Продолжение. Начало в № 3`2008

Краткий курс HDL. Часть 8. Моделирование в ModelSim SE

Иосиф КАРШЕНБОЙМ
iosifk@narod.ru

Как добраться до ModelSim?

Практически все современные программные инструменты, которые поставляют фирмы — производители микросхем, содержат в своем составе средства для симуляции проектов. То же можно сказать и о компаниях — разработчиках программного обеспечения. Их программные инструменты также содержат в своем составе средства для симуляции проектов. Часть этих программных продуктов имеет собственные встроенные симуляторы, другие программы — нет. Но практически все эти программы имеют возможность подключать внешние симуляторы. Наиболее часто для отладки проектов используют симулятор ModelSim, который стал фактически промышленным стандартом. Более полную информацию о программе можно получить на сайте фирмы Mentor Graphic [1] или на сайте фирмы «Мегратек».

Как добраться до ModelSim в Quartus?

Для того чтобы получить информацию по применению программы ModelSim совместно с программой Quartus, обратитесь к [2].

Как добраться до ModelSim в ISE?

Для того чтобы получить информацию по применению программы ModelSim совместно с программой ISE, обратитесь к [3].

Как добраться до ModelSim в Actel Libero IDE?

Перед тем как непосредственно описывать работу с симулятором ModelSim, необходимо сделать небольшое пояснение: какие действия надо выполнить в программе Actel Libero IDE для начала симуляции проекта.

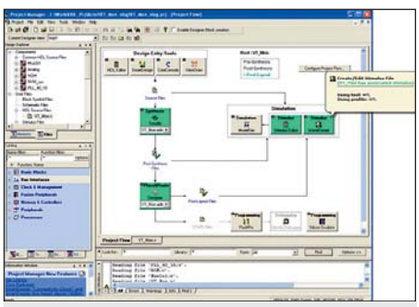

Рис. 1. Создание файла тестбенча

Для запуска симулятора ModelSim предварительно должен быть создан файл тестбенча. Его можно сделать в текстовом редакторе, добавив в каталог Stimulus в окне списка файлов проекта или нажав кнопку *WaveFormer* в окне на закладке ***Project Flow*** (рис. 1).

Файл, созданный в окне ***WaveFormer***, будет автоматически добавлен в список файлов для симуляции.

Кнопка *Stimulus Editor* позволяет добавлять и удалять файлы для тестбенча (рис. 2).

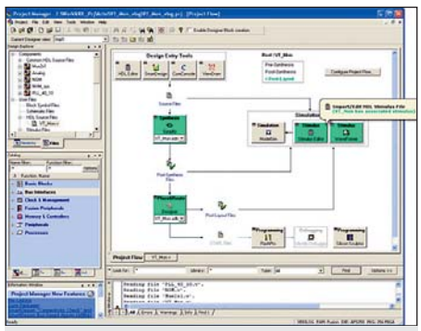

Рис. 2. Кнопка Stimulus Editor

Для добавления нового файла в список файлов для тестбенча нажмите кнопку *Stimulus Editor* и в открывшемся окне добавьте файл.

Для изменения списка файлов тестбенча нажмите мышкой на стрелку в левом верхнем углу кнопки *Stimulus Editor* и в раскрывшемся списке выберите пункт Organize Stimulus. В открывшемся окне (рис. 3) добавьте или удалите файлы для симуляции.

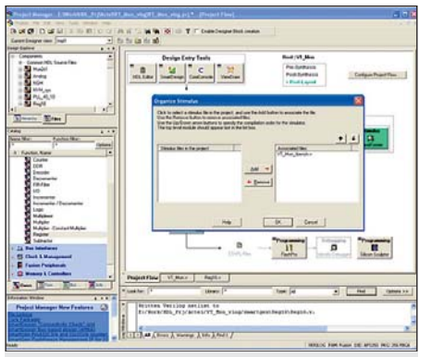

Рис. 3. Окно Organize Stimulus

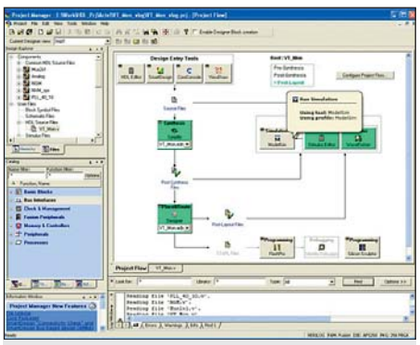

Рис. 4. Для запуска симуляции необходимо нажать кнопку ModelSim

Для запуска симуляции нажмите кнопку ModelSim (рис. 4).

