

# Использование приложения *DESim* с Verilog

В этом учебном пособии представлено приложение *DESim*, которое можно использовать для моделирования схем, заданных кодом на языке Verilog. Приложение *DESim* обеспечивает *графический интерфейс пользователя* (GUI), представляющий некоторые возможности платы *DE1-SoC*. Он служит как “front end” для симулятора *ModelSim*. С помощью *DESim* GUI можно вызывать как *компилятор*, так и *симулятор* ModelSim Verilog. Входные данные для симулятора *ModelSim* подаются нажатием на соответствующие элементы GUI *DESim*, который также выводит полученные симулятором результаты на отображающие элементы, имеющие вид тех, что расположены на реальной плате DE1-SoC.

## Содержание:

- Начало работы с *DESim*
- Компиляция и моделирование образцов проектов *DESim*
- Моделирование схемы, включающей модуль памяти
- Создание проекта *DESim*
- Устранение проблем при работе с приложением *DESim*

## Требования:

- Компьютер с ОС Microsoft® Windows® (рекомендованная версия: 10).
- Хорошее рабочее знание языка Verilog.
- ModelSim-Intel FPGA Starter Edition software, version 10.5b. Данное ПО должно быть установлено на компьютере, с которого будет запускаться *DESim* software. Требуемое *ModelSim* software является частью набора инструментов САПР *Quartus Prime*, предоставляемого корпорацией Intel®. Версия 10.5b ModelSim прилагается к ряду версий *Quartus* software, включая 18.0, 18.1, and 19.0.
- Вы должны знать, как использовать *ModelSim*® для моделирования кода Verilog с помощью *тестбенча*. Данный материал представлен в руководстве *Using the ModelSim-Intel FPGA Simulator with Verilog Testbenches*.
- Приложение *DESim*. Инструкции по загрузке и установке *DESim* находятся в руководстве *Installing the DESim Application*, доступном на сайте Intel FPGA Academic Program.

## Начало работы

Запустите программу *DESim*, чтобы открыть GUI, показанный на Рисунке 1. В верхней части *панели сообщений* должна отображаться строка “The server is running...”. Если такого не происходит, а вместо этого высвечивается сообщение **Server setup failed**, значит, *DESim* работает некорректно, и его следует закрыть. В этом случае обратитесь к разделу по устранению проблем при работе с приложением, расположенным в конце данного руководства.

В правой части Рисунка 1 элементы LEDs представляют собой красные светодиоды **LEDR<sub>9-0</sub>**, которые имеются на плате DE1-SoC. Переключатели (Switches) соответствуют ползунковым переключателям **SW<sub>9-0</sub>** платы, кнопки (Push Buttons) – **KEY<sub>3-0</sub>**, а семисегментные дисплеи (Seven-segment Displays) – **HEX5, HEX4, ..., HEX0**. Также в графическом интерфейсе есть несколько дополнительных блоков элементов: PS/2 Keyboard, Parallel Ports и VGA Display. Данные возможности *DESim* GUI не описываются в настоящем руководстве.

Рабочей единицей инструмента *DESim* является *проект*. Для ознакомления с возможностями *DESim* GUI сначала откроем какой-нибудь существующий проект. К примеру, возьмем многоразрядный сумматор

*addern*, который прилагается к *DESim* в качестве демонстрационного проекта. Выберите команду Open Project (Рисунок 1), чтобы открыть диалоговое окно, показанное на Рисунке 2. Затем перейдите в папку *demos*, выберите проект *addern* и нажмите кнопку **Select Folder**.

