Методичка по созданию автомата в среде Quartus в картинках на примере построения автомата с двумя состояниями, реализующего поведение триггера.

**Прим.: Данный способ, всего лишь, пример использования инструмента для создания автомата и, на самом деле, является наихудшим вариантом реализовать триггер, из всех которые только можно или, даже, нельзя придумать в языках VHDL и Verilog.**

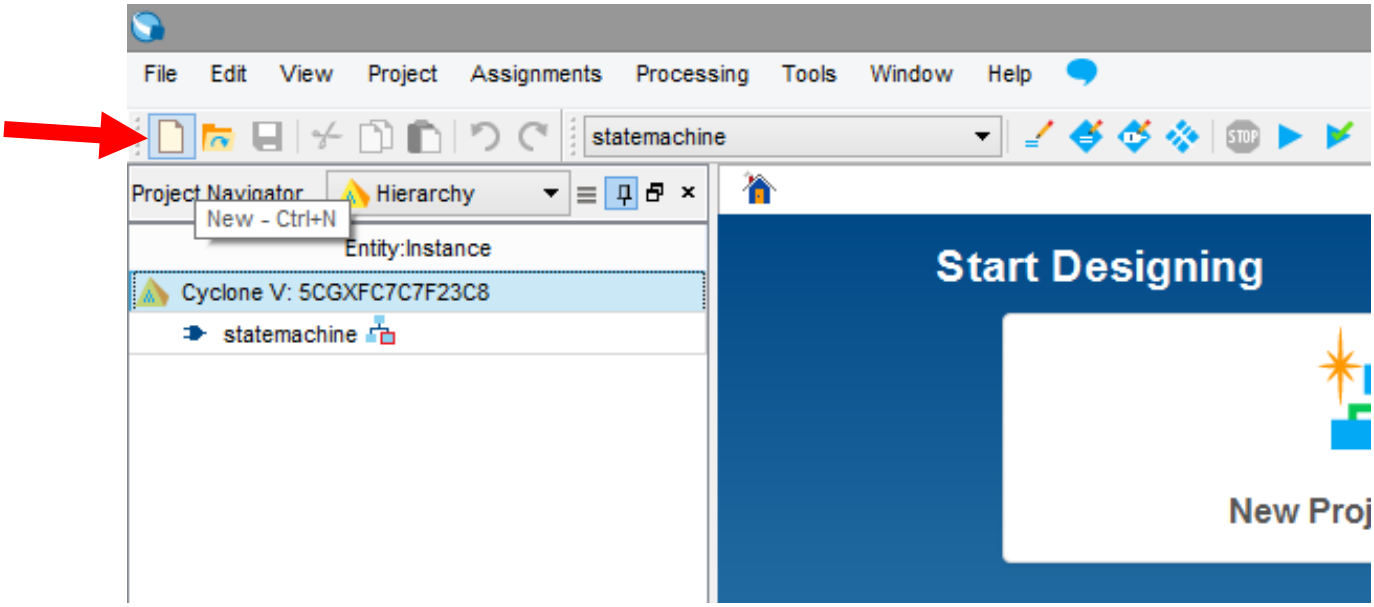


*Рисунок 1. Создаём новый документ*

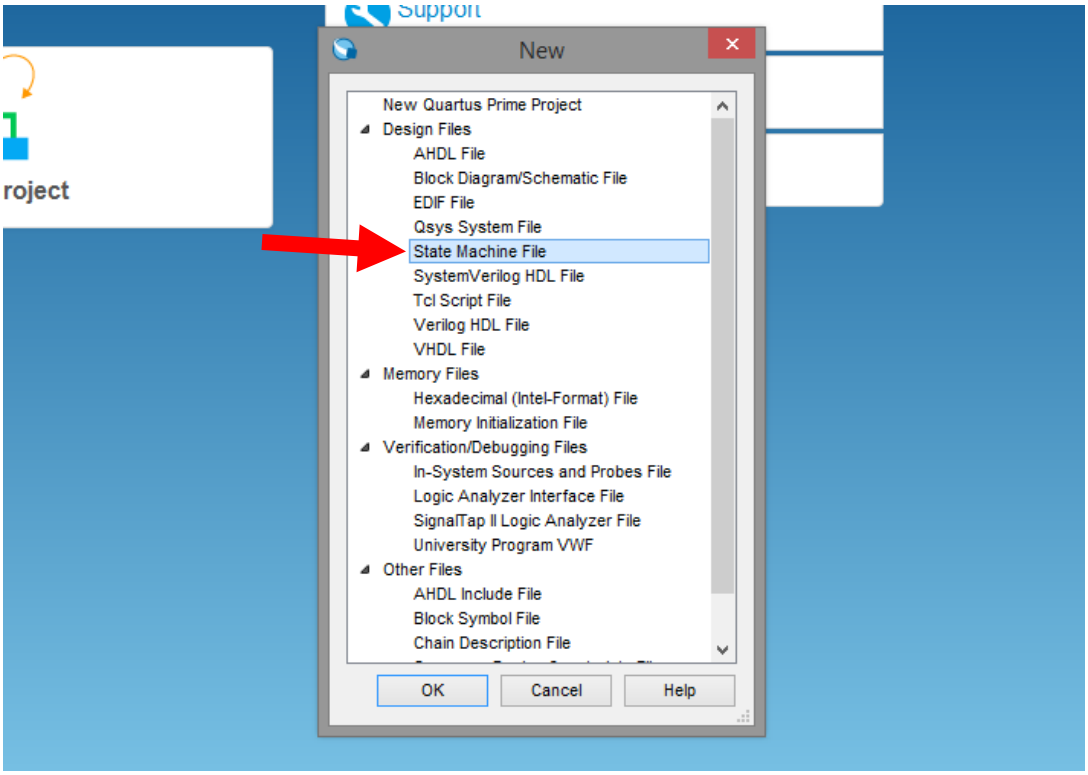


*Рисунок 2. В появившемся окне выбираем "State Machine File"*

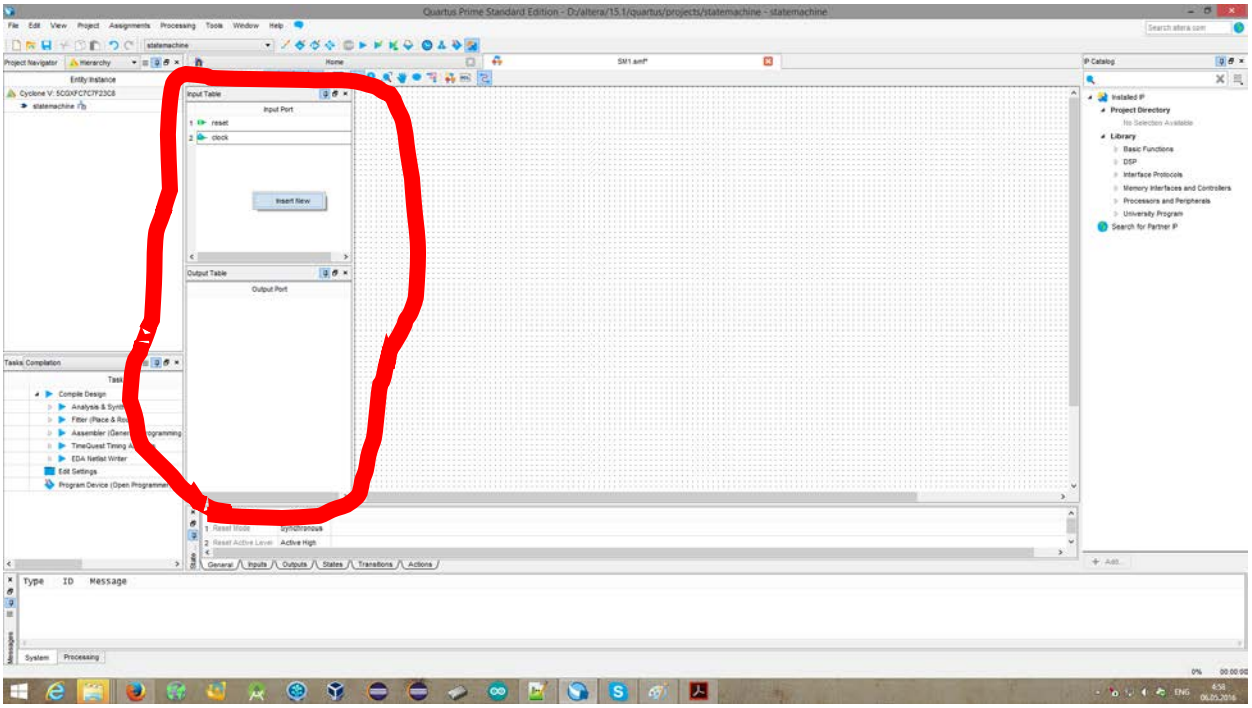

*Рисунок 3. В этой области мы можем видеть входные и выходные порты автомата. Нажатием правой кнопки мыши можно вызвать меню, позволяющие добавить новые порты.*

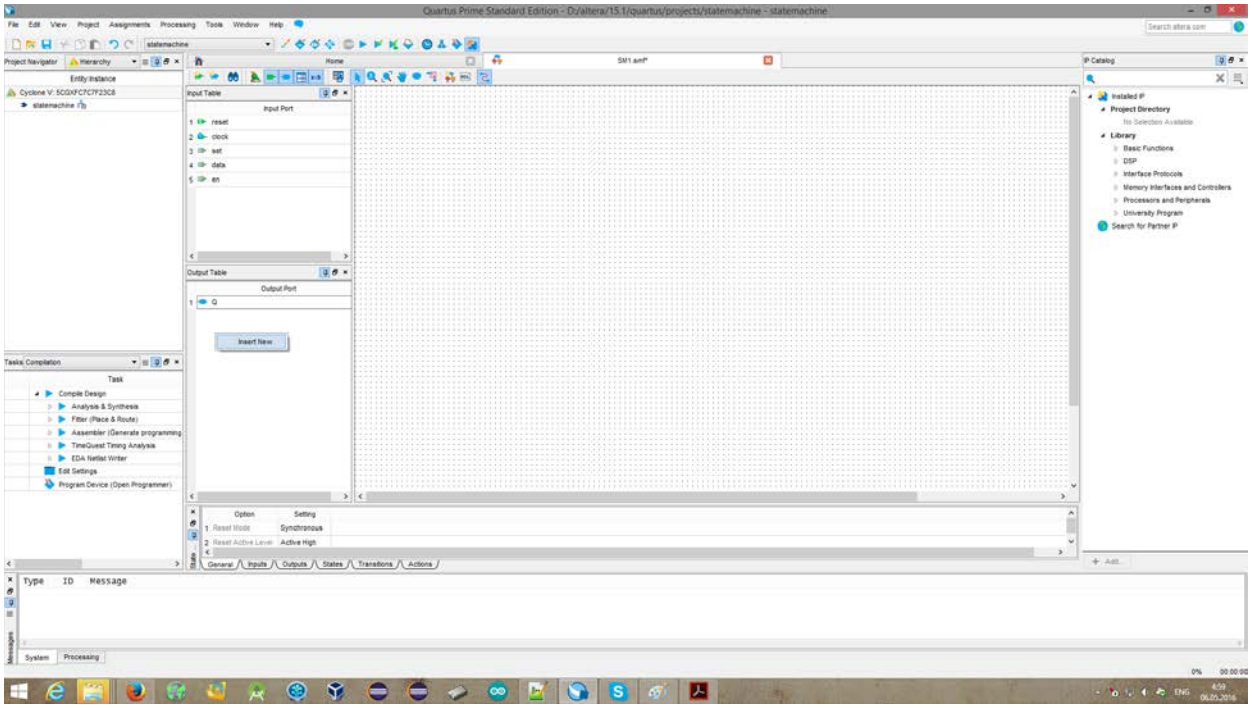

*Рисунок 4. После добавления входных портов, необходимых любому триггеру, добавляем выходной порт (для триггера достаточно одного).*