Программа Actel Libero IDE позволяет симулировать исходный проект (проекты) после синтеза, размещения и трассировки кристалла. Для выбора нажмите на стрелку в левом верхнем углу на кнопке *ModelSim* и в раскрывшемся списке выберите режим симуляции (рис. 5).

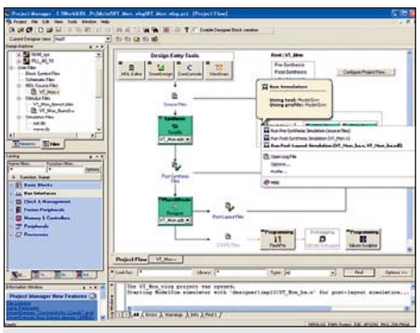

Рис. 5. Выбор режима симуляции

Создание Verilog-модели цифровой системы

В этом разделе мы начнем проверять модели в симуляторе. В примере 1 показан файл модуля, описывающий таймер. Таймер имеет разрядность, задаваемую параметром Width, и по умолчанию она равна 4. На таймер поступают обычные сигналы: Clk — сигнал системной синхрочастоты и Reset — сигнал асинхронного сброса. Таймер загружает-

ся данными, поступающими по шине Data при наличии разрешающего сигнала Load. Выходные сигналы — Out — это шина счетчика и сигнал Time, показывающий окончание счета и, следовательно, то, что выдержка времени завершена.

```
//////////////////////////////////////////////////
// Name File    : Timer4.v                      //
// Autor        : Iosif Karshenboim             //
// Company      :                               //
// Description  : Timer, Width=4 bits           //
// Start design : 16.10.2003                    //
// Last revision : 16.10.2003                   //
//////////////////////////////////////////////////

`timescale 1ns / 10 ps
module Timer4
// These parameters can be overridden
#( parameter Width=4)
   ( Out,
     Time,
     Load,
     Data,
     Clk,
     Reset);

output [Width-1:0]      Out;
output                  Time;

input [Width-1:0]       Data;
input                   Load;
input                   Clk;
input                   Reset;

reg [Width-1:0]         Out;
Reg                     Time;
//---------------------------------
   always @(posedge Clk or posedge Reset)
      if (Reset)
         Out = 0;
      else
      if (Load & Time)
         Out = Data;
      else
      if (!Time)
         Out = Out-1; // decrement(count)
      else
         Out = Out; // hold
//---------------------------------
   always @(posedge Clk or posedge Reset)
      if (Reset)
         Time =1'b1;
      else
      if (Out[Width-1:1]==0)// time OK
         Time =1'b1;
      else
         Time = 1'b0;
//---------------------------------

endmodule
```

Пример 1. Файл модуля, содержащий описание таймера

В примере 2 приведен файл простейшего тестбенча для проверки модуля Timer4. Однако проверка данного модуля — это только первый и довольно простой шаг, поэтому мы сразу перейдем к следующему файлу. В примере 3 приведен файл **STATMACH.v**, в который установлен Timer4. Кроме того, в этом же модуле дано описание статического автомата. Задержка, вырабатываемая таймером, — фиксированная и задается в строке:

```
reg [4:0] tmr_delay = 5;
```

Автомат выполняет переход только по получении сигнала от таймера:

```
if(tmr_ready)
state = next_state;
```

Пример 3 показывает, как можно выполнить программируемую задержку для статических автоматов. Обратите внимание на то, что в файле **STATMACH.v** нет строки:

```
`timescale 1ns / 10 ps
```

Отсутствие данной директивы приведет в дальнейшем к тому, что симулятор выдаст предупреждение об этом.

```
`timescale 1ns / 10 ps
module test_timer;
parameter TWidth=4;

reg Clk, Reset, Load;
reg [TWidth-1:0] Data;
wire [TWidth-1:0] Out;

Timer4 #(.Width (TWidth))
   dut (Out,
        Time,
        Load,
        Data,
        Clk,
        Reset);

initial // Clock generator
   begin
        Clk = 0;
        #10 forever #10 Clk = !Clk;
   end

initial // Test stimulus
   begin
        Reset = 0;
        #10 Reset = 1;
        #40 Reset = 0;

        Data=4'b0011;
        Load = 0;
        #90 Load = 1;
        #200 Load = 0;

        #50000 $stop;
   end

initial
        $monitor($stime, Reset,Data, Clk,Time,Load, Out);

endmodule
```