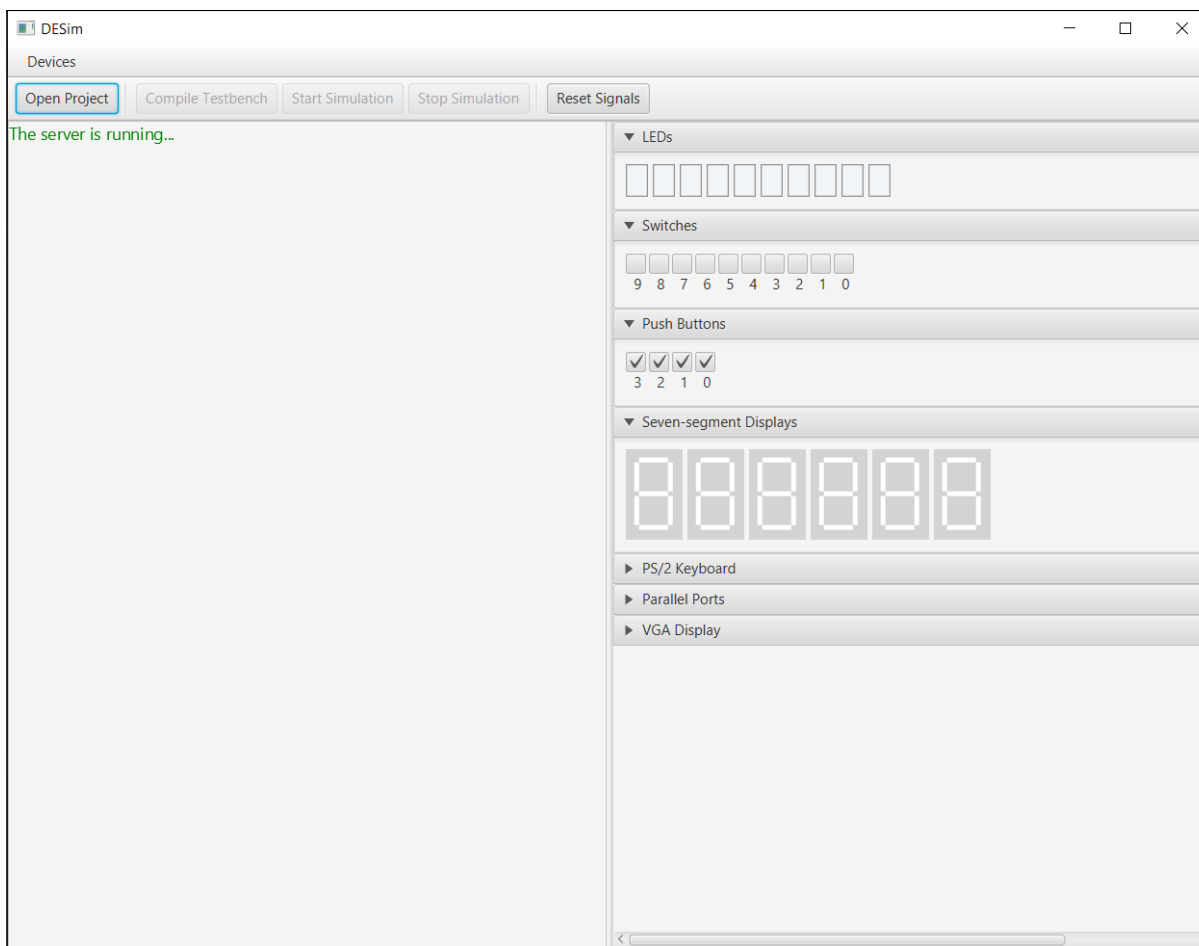


Рисунок 1: Графический интерфейс *DESim*.

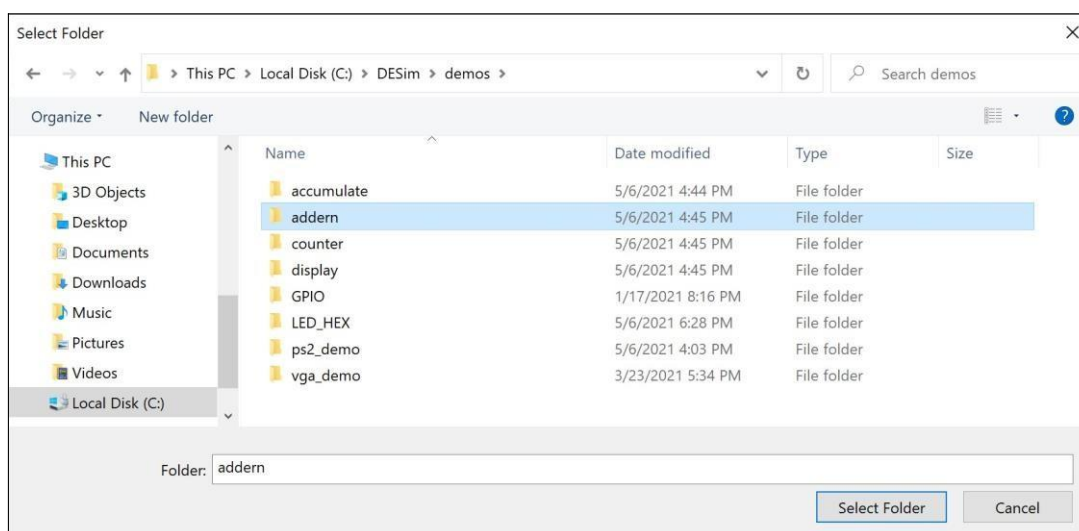


Рисунок 2: Открытие проекта *addern*.