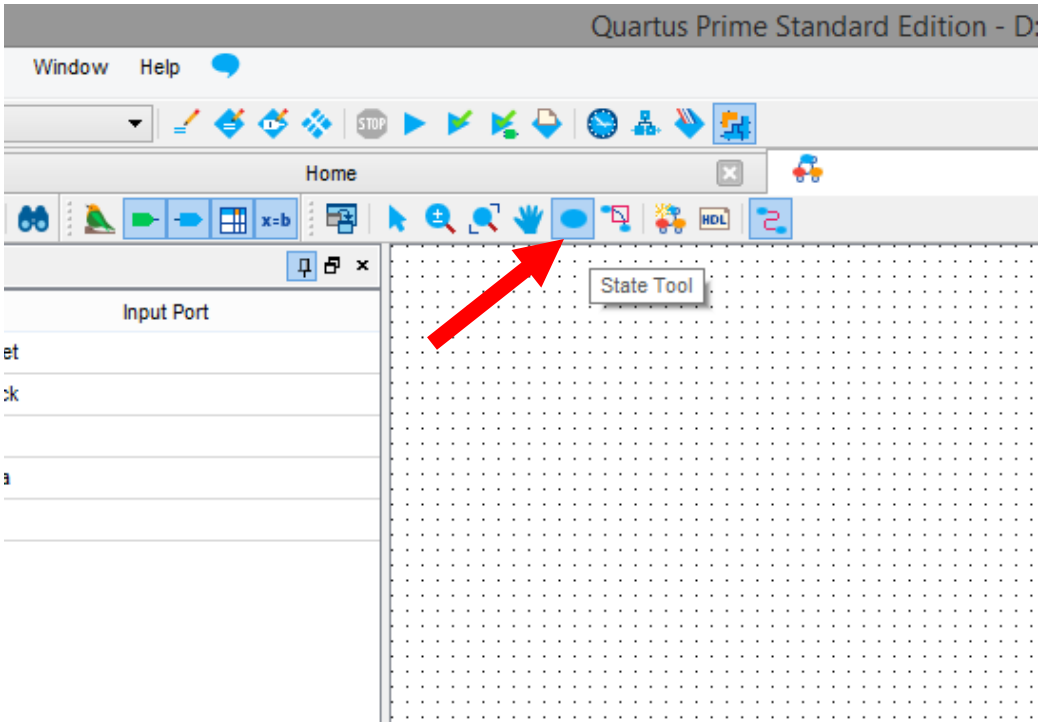


*Рисунок 5. Вот этой кнопкой можно вызвать инструмент, добавляющий состояния автомату*

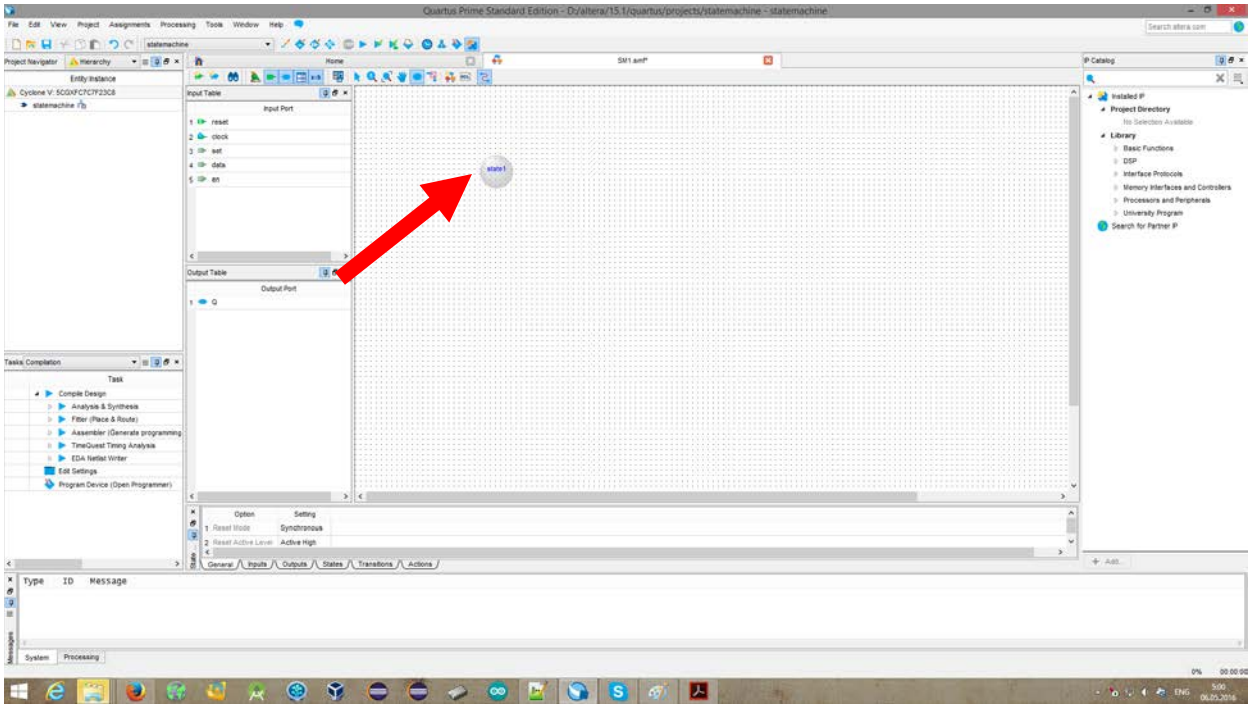

*Рисунок 6. Ваш курсор превращается в кружочек с надписью и при нажатии на свободное место, вы добавите туда состояние.*

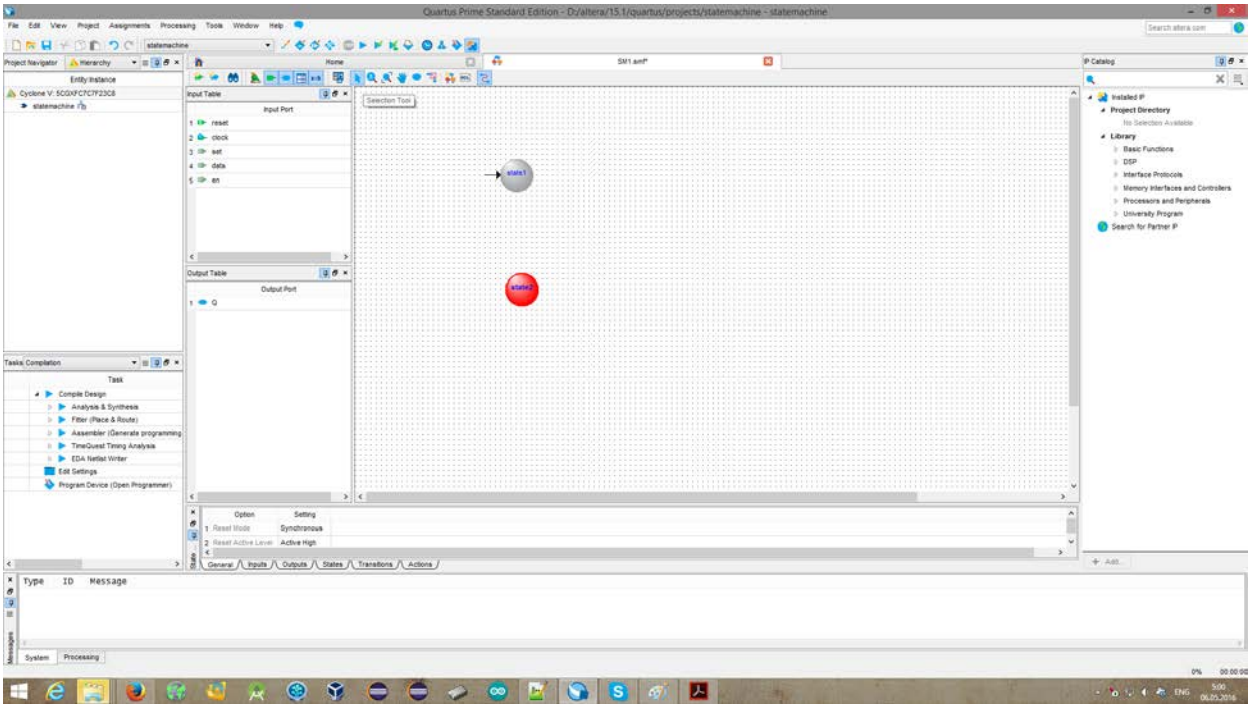

*Рисунок 7. Добавляем два состояния (в вашей задаче может потребоваться существенно больше). Тут видно состояние, в которое указывает стрелочка из неоткуда – это состояние сброса. Независимо от Вашей логики, при подаче на сигнал **reset** единицы автомат будет сброшен именно в это состояние. (Учитывайте это в логике переходов)*

