= counter + 1;**

**if (counter == 5)**

**begin**

**a <= 2'b11; // A3, c -> C3, b -> b4**

**end**

**if (counter == 7)**

**begin**

**a <= 2'b01; // A1, c->C4**

**end**

**// B state should not changed until a1**

**if (counter == 8)**

**begin**

**a <= 2'b00;**

**end**

**if (counter == 11)**

**begin**

**a <= 2'b01;**

**end**

**if (counter == 13)**

**begin**

**a <= 2'b11;**

**end**

**if (counter == 14)**

**begin**

**a <= 2'b11;**

**end**

**if (counter == 16)**

**begin**

**a <= 0;**

**counter <= 0;**

**end**

**end**

**endmodule**

Результат работы testbench приведен на рис.3.

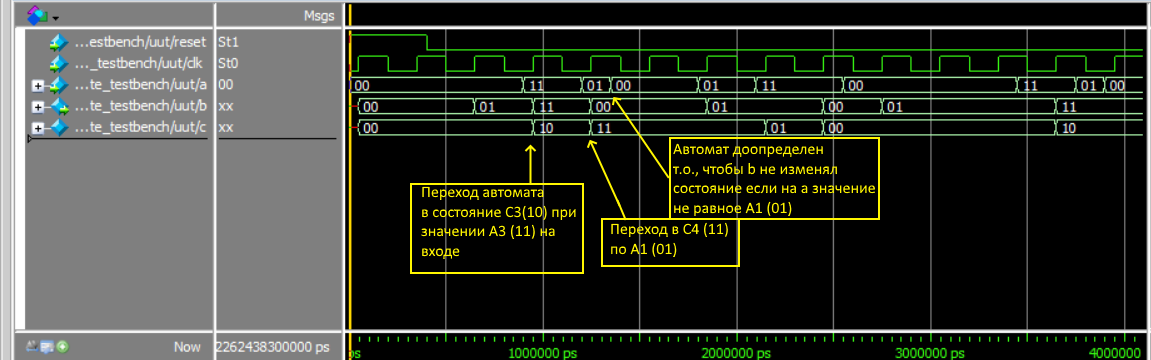


Рис.3. Результат проверки работы автомата Милли.

Следует отметить, что в приведенном выше testbench модуле отсутствует проверка значений и вывод их через функцию display, например:

**if(b != 2'b11)**

**$display("test failed for input combination 00");**

Аналогичным образом выполним реализацию автомата Мура и проверим его работоспособность, основное отличие в том, что состояние b привязано к текущему состоянию автомата с через инструкцию assign и выбор значения из функции. Проект расположен по пути:

*FpgaCourse\Basics\verilog\Labs\SyncAutomates\example\_automates\moore\_automate*

Verilog код модуля, реализующий автомат Мура, приведенный на рис.2 и в табл. 2:

**module moore\_automate(**

**input wire reset,**

**input wire clk,**

**input wire [1:0] a,**

**output wire [1:0] b**

**);**

**// a1 - 1, a2 - 2, a3 - 3**

**// b1 - 0, b2 - 1, b3 - 2, b4 -3**

**localparam reg [1:0] A1 = 2'b00;**

**localparam reg [1:0] A2 = 2'b01;**

**localparam reg [1:0] A3 = 2'b10;**

**localparam reg [1:0] B1 = 2'b00;**

**localparam reg [1:0] B2 = 2'b01;**

**localparam reg [1:0] B3 = 2'b10;**

**localparam reg [1:0] C1 = 2'b00;**

**localparam reg [1:0] C2 = 2'b01;**

**localparam reg [1:0] C3 = 2'b10;**

**reg[1:0] c;**

**assign b = get\_output\_state(c);**

**always @(posedge clk)**

**begin**

**if (reset)**

**begin**

**c <= C1;**

**end**

**else**

**begin**

**case (c)**

**C1:**

**begin**

**if (a == A1)**

**c <= C2;**

**else if (a == A2)**

**c <= C3;**

**else**

**c <= C1;**

**end**

**C2:**

**begin**

**if (a == A3)**

**c <= C1;**

**else**

**c <= C2;**

**end**

**C3:**

**begin**

**if (a == A3)**

**c <= C2;**

**else**

**c <= C3;**

**end**

**default:**

**begin**

**c <= C1;**

**end**

**endcase**

**end**

**end**

**function [1:0] get\_output\_state;**

**input [1:0] state;**

**case (state)**

**C1:**

**get\_output\_state = B3;**

**C2:**

**get\_output\_state = B2;**

**C3:**

**get\_output\_state = B1;**

**default:**

**get\_output\_state = B1;**

**endcase**

endfunction

endmodule

Аналогично автомату Милли реализуется тестбенч для автомата Мура:

**module moore\_automate\_testbench();**

**reg clk;**

**reg reset;**

**reg [1:0] a;**

**wire [1:0] b;**

**reg [31:0] counter;**

**moore\_automate uut(.clk(clk), .reset(reset), .a(a), .b(b));**

**initial**

**begin**

**clk <= 0;**

**reset <= 1;**

**a <= 0;**

**counter <= 0;**

**# 300**

**reset <= 0;**

**end**

**always**

**begin**

**#50 clk <= ~clk;**

**#100 counter <= counter + 1;**

**if (counter == 5)**

**begin**

**a <= 2'b00; // A1, c -> C2**

**end**

**if (counter == 7)**

**begin**

**a <= 2'b10; // A3, c -> C1**

**end**

**if (counter == 8)**

**begin**

**a <= 2'b01; // A2, c -> C3**

**end**

**if (counter == 11)**

**begin**

**a <= 2'b10; // A3 -> c2**

**counter <= 0;**

**end**

**end**

**endmodule**

Проверка работоспособности модуля на тестбенч приведена на рис.4. (также как и для автомата Милли тестбенч выполнен только для демонстрации и поэтому не проверяет всех последовательностей смены состояния и отсутствуют проверки через display(), см. выше)