Пример 2. Файл простейшего тестбенча для проверки модуля Timer4

```
//////////////////////////////////////////////////////////////////
// Name File    : STATMACH.v                                    //
// Autor        : Iosif Karshenboim                             //
// Company      :                                               //
// Description  : Timer, Width=4 bits and FSM                   //
// Start design : 16.10.2003                                    //
// Last revision : 16.10.2003                                   //
//////////////////////////////////////////////////////////////////

module statmach
          (clk,
           reset,
           in,
           out);
     input clk;
     input [3:0] in;
     input reset;
     output [4:0] out;

     reg [4:0] out;
     reg [3:0] state, next_state;
     wire tmr_ready;
     reg tmr_load = 1;
     reg [4:0] tmr_delay = 5;

     parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3;

// FSM register
   always @(posedge clk or posedge reset)
   begin: statreg
        if (reset)
           state = s0;
        else
        if (tmr_ready)
           state = next_state;
   end

// FSM conbinational block
   always @(state)
   begin: FSM
        case (state)
           s0: next_state = s1;
           s1: next_state = s2;
           s2: next_state = s3;
           s3: next_state = s0;
        endcase
   end

always @(state)
   begin: outputs
        case (state)
           s0: out = 5'b00011;
           s1: out = 5'b00110;
           s2: out = 5'b01100;
           s3: out = 5'b11000;
           default: out = 5'b11111;
        endcase
   end // outdata

Timer4 #(5) TIMER // These parameters can be overridden
   (.Out (),
    .Time (tmr_ready),
    .Load (tmr_load),
    .Data (tmr_delay),
    .Clk (clk),
    .Reset (reset));

endmodule
```

Пример 3. Файл STATMACH.v, в который установлен Timer4, и в этом же модуле дано описание статического автомата

Теперь переходим к файлу тестбенча для проверки модуля **STATMACH.v**.

В примере 4 приведен файл **tstatmach.v**, в который установлен модуль **STATMACH.v**. В примере 5 приведен файл паттернов для тестбенча по проверке модуля **STATMACH.v**. Основная цель данного примера — показать то, как производится чтение данных из файла паттерна и как эти данные можно использовать при симуляции. В файле паттернов в каждой строке считываются различные по формату данные.

Работа с файлами и примеры отладки

Итак, в примере 4 приведен файл tstatmach.v. А в примере 5 приведен файл паттернов для тестбенча по проверке модуля tstatmach.v. В данном примере тестбенча мы будем производить чтение данных из файла паттерна, а полученные данные — использовать при симуляции. В файле паттернов в каждой строке считываются различные по формату данные — это **time**, **bin**, **dec** и **hex**. В приведенном примере из этих данных в тестбенче будут использоваться только первые две переменные — **time** и **bin**, а остальные переменные — **dec** и **hex** — будут считываться, но не будут использоваться в тестбенче.

Тестбенч состоит из трех фрагментов, начинающихся с initial.

Первый фрагмент формирует синхрочастоту, подаваемую на DUT. Это можно сделать при помощи выражения:

```
#10 forever #10 clk = !clk;
```

Второй фрагмент формирует сигнал «сброс», подаваемый на DUT. Это делается при помощи трех выражений:

```
      reset = 0;
#10 reset = 1;
#30 reset = 0;
```

Третий фрагмент формирует все остальные сигналы, подаваемые на DUT. Это делается в блоке file_block.

Файл, определенный как INPUT_FILE_NAME “vectors.pat”, открывается только для чтения. Если файл открылся успешно, то в цикле **while (c != `EOF)** производится считывание строк файла паттерна до тех пор, пока не будет обнаружен конец файла. При считывании строк паттерна выполняется проверка на то, что данная строка не заремаркирована. Сначала производится считывание первого символа строки паттерна. Это происходит в строке файла **if (c == “/”)**, и, если это условие выполняется, то считывается вся строка паттерна: **r = $fgets(line, file)**, но эти данные не используются. Если же первый считанный символ не равен “/”, то он возвращается в строку. Далее из строки паттерна считывается значение времени. Проверяется, не обогнало ли текущее время моделирования время, читаемое из паттерна. Затем программа выполняет задержку, так, чтобы текущее время моделирования стало равным тому значению времени, которое мы прочли из паттерна. И после этого считываются все остальные данные из паттерна. Бинарные данные, считываемые из паттерна, имеют разрядность 4 бита. Но из этих 4 битов в тестбенче используется только младший бит. И его значение назначается регистру:

```
in = bin[0];
```