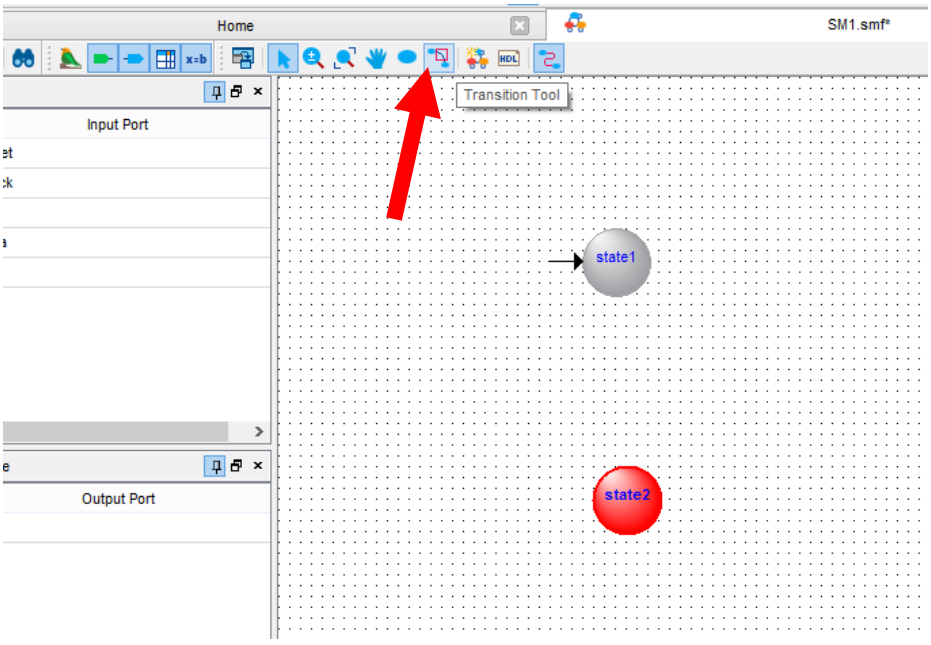


*Рисунок 8. Этот инструмент позволяет создавать переходы между состояниями.*

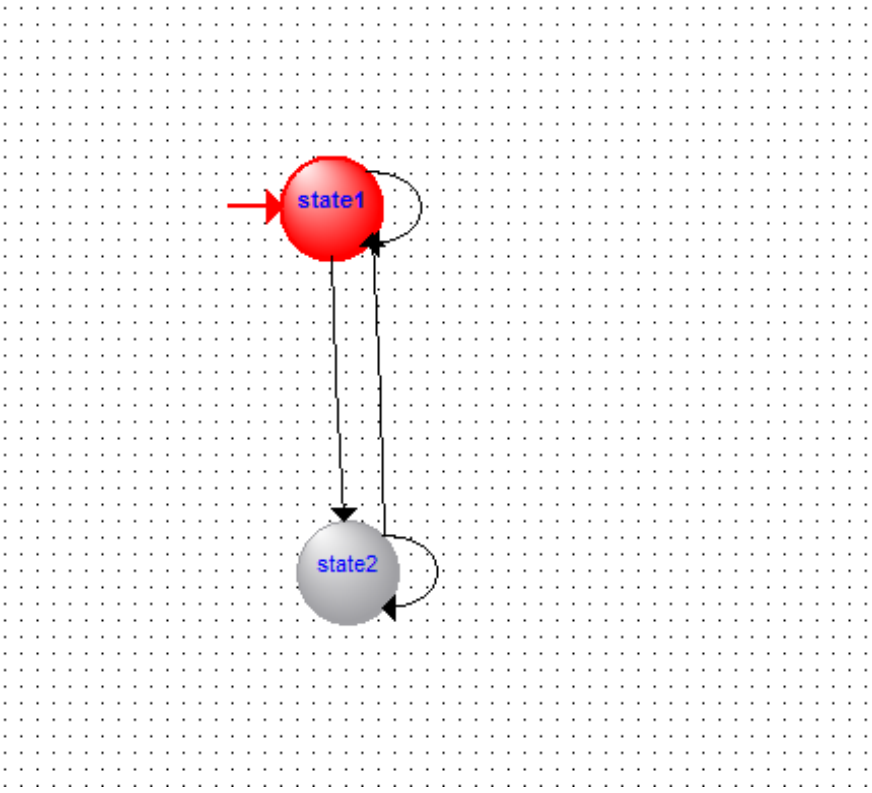


*Рисунок 9. По логике работы триггера добавляем четыре перехода. Два из них будут переводить триггер в противоположное состояние, а ещё два будут оставлять всё как есть.*

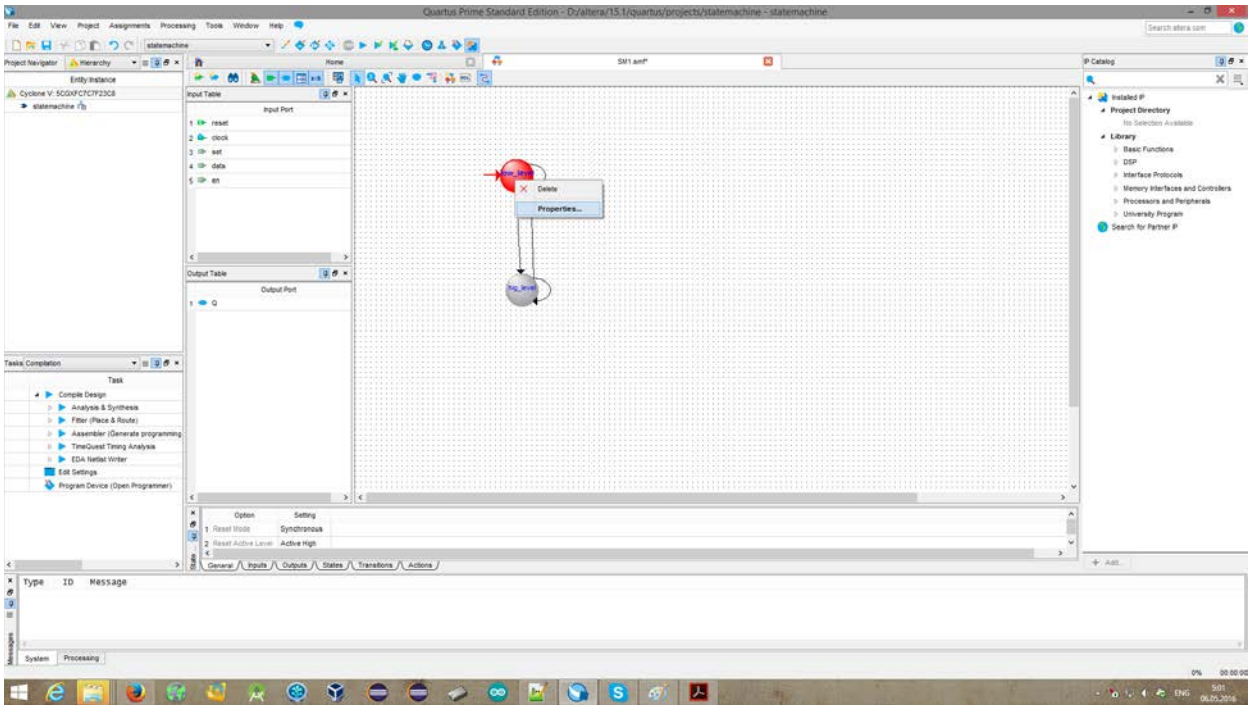


*Рисунок 10. Если нажать на состояние правой кнопкой мыши, то появляется контекстное меню. Нажав кнопку Properties, мы попадём в окно свойств состояния.*

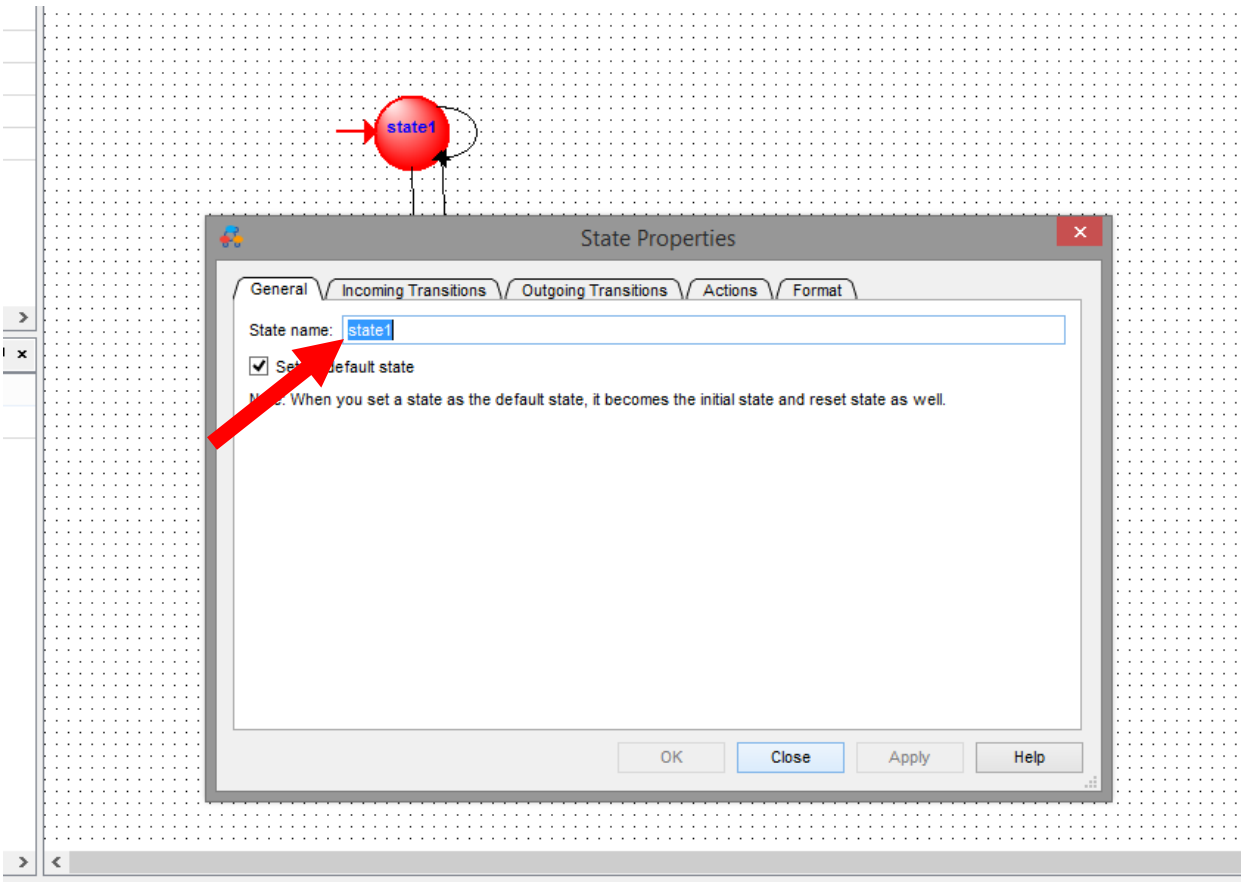


*Рисунок 11. Окно свойств состояния. Тут можно его переименовать, например, а, так же, сделать много ещё чего интересного.*