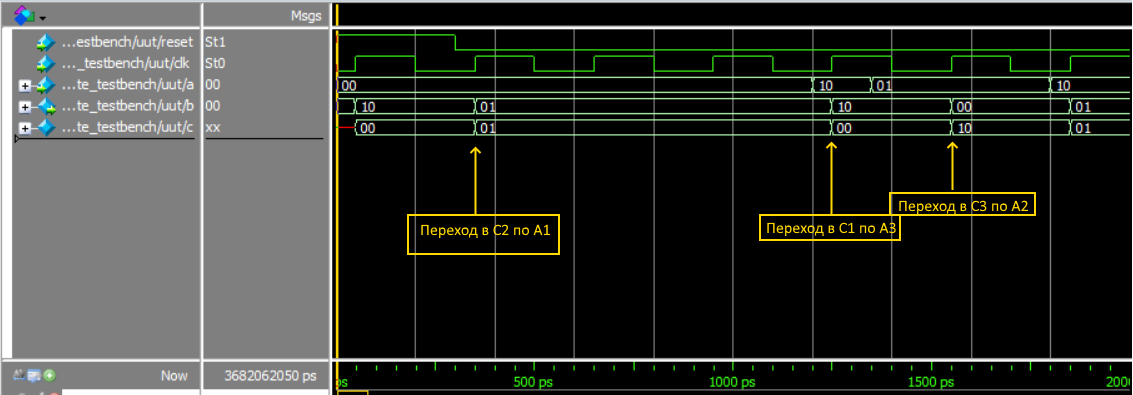


Рис. 4. Проверка работоспособности автомата Мура, приведенного на рис.2 и в табл.2.

В заключение следует отметить, что два автомата были разработаны случайным образом с помощью графа, имена переменных a, b и с следует задавать такими, чтобы они отражали назначение автомат, например, dram\_address, dram\_control, dram\_state и т.п.

**Задание для самостоятельной работы**

Необходимо:

1. нарисовать граф, описывающий работу синхронного цифрового автомата
2. описать код цифрового автомата с использованием языка Verilog
3. написать тестбенч для проверки цифрового автоматов и выполнить проверку переходов

Для реализации выбрать один из следующих вариантов цифрового автомата:

1. Автомат анализа двоичных тетрад. В последовательном синхронном потоке 4х разрядных кодовых последовательностей (код 8421) распознать коды не соответствующие десятичным цифрам (0000-1001 соответствует, 1010-1111 нет), данные поступают синхронно по однобитной линии в формате MSB First (старшими битами вперед), автомат выдает сигнал 0, если кодовая последовательность соответствую кодировке и 1 если не соответствует.

2. Автомат анализа двоичных триад. По последовательному одноразрядному+ каналу синхронно поступает двоичная информация. Анализируются триады бит, если на втором такте значение на линии лог.1., то на третьем такте формируется лог.1. на выходе автомата Z, см. рис. 5.

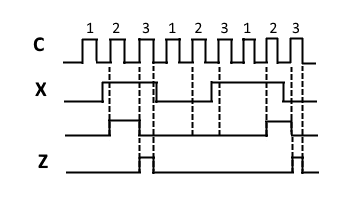


Рис.5. Логика работы автомата по анализу триад.

3. Автомат подсчета числа посетителей/автотранспорта: существует два близкорасположенных оптических датчика (источник-приемник), которые выдают лог.0, когда датчик перекрыт и лог. 1 когда датчик не перекрыт (см. рис.6). Примем допущение о том, что автомат синхронизируется настолько тактовыми импульсами настолько высокой частоты, что данный автомат функционирует подобно асинхронному автомату. На выход должно выводиться число посетителей в виде 16-разрядного числа

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рис.6. Автомат подсчета числа посетителей

4. Автомат контроля продукта в накопительной емкости. Существует накопительный бак (см. рис.7.), в который накачивается продукт насосами P1 и P2, в баке имеется 3 оптических датчика уровня продукта -a, b, c (c расположен выше b, который в свою очередь расположен выше a). автомат должен обеспечивать автоматическое управление насосами в зависимости от состояния датчиков (датчик формирует лог.1, если он не перекрыт и 0, если перекрыт). Если уровень в баке меньше чем a, то работает оба насоса (подача лог.1 на насос означает, что он включен), если выше b, то только P2 и если выше с, то оба насоса выключены.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рис.7. Автомат контроля уровня продукта в накопительном баке.

**Список использованных источников**

1. Безуглов Д.А., Калиенко И.В. Цифровые устройства и микропроцессоры. Ростов н/Д : Феникс. 2008.
2. Захаров Н.Г., Рогов В.Н. Синтез цифровых автоматов. Ульяновск : УлГТУ. 2003.
3. Краткий курс Verilog: <http://e-learning.bmstu.ru/moodle/pluginfile.php/2895/mod_folder/content/0/Viki_kratkii_kurs_verilog.pdf?forcedownload=1> (дата обращения 19.12.2019)
4. Руководство по Verilog от компании Xilinx: <http://in.ncu.edu.tw/ncume_ee/digilogi/vhdl/Verilog_Reference_Guide.pdf> (дата обращения 19.12.2019)
5. Поляков А.К. Языки VHDL и Verilog в проектировании цифровой аппаратуры. М.:Солон-Пресс. 2003.
6. Clifford E.C. The fundamentals of efficient synthesizable finite state machine design using nc-verilog and buildgates. Proceedings of International Cadence Usergroup Conference. San Jose. 2002. P. 1-27.