По окончании работы с паттерном файл закрывается. При изменении значений переменных командой **$display** на монитор выводятся значения этих переменных.

```
////////////////////////////////////////////////////
// Name File    : tstatmach.v                     //
// Autor        : Iosif Karshenboim               //
// Company      :                                 //
// Description  : Testbench                       //
// Start design : 16.10.2003                      //
// Last revision : 16.10.2003                     //
////////////////////////////////////////////////////

`timescale 1ns / 10 ps
`define EOF 32'hFFFF_FFFF
`define NULL 0
`define MAX_LINE_LENGTH 1000
`define INPUT_FILE_NAME “vectors.pat”

module test_statmach;

reg clk, reset, in;
reg [31:0] tmp_value;
wire out;

statmach
   dut (clk, in, reset, out);

initial // Clock generator
   begin
      clk = 0;
      #10 forever #10 clk = !clk;
   end

initial // Test stimulus
   begin
      reset = 0;
      #10 reset = 1;
      #30 reset = 0;
   end

integer file;
reg [31:0] c, r;
reg [3:0] bin;
reg [31:0] dec, hex;
real real_time;
reg [8*`MAX_LINE_LENGTH:0] line; /* Line of text read from file */

initial
      begin : file_block
   /*timeformat_task ::= $timeformat [(units_number,
                                       precision_number,
                                       suffix_string,
                                       minimum_field_width) ] ; */
   $timeformat(-9, 3, “ns”, 6);
   file = $fopen(`INPUT_FILE_NAME, “r”);
      if ((file != `EOF) & (file != 0))
         $display(“The input file was opened successfully.”);
      else
         begin
            $display(“ERROR: unable to open the input file.”);
            $finish;
         end

// if (file == `NULL) // If error opening file
//  disable file_block; // Just quit

         c = $fgetc(file);

      $display(“time bin decimal hex”);

      while (c != `EOF)
         begin
            /* Check the first character for comment */
            if (c == “/”)
               r = $fgets(line, file);
            else
               begin
               // Push the character back to the file then read the next time
               r = $ungetc(c, file);
               r = $fscanf(file, “%f:\n”, real_time);

               // Wait until the absolute time in the file, then read stimulus
               if ($realtime > real_time)
                  $display(“Error — absolute time in file is out of order — %t”, real_time);
                  else
                     #(real_time — $realtime)
                     r = $fscanf(file,” %b %d %h\n”,bin,dec,hex);
                     //---------------------------------------------
                     in = bin[0];
                     //---------------------------------------------
               end // if c else
               c = $fgetc(file);

         end // while not EOF
      $fclose(file);
$stop;

end // initial

//=================================================
// Display changes to the signals
//=================================================
always @(bin or dec or hex)
      $display(“%t %b %d %h”, $realtime, bin, dec, hex);

endmodule // read_pattern
```

Пример 4. Тестбенч — модуль test_statmach, файл tstatmach.v

```
// This is a pattern file
// time bin dec hex
0         : 0001 1 1
10        : 0001 1 1
20.0      : 0010 20 020
50.02     : 0111 5 FFF
62.345    : 0100 4 DEADBEEF
75.789    : 0XX1 2 ZzZzZzZz
200.0     : 0010 20 020
400.0     : 1010 25 F2A
```

Пример 5. Файл паттернов для тестбенча tstatmach.v

Моделирование тестбенча

Для моделирования будем использовать программу ModelSim, ставшую за последнее время промышленным стандартом. Первым шагом будет создание проекта. После запуска программы ModelSim SE открывается основное окно программы (рис. 6).