В окне проводника Microsoft Windows на Рисунке 3 показано содержимое папки, в которой находится проект *addern*. Она состоит из папок с именами *sim* и *tb*, и файлов *Addern.v* и *top.v*. Также там расположен файл *Readme.txt*, но он лишь содержит документацию и в действительности не является частью системы проекта *DESim*.

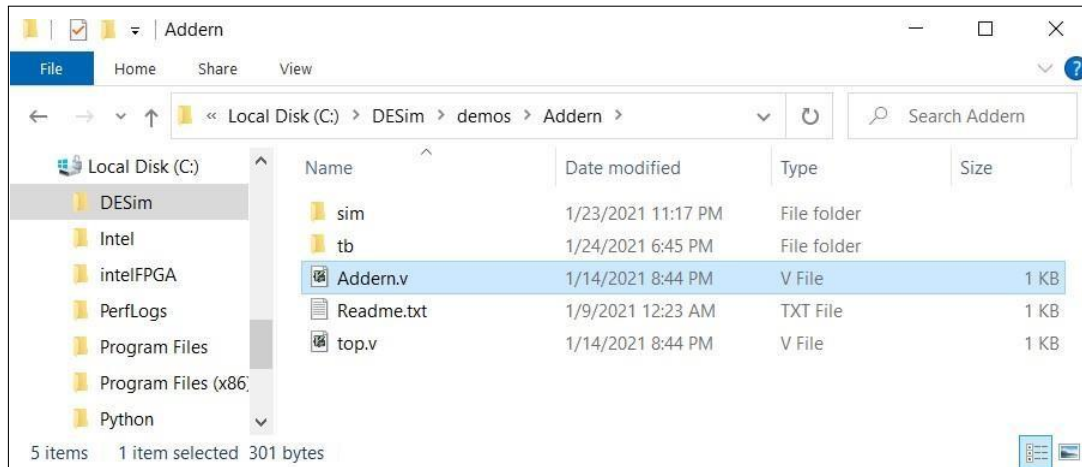


Рисунок 3: Папка *addern*.

Файл *Addern.v*, проиллюстрированный Рисунком 4, представляет собой Verilog код, на примере которого в данном разделе руководства будет разобран процесс моделирования. Будем использовать GUI *DESim* для подачи сигналов на входы сумматора *Cin*, *X* и *Y*, а затем выводить результаты моделирования, полученные для выходов *Sum* и *Cout*, на светодиоды (LEDs). Для установления соответствия между портами сумматора и сигналами с самого графического интерфейса *DESim*, мы *инстанцируем* модуль *Addern* в другом Verilog-модуле под названием *top*. Этот модуль определен в файле *top.v*, показанном на Рисунке 5. Его порты задействуют названия сигналов, которые используются в Verilog-модулях верхнего уровня иерархии (top-level), предназначенных для загрузки на плату DE1-SoC. Эти названия портов включают в себя *CLOCK\_50*, *SW*, *KEY*, *HEX0*, ... , *HEX5*, and *LEDR*. В модуле *Addern* мы, на самом деле, задействуем только некоторые из них, а остальные же оставляем неподключенными (эти неиспользуемые порты включены в список аргументов лишь для простоты согласованности с другими демонстрационными проектами *DESim*).

Модуль *Addern* инстанцируется в строке 15 файла *top.v* следующим объявлением:

```
Addern U1 (SW[9], SW[3:0], SW[7:4], LEDR[3:0], LEDR[4]);
```

Данное объявление подключает переключатель *SW<sub>9</sub>* к входу переноса сумматора *Cin*, а переключатели *SW<sub>3-0</sub>* и *SW<sub>7-4</sub>* – к информационным входам *X* и *Y* соответственно. Выход *Sum* подключен к *LEDR<sub>3-0</sub>*, а перенос *Cout* – к *LEDR<sub>4</sub>*.