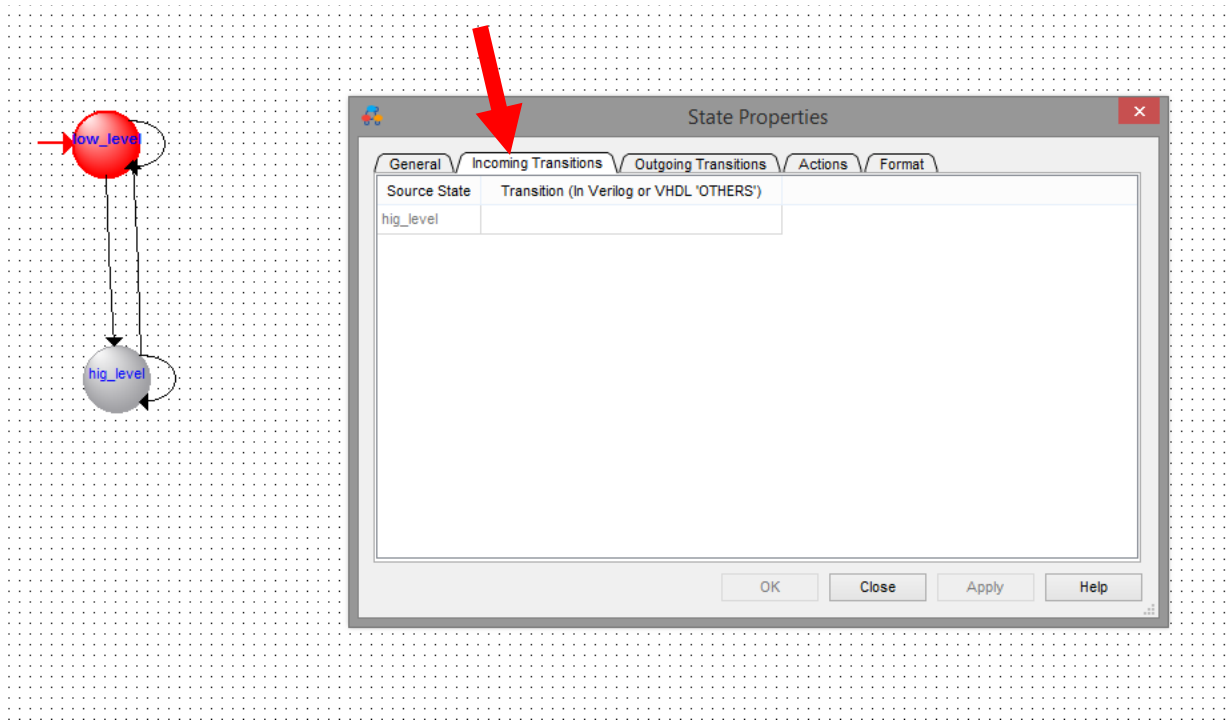


*Рисунок 12. Это вкладка, на которой отображены переходы, ведущие в данное состояние. В колонке **Transition** можно указывать условие перехода.*

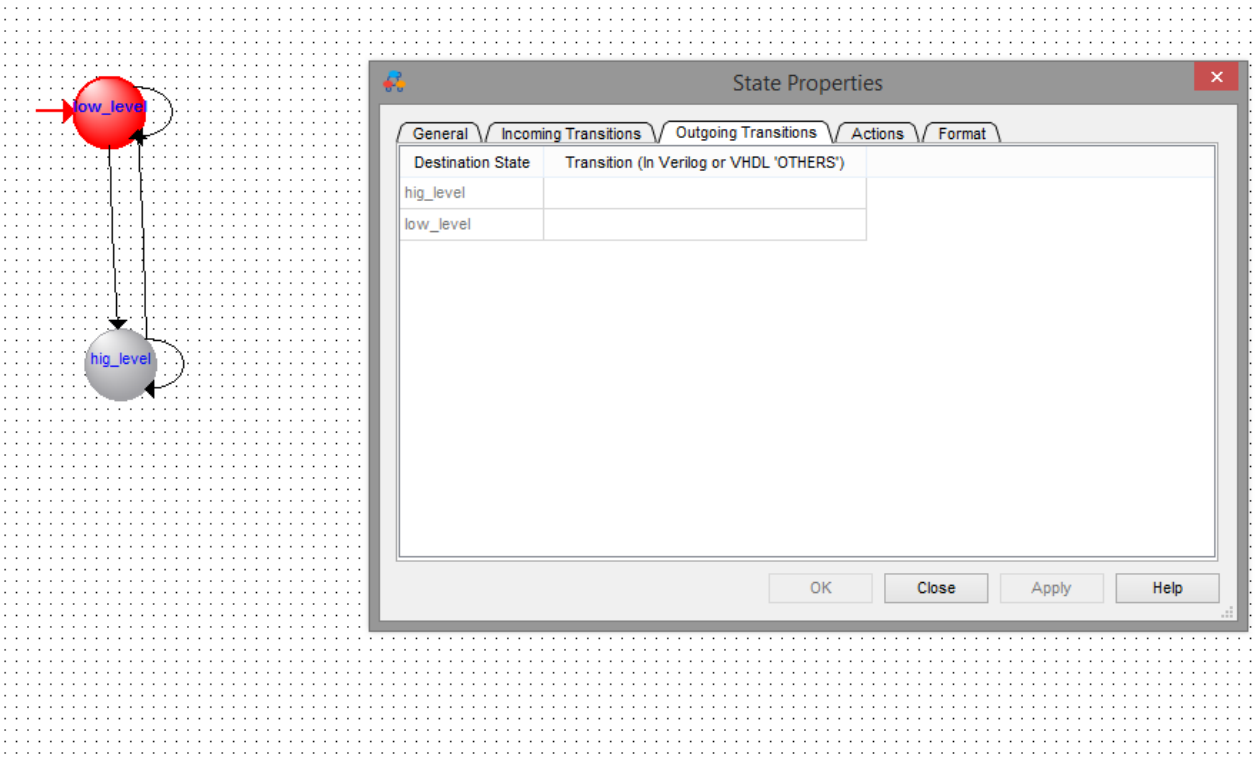


*Рисунок 13. Это вкладка с переходами, ведущими из данного состояния. И тут так же можно указывать условия выполнения переходов*

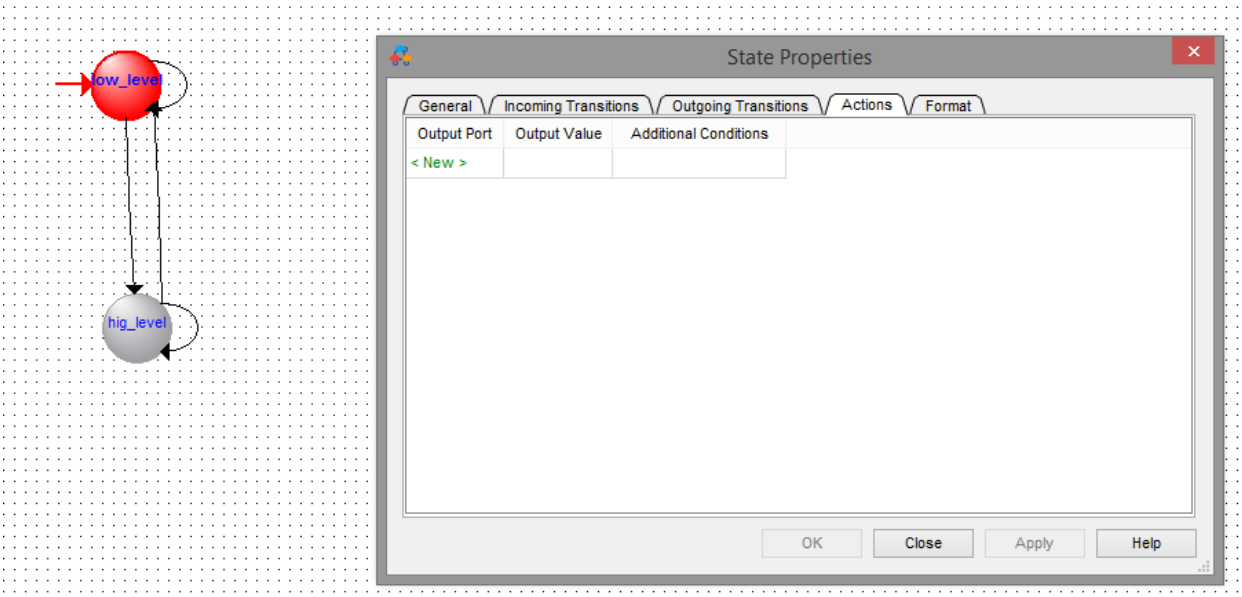


*Рисунок 14. Вкладка Action показывает список действий, а точнее значений сигналов, которые Вы будете выдавать на порт в данном состоянии, в зависимости от определённых условий*