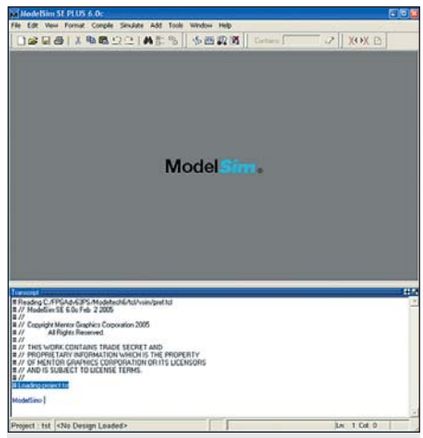

Рис. 6. Основное окно системы моделирования ModelSim SE

Для того чтобы начать моделирование, необходимо создать новый проект. Для этого в главном меню выбираем ***File→New→Project***. Появится окно ***Create Project***.

Для создания проекта выполним следующие шаги: нам необходимо указать имя проекта и путь к файлам проекта, название рабочей библиотеки:

1. В окне ***Project Name*** укажем имя проекта (**My_testbench**).
2. В окне ***Project Location*** укажем рабочую директорию (**E:/My_testbench**).
3. В окне ***Default Library Name*** оставим имя **work**.
4. Нажмем OK (рис. 7).

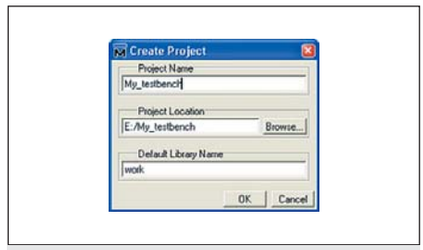

Рис. 7. Задание имени проекта, пути к файлам проекта и название рабочей библиотеки

Если пользователь программы захочет повторно создать проект с тем же именем, в той же рабочей директории, то программа выдаст предупреждение о том, что проект с этим именем уже существует. Если мы не хотим менять имя проекта, и старый проект нам не нужен, то в таком случае в этом окне необходимо указать «Да» (рис. 8).

Рис. 8. Подтверждение имени проекта

Если же файлов для проекта еще нет, то в открывшемся окне выбираем пункт **Create New File**. Откроется окно для ввода текстовых файлов. И далее создаем файлы проекта один за другим. Если файлы для данного проекта уже имеются, то в открывшемся окне ***Add Items to the Project*** выбираем **Add Existing File** (рис. 9).

Рис. 9. Добавление файлов в проект в окне Add Items to the Project

После выбора **Add Exiting File** появится окно ***Add file to Project*** (рис. 10).

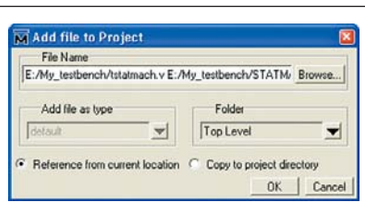

Рис. 10. Окно для выбора добавляемого в проект файла

Добавляем файлы в проект либо по очереди, либо это можно сделать одновременно, если отметить сразу несколько файлов. В окне ***Workspace*** поочередно появляются имена добавляемых файлов проекта (рис. 11). Файлы будут расположены в том порядке, в котором они были добавлены в проект. Но для работы компилятора необходимо указать иерархию файлов. Это можно сделать либо вручную, либо автоматически.

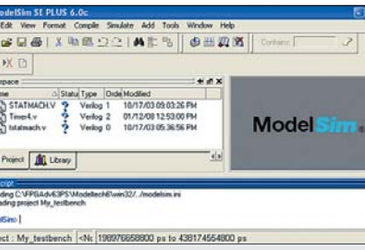

Рис. 11. Файлы в проект добавлены

Чтобы установить порядок компиляции файлов вручную, необходимо учесть, что компиляция осуществляется согласно иерархии описания проекта, и сначала на компиляцию должны поступать следующие файлы:

- описания модулей нижнего уровня (в примере это описание Timer4.V);
- затем описания модулей верхнего уровня, содержащие установленные модули, которые описаны в модулях нижнего уровня, и т. д. (в примере это описание **STATMASH.v**);
- последний файл — это тестбенч.

Выберем второй путь. Для этого в главном меню надо выбрать ***Compile→Compile Order***, далее появится окно ***Compile Order***.

Итак, выбираем Auto Generate и после этого нажимаем OK. Если файлы не содержат ошибок, то компилятор выдаст в нижнем окне ***Transcript*** соответствующие сообщения. В данном примере эти сообщения имеют вид, представленный на рис. 12.

```
# reading C:\FPGAdv63PS\Modeltech6\win32/../modelsim.ini
# Loading project My_testbench
# Compile of tstatmach.v was successful.
# Compile of STATMACH.V was successful.
# Compile of Timer4.v was successful.
# 3 compiles, 0 failed with no errors.
```

Рис. 12. Сообщения компилятора

В окне ***Workspace*** напротив имен файлов знак вопроса после компиляции заменяем на «галочку» (рис. 13).