Для компиляции проекта *addern* выберите в интерфейсе команду *Compile Testbench*. Она выполнит пакетный файл *run\_compile.bat*, который находится в папке *sim* проекта *addern*. Данный пакетный файл содержит некоторые команды *ModelSim*, приведенные ниже:

```
if exist work rmdir /S /Q work

vlib work
vlog ../tb/*.v
vlog ../*.v
```

Пакетный файл выполняет команду *vlib*, входящую в пакет *ModelSim* software, чем создает рабочую папку (сперва удалив таковую, если она уже существует). Затем пакетный файл дважды вызывает компилятор *ModelSim* Verilog compiler, *vlog*.

```
// A multi-bit adder
module Addern (Cin, X, Y, Sum, Cout);
    parameter n = 4;
    input Cin;
    input [n-1:0] X, Y;
    output [n-1:0] Sum;
    output Cout;

    assign {Cout, Sum} = X + Y + Cin;
endmodule
```

Рисунок 4: Листинг модуля *addern*.

```
1 module top (CLOCK_50, SW, KEY, LEDR, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
2
3     input CLOCK_50;                // DE-series 50 MHz clock signal
4     input wire [9:0] SW;           // DE-series switches
5     input wire [3:0] KEY;          // DE-series pushbuttons
6     output wire [9:0] LEDR;        // DE-series LEDs
7
8     output wire [6:0] HEX0;        // DE-series HEX displays
9     output wire [6:0] HEX1;
10    output wire [6:0] HEX2;
11    output wire [6:0] HEX3;
12    output wire [6:0] HEX4;
13    output wire [6:0] HEX5;
14
15    Addern U1 (SW[9], SW[3:0], SW[7:4], LEDR[3:0], LEDR[4]);
16
17 endmodule
```

Рисунок 5: Листинг модуля *top*.

Первый вызов компилятора, `vlog ../tb/*.v`, компилирует исходный код Verilog в папке `tb` проекта *addern*. В этой папке хранится *тестбенч* проекта, описание которого приведено далее. Второй вызов `vlog` компилирует исходный код проекта *DESim*, который включает файлы *addern.v* и *top.v*. Все сообщения, полученные при выполнении *run\_compile.bat*, отображаются в панели сообщений *DESim* GUI, как показано на Рисунке 6.