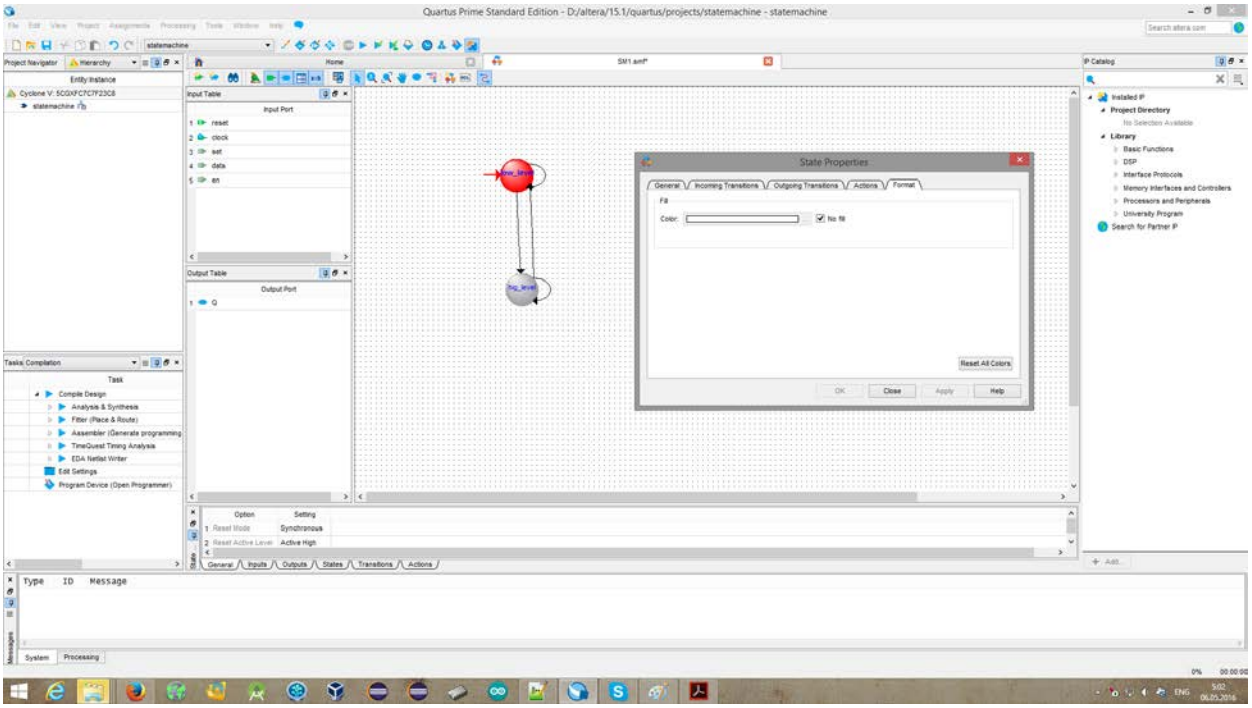


*Рисунок 15. На последней вкладке можно поменять цвет кружочка.*

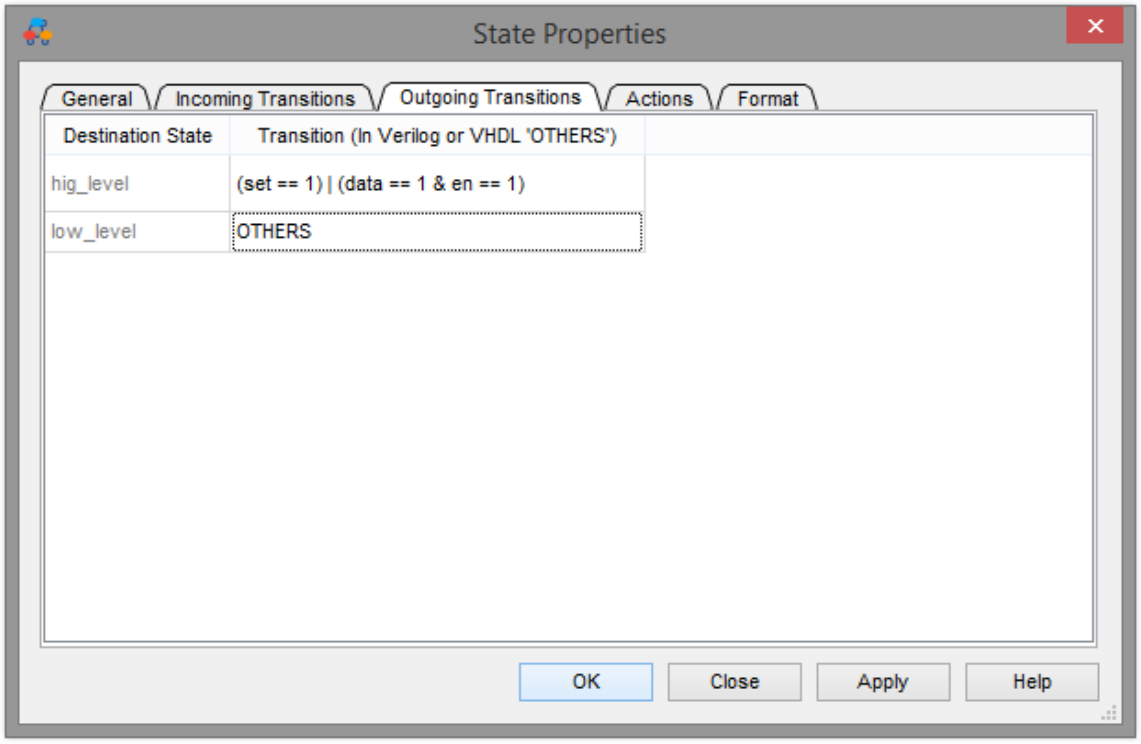


*Рисунок 16. Вот такие условия я использую для переходов из данного состояния (пишутся в стиле **Verilog**). Переход, который должен сработать лишь тогда, когда не сработали остальные, помечаем **OTHERS**.*

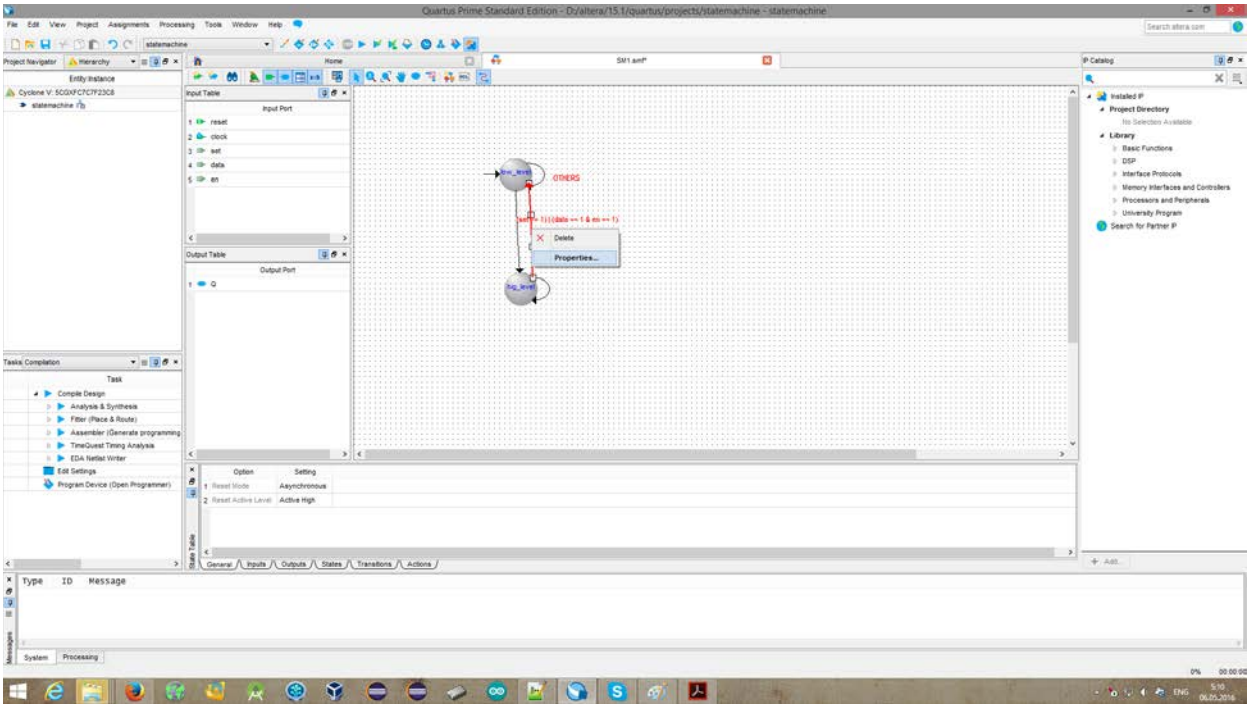


*Рисунок 17. Если нажать на переход правой кнопкой мыши, то появляется контекстное меню. Нажав кнопку Properties, мы попадём в окно свойств перехода.*