Если программа содержит ошибку, то в нижнем окне ***Transcript*** появится сообщение об этом.

Двойной щелчок на сообщении об ошибке выдает строку Verilog-текста, в которой может быть ошибка (однако на самом деле ошибка может быть совершенно в другой строке — синтаксис Verilog сложный, и так же, как и в компиляторах языка Си, ошибка может быть в строке перед указанной строкой).

Выдача текста Verilog-файла для редактирования осуществляется после двойного щелчка по имени соответствующего файла.

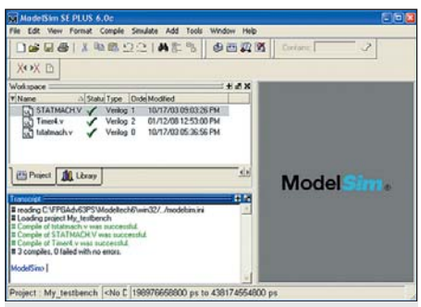

Рис. 13. Компиляция файлов проекта проведена

Далее переходим к заданию опций моделирования. Для этого в главном меню необходимо выбрать: ***Simulate→Runtime Options***, появится окно ***Runtime Options***. Задать:

- ***Default Radix→Symbolic*** (формат представления данных);
- ***Default Run→100 ns*** (время одного шага моделирования при пошаговом моделировании);
- ***Iteration Limit→5000*** (число событий моделирования, событие — изменение сигнала).

Остальное — по умолчанию, нажать OK (рис. 14).

Рис. 14. Задание опций моделирования в окне Runtime Options

Следующим шагом будет установка верхнего модуля проекта для симуляции в окне ***Start Simulation***. Выбрать в главном меню ***Simulate→Start Simulation***, появится окно ***Start Simulation*** (рис. 15).

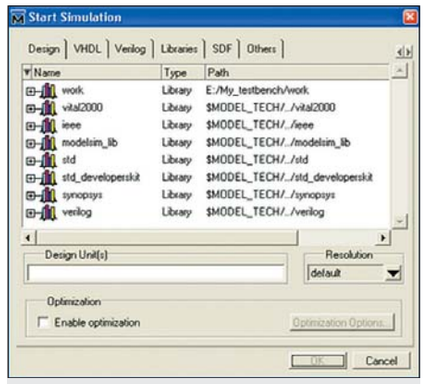

Рис. 15. Окно Start Simulation для указания верхнего модуля проекта

В окне ***Start Simulation*** необходимо выполнить следующие установки: раскрыть библиотеку work, выбрать (двумя щелчками) имя верхнего модуля проекта (test_statmach), остальное можно оставить по умолчанию. Нажать OK (рис. 16).

После выбора верхнего модуля проекта в окне ***Workspace*** появится закладка ***sim***, и внешний вид программы будет такой, как на рис. 17.

При загрузке скомпилированных файлов будут выданы следующие сообщения (рис. 18).

Первое из этих сообщений говорит о том, что один из файлов проекта не содержит директивы `timescale, а два других — о том, что

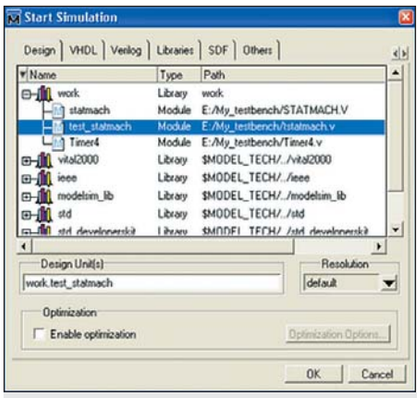

Рис. 16. Выбран модуль верхнего уровня

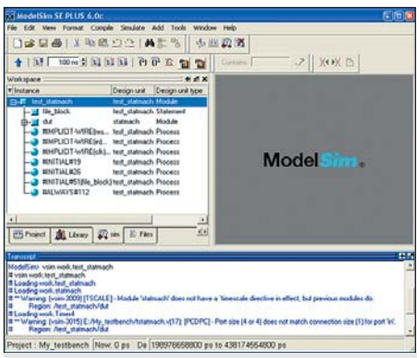

Рис. 17. Проект для моделирования прочитан без ошибок

некоторые из портов вывода одного из модулей задействованы лишь частично.