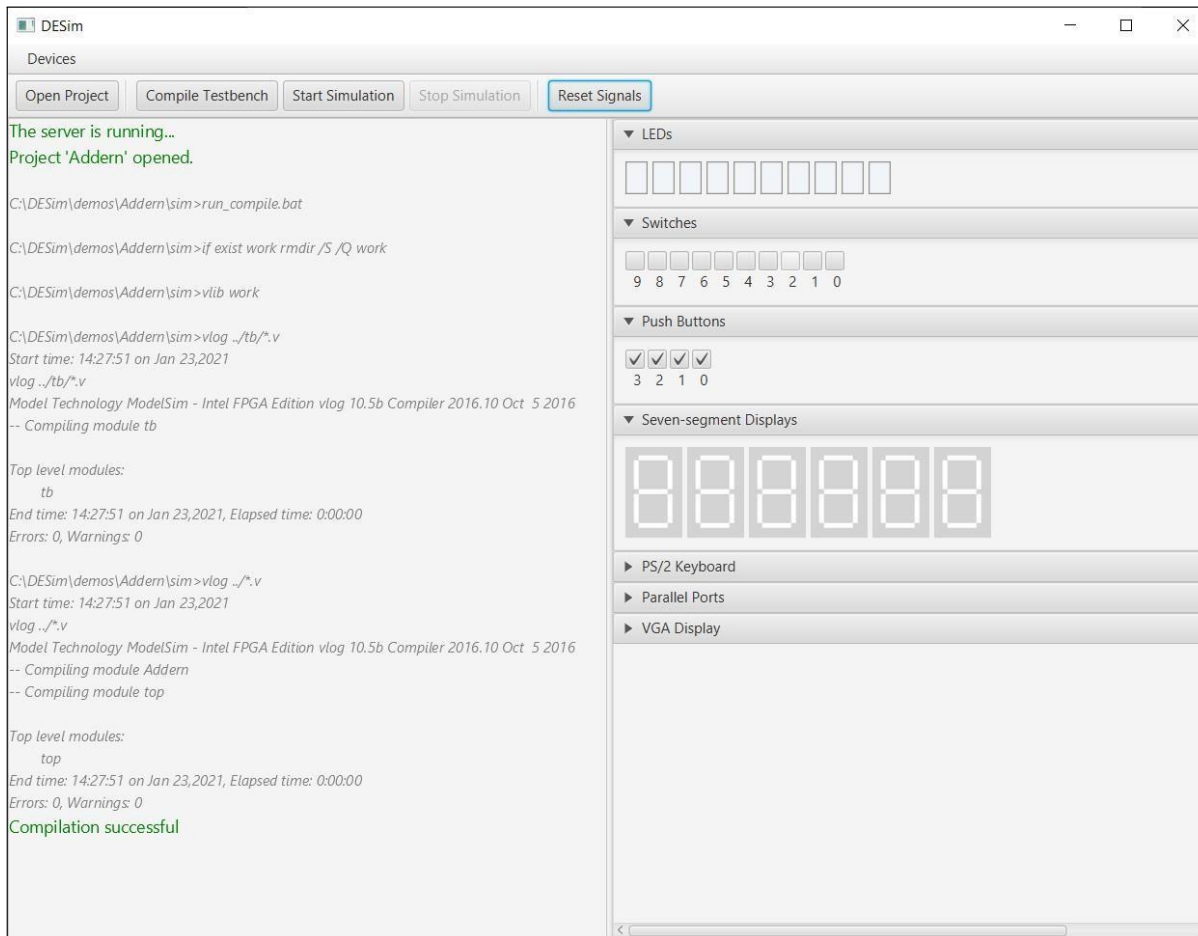


Рисунок 6: Сообщения, полученные при выполнении *run\_compile.bat*.

Файл тестбенча для проекта *addern*, который компилируется командой `vlog ./tb/*.v`, называется *tb.v* и показан на Рисунке 7. Для использования программы *DESIm* нет необходимости изменять (или даже изучать) большую часть этого кода, но для полноты картины мы описываем некоторую его часть здесь. Код тестбенча на рисунке имеет общую структуру, позволяющую использовать его для моделирования абсолютно разных Verilog-кодов, которые затем могут быть использованы в различных проектах *DESIm*. Следовательно, не весь код данного тестбенча необходим для проекта *addern*. В строке 6 объявляется модуль тестбенча, который называется *tb*. Следующие строки кода объявляют некоторые сигналы, которые используются в тестбенче. Объявление

```

initial $sim_fpga(CLOCK_50, SW, KEY, LEDR, HEX, key_action, scan_code,
                 ps2_lock_control, VGA_X, VGA_Y, VGA_COLOR, plot, GPIO);
  
```

является уникальным для программы *DESIm*. Оно использует специальную функцию *ModelSim* software, которая обеспечивает связь с пользовательской функцией (*custom software function*). В данном случае пользовательская функция является частью ПО *DESIm* и называется *sim\_fpga*. Она хранится в файле *simfpga.vpi*, который должен быть включен в папку *sim* каждого *DESIm* проекта. Графический интерфейс *DESIm* отправляет/получает значения сигналов в/из *ModelSim* через функцию *sim\_fpga*. Эта возможность *ModelSim* известна как *процедурный интерфейс Verilog (VPI)*.

```

1  `timescale 1ns / 1ns
2  `default_nettype none
3
4  // This testbench is designed to hide the details of using the VPI code
5
6  module tb();
7
8      reg CLOCK_50 = 0;           // DE-series 50 MHz clock
9      reg [9:0] SW = 0;           // DE-series SW switches
10     reg [3:0] KEY = 0;           // DE-series pushbutton keys
11     wire [(8*6) -1:0] HEX;       // HEX displays (six ports)
12     wire [9:0] LEDR;             // DE-series LEDs
13
14     reg key_action = 0;
15     reg [7:0] scan_code = 0;
16     wire [2:0] ps2_lock_control;
17
18     wire [7:0] VGA_X;             // "VGA" column
19     wire [6:0] VGA_Y;             // "VGA" row
20     wire [2:0] VGA_COLOR;         // "VGA pixel" colour (0-7)
21     wire plot;                   // "Pixel" is drawn when this is pulsed
22     wire [31:0] GPIO;            // DE-series GPIO port
23
24     initial $sim_fpga(CLOCK_50, SW, KEY, LEDR, HEX, key_action, scan_code,
25                       ps2_lock_control, VGA_X, VGA_Y, VGA_COLOR, plot, GPIO);
26
27     wire [6:0] HEX0;              // DE-series HEX0 port
28     wire [6:0] HEX1;              // DE-series HEX1 port
29     wire [6:0] HEX2;              // ...
30     wire [6:0] HEX3;
31     wire [6:0] HEX4;
32     wire [6:0] HEX5;
33
34     // create the 50 MHz clock signal
35     always #10
36         CLOCK_50 <= ~CLOCK_50;
37
38     // connect the single HEX port on "sim_fpga" to the six DE-series HEX ports
39     assign HEX[47:40] = {1'b0, HEX0};
40     assign HEX[39:32] = {1'b0, HEX1};
41     assign HEX[31:24] = {1'b0, HEX2};
42     assign HEX[23:16] = {1'b0, HEX3};
43     assign HEX[15: 8] = {1'b0, HEX4};
44     assign HEX[ 7: 0] = {1'b0, HEX5};
45
46     top DUT (.CLOCK_50(CLOCK_50), .SW(SW), .LEDR(LEDR), .KEY(KEY), .HEX0(HEX0),
47             .HEX1(HEX1), .HEX2(HEX2), .HEX3(HEX3), .HEX4(HEX4), .HEX5(HEX5));
48
49 endmodule