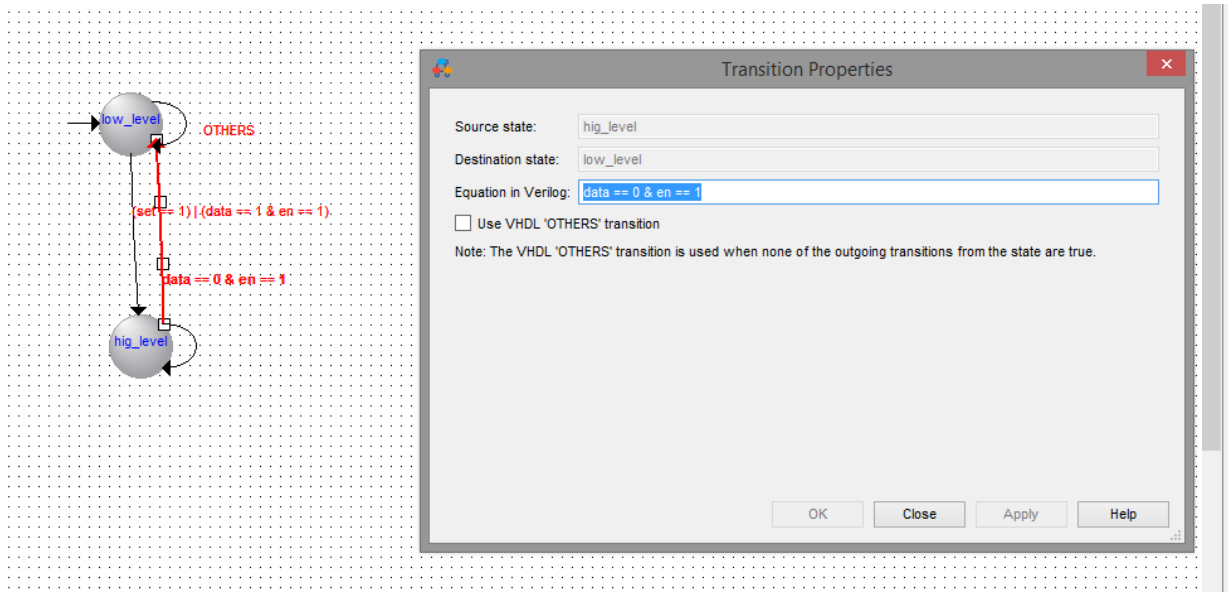


*Рисунок 18. Тут так же можно задавать условие для перехода.*

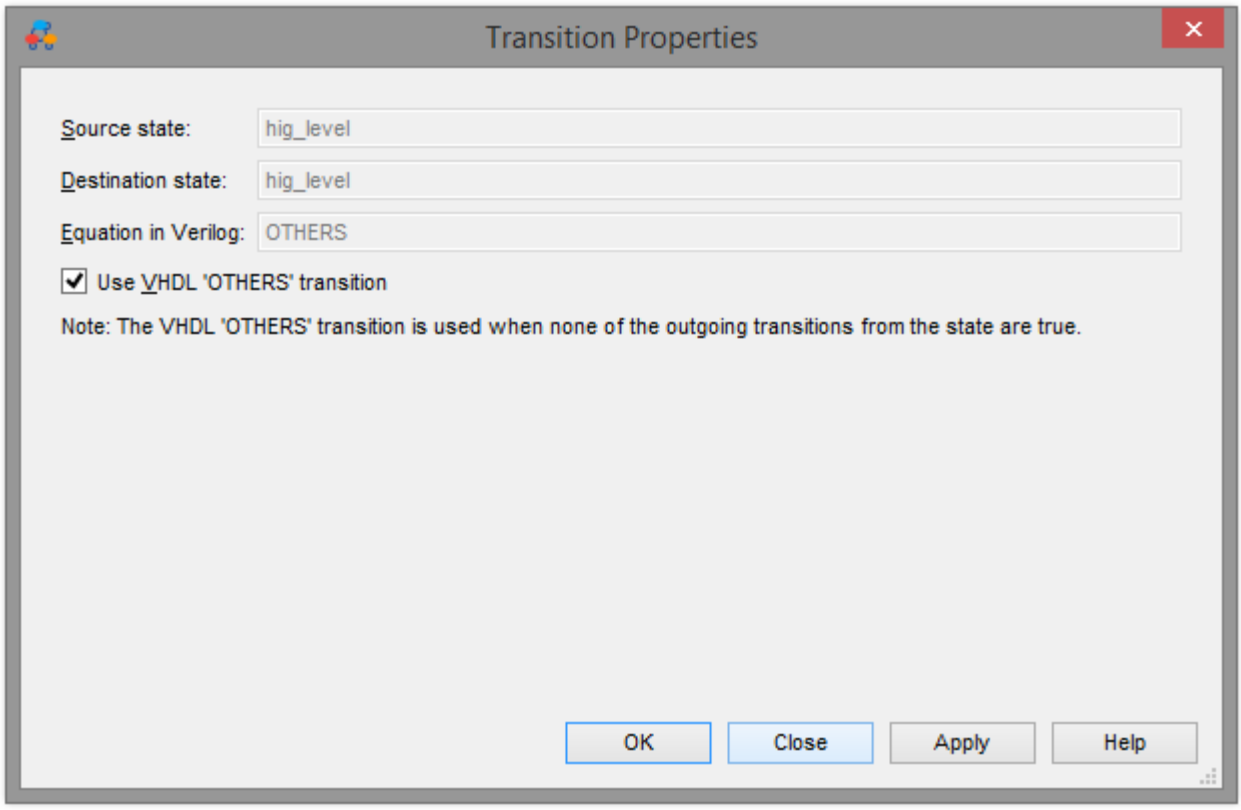


*Рисунок 19*

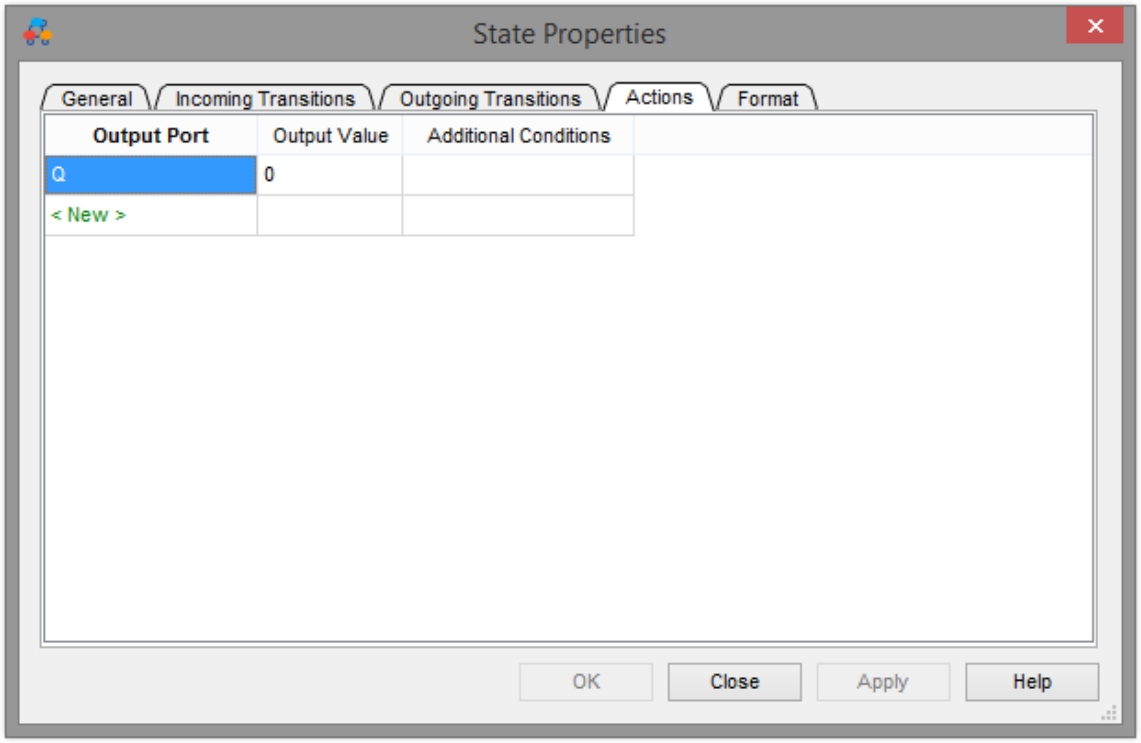


*Рисунок 20. Далее в Action обоих состояний прописываем соответствующее значение выхода.*