Следующим шагом будет открытие окна ***Wave***. Для этого в окне ***Workspace***, в закладке ***sim***, выделим строку с названием модуля верхнего уровня **test_statmach**. Теперь кликнем правой клавишей мышки по этой строке и в открывшемся меню выберем строку ***Add→Add to Wave***. При этом откроется окно ***Wave*** и в него будут загружены названия сигналов модуля **test_statmach**. Внешний вид этого окна программы показан на рис.19.

Выполним команду Run All. Это можно сделать, нажав кнопку . При выполнении программы симулятор выдаст сообщения (рис. 20).

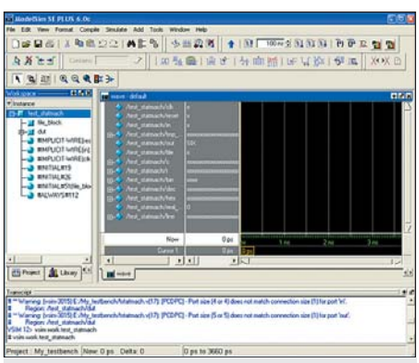

Рис. 19. Окно временных диаграмм

```
vsim work.test_statmach
# vsim work.test_statmach
# Loading work.test_statmach
# Loading work.statmach
# ** Warning: (vsim-3009) [TSCALE] — Module 'statmach' does not have a `timescale directive in effect, but previous modules do.
#    Region: /test_statmach/dut
# Loading work.Timer4
# ** Warning: (vsim-3015) E:/My_testbench/tstatmach.v(17): [PCDPC] — Port size (4 or 4) does not match connection size (1) for port 'in'.
#    Region: /test_statmach/dut
# ** Warning: (vsim-3015) E:/My_testbench/tstatmach.v(17): [PCDPC] — Port size (5 or 5) does not match connection size (1) for port 'out'.
#    Region: /test_statmach/dut.
```

Рис. 18. Сообщения компилятора

```
run -all
# The input file was opened successfully.
# time bin decimal hex
# 0.000ns 0001      1 00000001
# 20.000ns 0010     20 00000020
# 50.020ns 0111     5 00000fff
# 62.340ns 0100     4 deadbeef
# 75.790ns 0xx1     2 zzzzzzzz
# 200.000ns 0010    20 00000020
# Break at E:/My_testbench/tstatmach.v line 103
```

Рис. 20. Сообщения компилятора

Появилось сообщение о том, что симуляция закончилась по команде $stop, находящейся на строке 103, в файле tstatmach.v.

Теперь в окне ***Wave*** появились временные диаграммы (рис. 21).

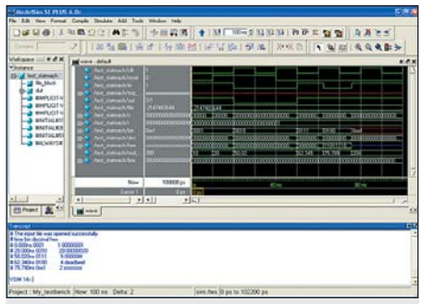

Рис. 21. Сигналы для модуля test_statmach изображены в окне Wave

Необходимо заметить, что, выполнив команду Run All, можно получить временную диаграмму, которая будет иметь неудобный для визуального восприятия масштаб. Возможно, диаграмма будет или сильно сжата, или, наоборот, растянута. Поэтому она визуально может не восприниматься как правильно выполненная. Чтобы изменить масштаб по времени для временных диаграмм, можно воспользоваться кнопками для изменения масштаба. Кнопка — «увеличить масштаб», кнопка — «уменьшить масштаб».

Если по каким-либо причинам моделирование не останавливается при симуляции файла, то такое бесконечное (зациклившееся) моделирование можно остановить командой главного меню ***Simulate→Break***, либо нажатием кнопки *Break* . Во время работы симулятора текущее время моделирования непрерывно увеличивается, и текущее время выводится внизу окна ***Wave***.