```

Рисунок 7: Файл тестбенча, *tb.v*, для проекта *addern*.

Строка 46 в коде тестбенча инстанцирует тестируемый объект (DUT – *design under test*), который представляет собой Verilog-модуль под названием *top*, показанный на Рисунке 5. Чтобы выполнить тестбенч с помощью симулятора *ModelSim*, нажмите на команду Start Simulation в GUI *DESIm*. Эта команда запускает *пакетный* файл *run\_sim.bat*, который находится в папке *sim* проекта *addern*. Данный пакетный файл запускает *vsim*, симулятор *ModelSim* Verilog, с помощью команды:

```
vsim -pli simfpga.vpi -Lf 220model -Lf altera_mf_ver -Lf verilog -c -do "run -all" tb
```

Аргумент `-pli` для программы `vsim` говорит ей ссылаться на программную функцию `sim_fpga`, которая была (ранее) скомпилирована в файл `simfpga.vpi`. Аргументы с буквой `-L` подключают некоторые библиотеки моделирования для ПЛИС Intel, которые могут понадобиться симулятору. Наконец, остальные аргументы запускают моделирование для модуля верхнего уровня, которым является `tb`. Любые сообщения, полученные при выполнении `run_sim.bat`, отображаются в панели сообщений графического интерфейса `DESim`, как показано на Рисунке 8.

Как упоминалось ранее, проект `addern` включает в себя файл `Readme.txt`, который документирует его использование. Содержание файла показано на Рисунке 9. Вы можете следовать инструкциям в нем, чтобы увидеть, как в проекте используются переключатели и светодиоды (конечно, Вы также можете узнать это, просмотрев исходный код Verilog). Пример результата моделирования проиллюстрирован Рисунком 8. Он соответствует значениям параметров  $Cin = 1$ ,  $X = (0110)_2 = (6)_{10}$  и  $Y = (1010)_2 = (10)_{10}$ . Результат сложения равен  $(10001)_2 = (17)_{10}$  ( $Cout = 1$ ,  $Sum = (0001)_2$ ), что отображается на светодиодах. Попробуйте по-разному выставлять переключатели SW и наблюдайте за результатами, отображаемыми на LEDs.