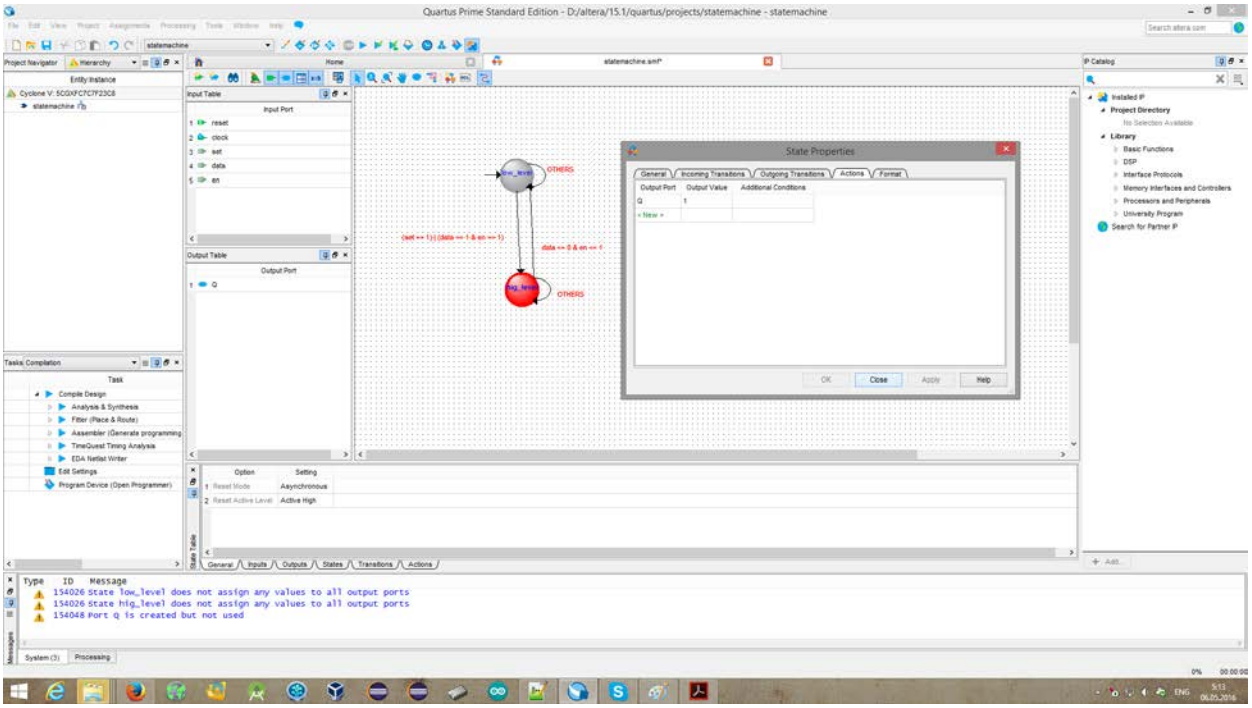


*Рисунок 21*

Во вкладках низу, так же, можно найти много интересного

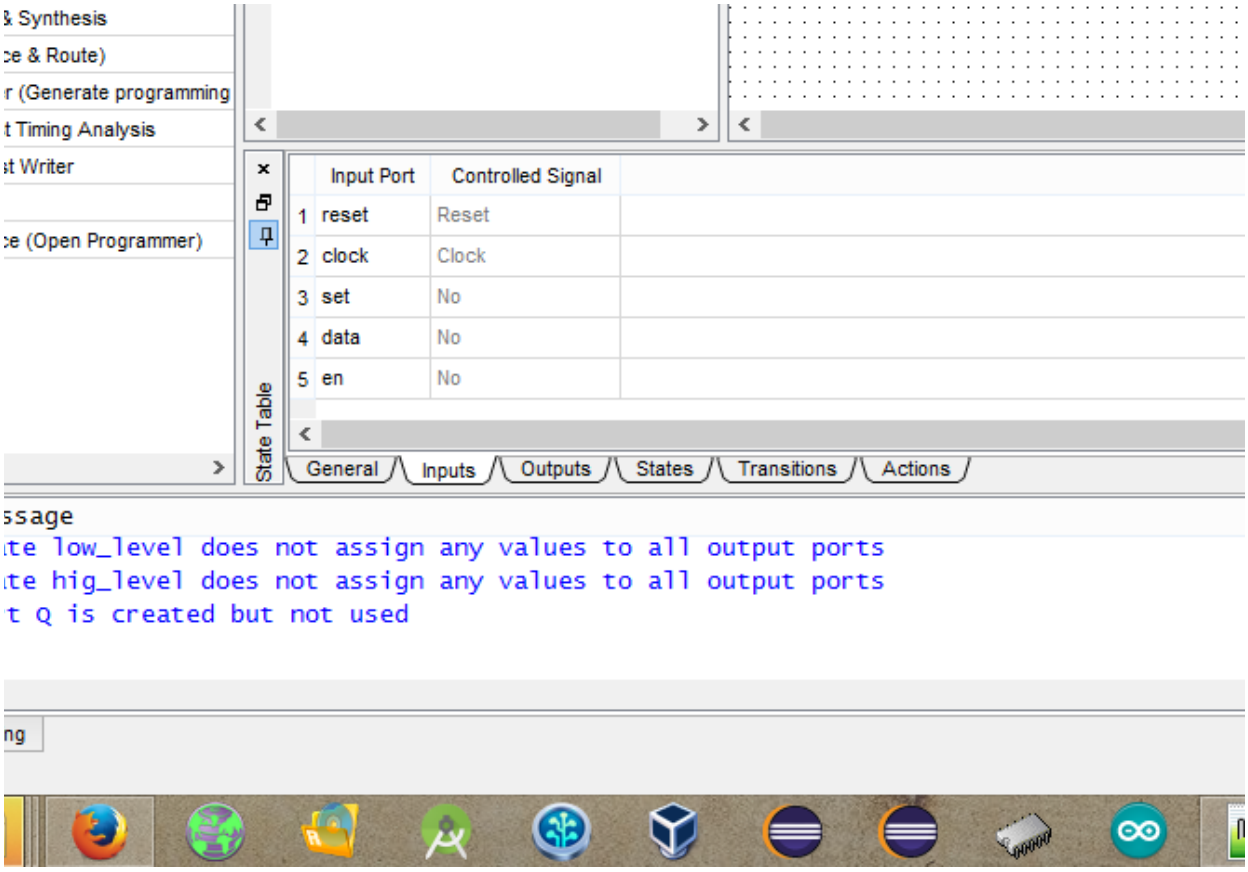


*Рисунок 22. Входные порты*

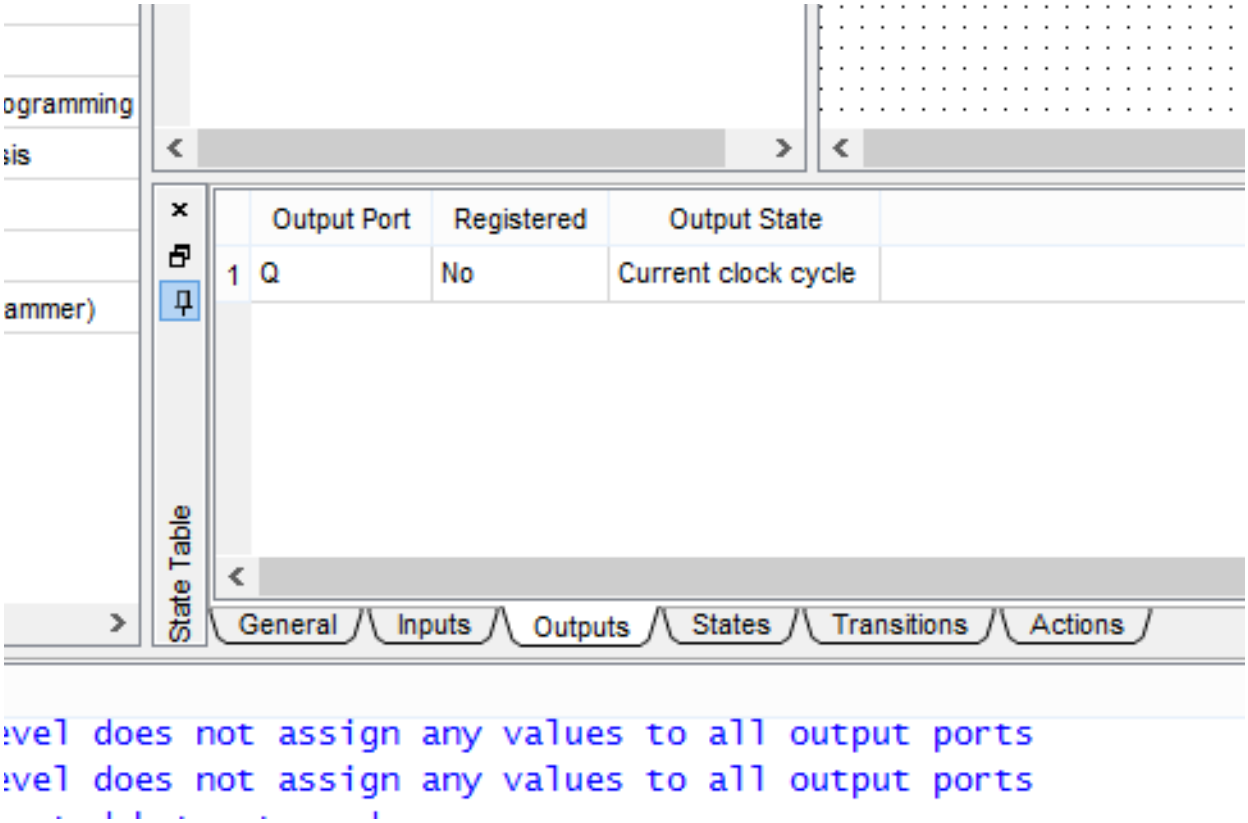


*Рисунок 23. Выходные порты*

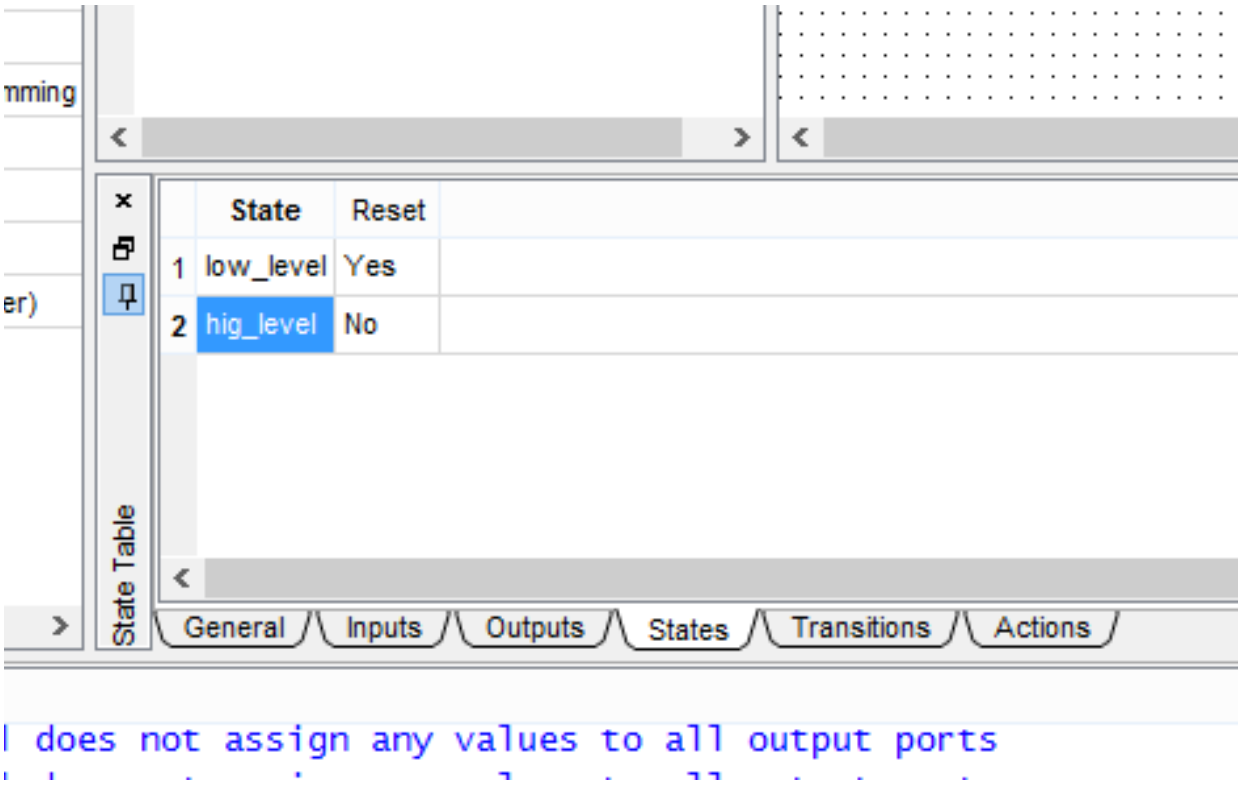


*Рисунок 24. Все состояния*

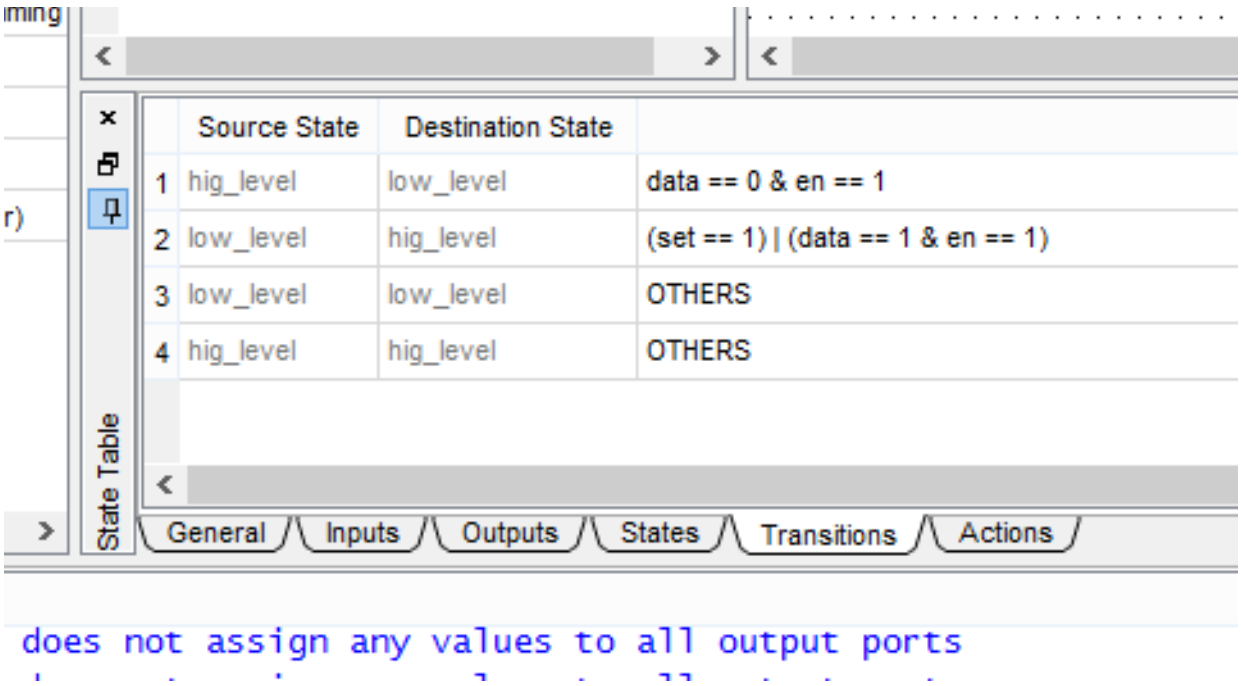


*Рисунок 25. Все переходы.*

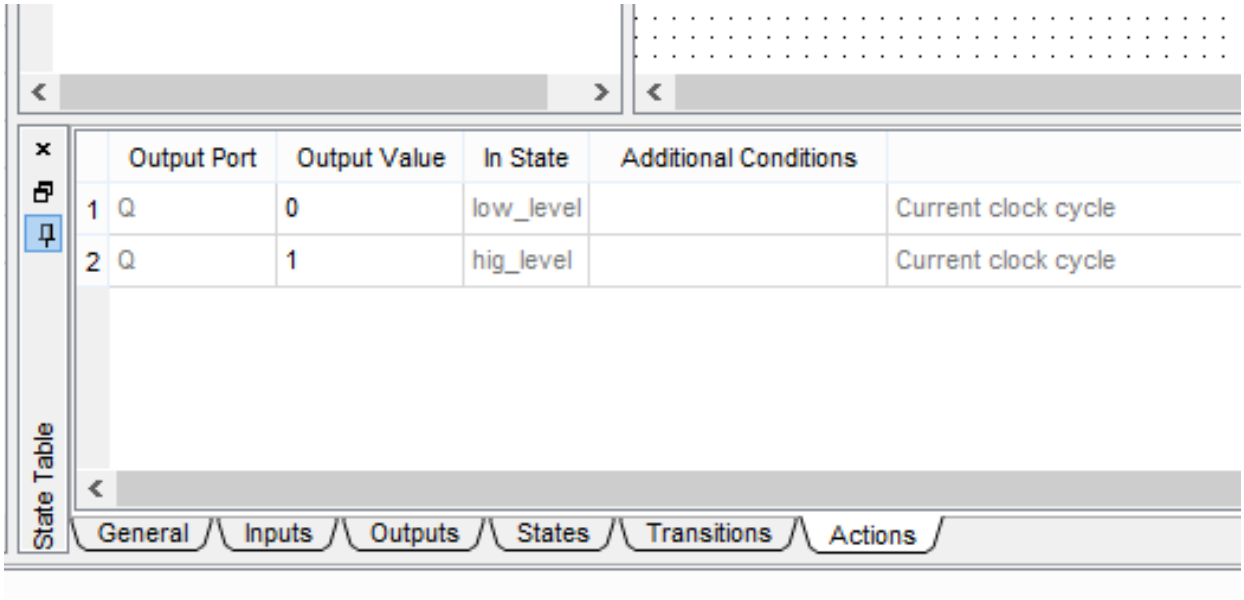


*Рисунок 26. Все Action's*

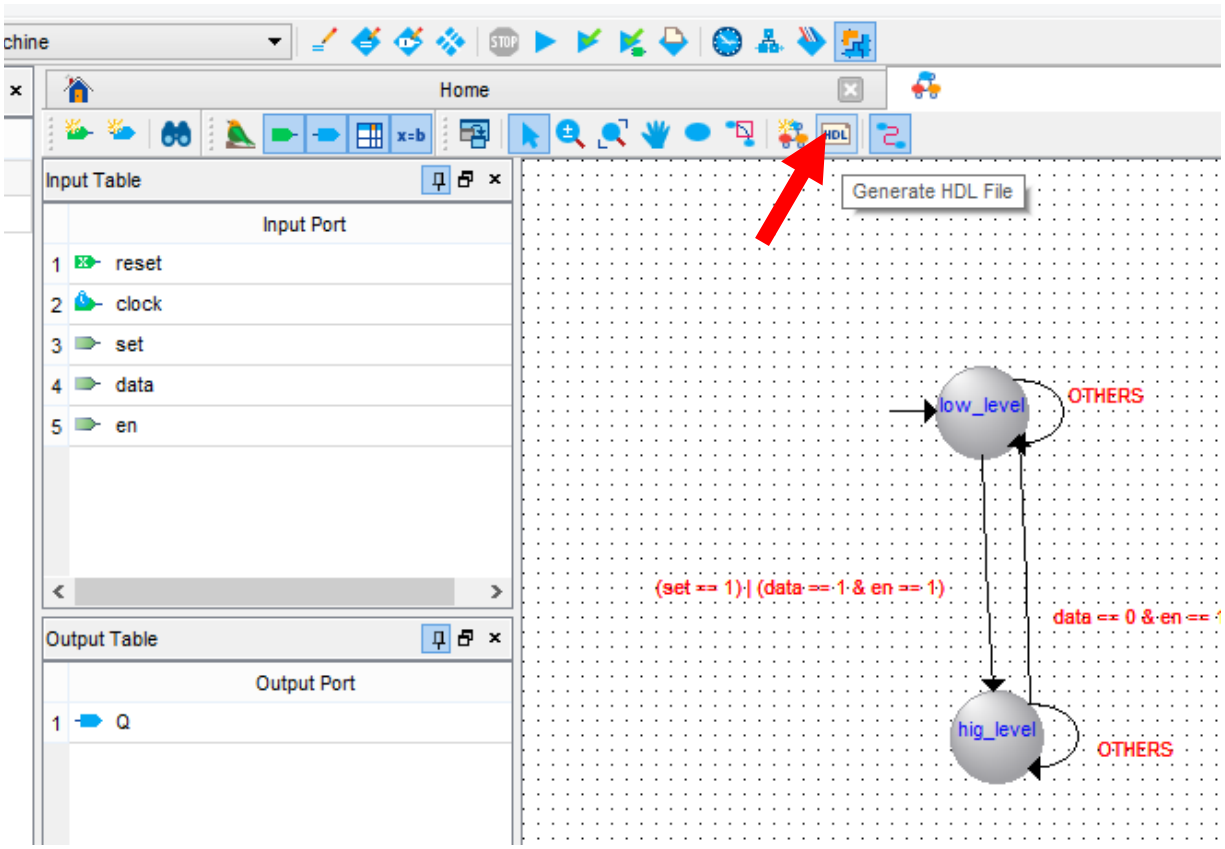


*Рисунок 27. Нажатием на эту кнопку можно вызвать диалог экспорты автомата в HDL код.*

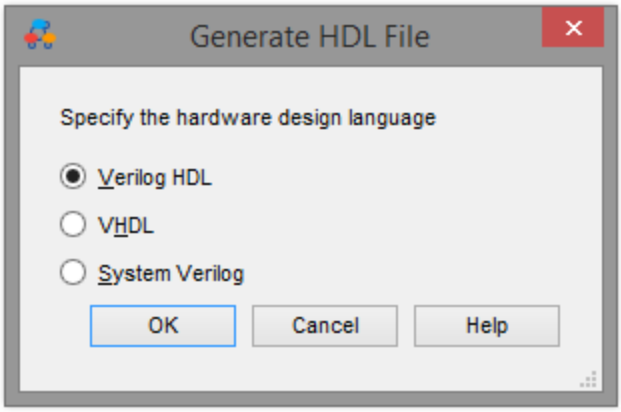


*Рисунок 28. Выбор языка для экспорта.*

Резултата на Verilog

```
// Copyright (C) 1991-2015 Altera Corporation. All rights reserved.
// Your use of Altera Corporation's design tools, logic functions
// and other software and tools, and its AMPP partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Altera Program License
// Subscription Agreement, the Altera Quartus Prime License Agreement,
// the Altera MegaCore Function License Agreement, or other
// applicable license agreement, including, without limitation,
// that your use is for the sole purpose of programming logic
// devices manufactured by Altera and sold by Altera or its
// authorized distributors. Please refer to the applicable
// agreement for further details.

// Generated by Quartus Prime Version 15.1.0 Build 185 10/21/2015 SJ Standard Edition
// Created on Fri May 06 05:15:40 2016

// synthesis message_off 10175

`timescale 1ns/1ns

module statemachine (
    reset,clock,set,data,en,
    Q);

    input reset;
    input clock;
    input set;
    input data;
    input en;
    tri0 reset;
    tri0 set;
    tri0 data;
    tri0 en;
    output Q;
    reg Q;
    reg [1:0] fstate;
    reg [1:0] reg_fstate;
    parameter low_level=0,hig_level=1;

    always @(posedge clock or posedge reset)
    begin
        if (reset) begin
            fstate <= low_level;
        end
        else begin
            fstate <= reg_fstate;
        end
    end

    always @(fstate or set or data or en)
    begin
        Q <= 1'b0;
        case (fstate)
            low_level: begin
                if (((set == 1'b1) | ((data == 1'b1) & (en == 1'b1))))
                    reg_fstate <= hig_level;
                else
                    reg_fstate <= low_level;
```

```
            Q <= 1'b0;
          end