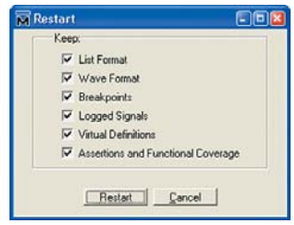

Рис. 22. Меню в окне Restart

Если мы хотим перезапустить симуляцию, то сначала нам необходимо будет выполнить Restart. Для этого нажимаем кнопку *Restart* . Откроется окно ***Restart*** (рис. 22). После нажатия на кнопку *Restart* в этом окне программа снова вернется в основное окно, при этом диаграммы сигналов будут очищены.

Теперь можно добавить еще несколько диаграмм в окно ***Wave***.

В окне ***Workspace*** выделим строку **dut** — это наш проверяемый модуль. В меню нужно выполнить команды ***Add→Wave→Selected Instance***. В результате этих действий сигналы компонента **dut**, установленного в модуле test_statmach, будут добавлены в окно ***Wave***.

После этого можно снова нажимать кнопку *Run*. Симулятор вновь выполняет моделирование (рис. 23) и при этом в окне ***Transcript*** выдаются сообщения (рис. 24).

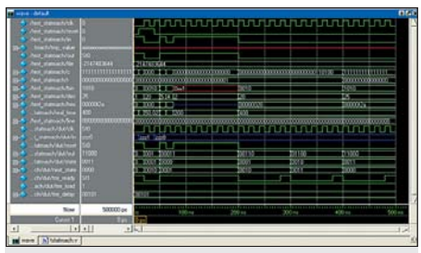

Рис. 23. Наблюдаемые сигналы для модулей test_statmach и dut в окне Wave

```
# time bin decimal hex
# 0.000ns 0001      1 00000001
# 20.000ns 0010     20 00000020
# 50.020ns 0111     5 00000fff
# 62.340ns 0100     4 deadbeef
# 75.790ns 0xx1     2 zzzzzzzz
```

Рис. 24. Сообщения при выполнении программы

Какие цели достигнуты в этом примере?

1. Показано, как создаются, устанавливаются и используются модули разных уровней. В этом примере есть модули трех уровней. Показано, как передавать параметры в установленный модуль. Показано, как привязывать сигналы к установленному компоненту.
2. Показано, как выполнять тестбенч с применением файлов паттерна. Преимущества такого выполнения проектов в том, что после изменений параметров в паттерне нет необходимости выполнять компиляцию проекта. Надо просто использовать Reset и запустить симуляцию.
3. Показано, как выводить на монитор информацию во время симуляции.

На этом моделирование примера будет завершено. При выходе из системы моделирования в рабочей директории появятся (рис. 25):

- директория ***work*** (в ней находятся скомпилированные файлы проекта);
- системный файл ***My_testbench.mpf*** сохраненного проекта цифровой системы (двойной щелчок вызовет систему моделирования и загрузит сохраненное состояние проекта);
- системный файл ***vsim.wlf*** временной диаграммы (двойной щелчок вызовет систему моделирования и откроет окно ***Wave*** с сохраненной временной диаграммой).

На этом описание работы с программой ModelSim закончим. В этом разделе показаны только самые необходимые действия для первого ознакомления с работой программы. Автор не ставил себе целью объяснить все тонкости ее работы. Поэтому он настоятельно рекомендует читателям самостоятельно более глубоко ознакомиться с руководством по использованию программы ModelSim.

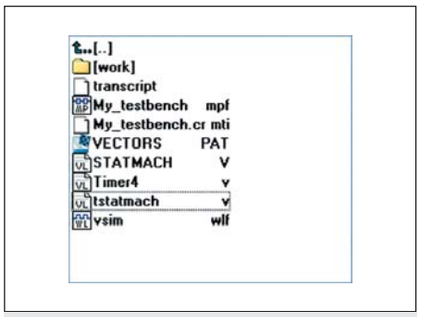

Рис. 25. Рабочая директория после сохранения проекта

В следующем разделе мы рассмотрим вопросы, связанные с использованием параметров при написании файла модели.

Литература

1. http://www.mentor.com/training_and_services/training/courses/functional_verification/202339.cfm
2. http://www.altera.com/support/software/nativelink/simulation/modelsim/eda_pro_msimfull_setup_proj.html
3. http://toolbox.xilinx.com/docsan/xilinx9/help/iseguide/mergedProjects/dkxilinx/html/pp_process_configure_modelsim_simulator.htm
4. www.megratec.ru