          hig_level:  begin
            if (((data  == 1'b0)  & (en == 1'b1)))
              reg_fstate <= low_level;
            else
              reg_fstate <= hig_level;

            Q <= 1'b1;
          end
          default: begin
            Q <= 1'bx;
            $display ("Reach undefined  state");
          end
        endcase
    end
endmodule // statemachine
```

Резултат на VHDL

```
-- Copyright (C) 1991-2015 Altera Corporation. All rights reserved.
-- Your use of Altera Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Altera Program License
-- Subscription Agreement, the Altera Quartus Prime License Agreement,
-- the Altera MegaCore Function License Agreement, or other
-- applicable license agreement, including, without limitation,
-- that your use is for the sole purpose of programming logic
-- devices manufactured by Altera and sold by Altera or its
-- authorized distributors. Please refer to the applicable
-- agreement for further details.

-- Generated by Quartus Prime Version 15.1.0 Build 185 10/21/2015 SJ Standard Edition
-- Created on Fri May 06 05:16:24 2016

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY statemachine IS
   PORT (
      reset : IN STD_LOGIC  := '0';
      clock : IN STD_LOGIC;
      set : IN STD_LOGIC  := '0';
      data : IN STD_LOGIC  := '0';
      en : IN STD_LOGIC  := '0';
      Q : OUT STD_LOGIC
   );
END statemachine;

ARCHITECTURE BEHAVIOR OF statemachine IS
   TYPE type_fstate IS (low_level,hig_level);
   SIGNAL fstate : type_fstate;
   SIGNAL reg_fstate : type_fstate;
BEGIN
   PROCESS (clock,reset,reg_fstate)
   BEGIN
      IF (reset='1') THEN
         fstate <= low_level;
      ELSIF (clock='1' AND clock'event) THEN
         fstate <= reg_fstate;
      END IF;
   END PROCESS;

   PROCESS (fstate,set,data,en)
   BEGIN
      Q <= '0';
      CASE fstate IS
         WHEN low_level =>
            IF (((set = '1') OR ((data = '1') AND (en = '1')))) THEN
               reg_fstate <= hig_level;
            ELSE
               reg_fstate <= low_level;
            END IF;

            Q <= '0';
         WHEN hig_level =>
            IF (((data = '0') AND (en = '1'))) THEN
               reg_fstate <= low_level;
```

```
            ELSE
               reg_fstate <= hig_level;
            END IF;

            Q <= '1';
         WHEN OTHERS =>
            Q <= 'X';
            report "Reach undefined state";
      END CASE;
   END PROCESS;
END BEHAVIOR;
```