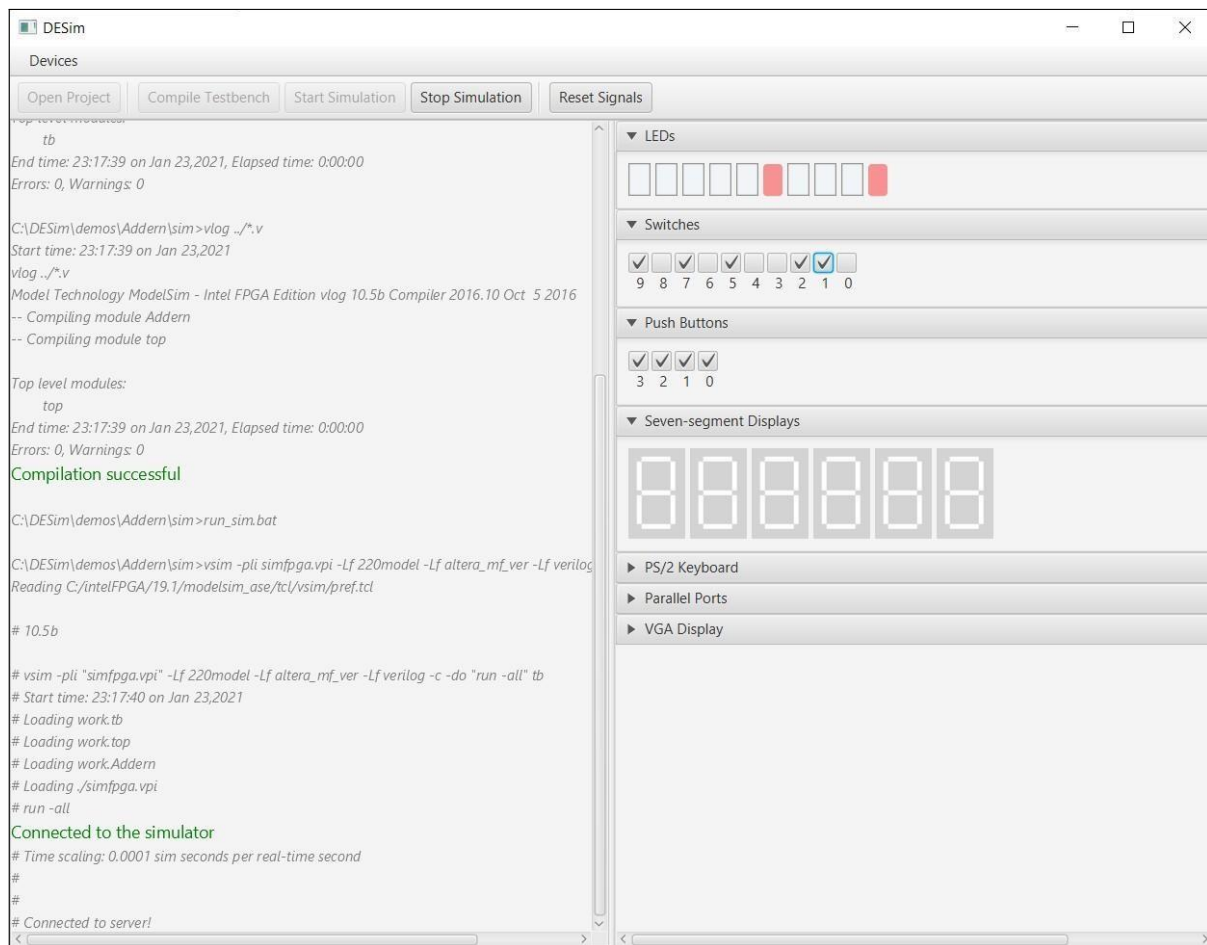


Рисунок 8: Сообщения, полученные при выполнении `run_sim.bat`.

На этом разбор проекта-образца `addern` подходит к концу.



To use this demo:

```
-- set a value, X, using SW[3:0]
-- set a value, Y, using SW[7:4]
-- set a carry-in, Cin, using SW[9]
```

The circuit produces the 5-bit Sum =  $X + Y + \text{Cin}$ , which is displayed on LED[4:0]

Рисунок 9: Файл *Readme.txt* для проекта *addern*

## Моделирование последовательстной схемы

Другой проект-образец *DESim* под названием *counter* также включен в папку *DESim demos*. Используйте команду *Open Project* в GUI *DESim*, чтобы открыть этот проект. Как показано на Рисунке 10 в Проводнике Microsoft Windows, содержимое папки файловой системы для этого проекта выглядит так же, как и для проекта *addern* (Рисунок 3), за исключением только имени файла исходного Verilog-кода *Counter.v*. На Рисунке 11 показано содержимое файла *Counter.v*. Он представляет собой 24-битный счетчик с синхронным сбросом. Названия портов для этого модуля соответствуют названиям сигналов платы DE1-SoC: *KEY<sub>0</sub>* используется для сброса, *CLOCK\_50* – для тактового сигнала, а *LEDR* – для выходов. Поскольку *LEDR* – это 10-битный сигнал, а рассчитан на 24 бита, к *LEDR* подключено только подмножество выходов счетчика (самые значимые).

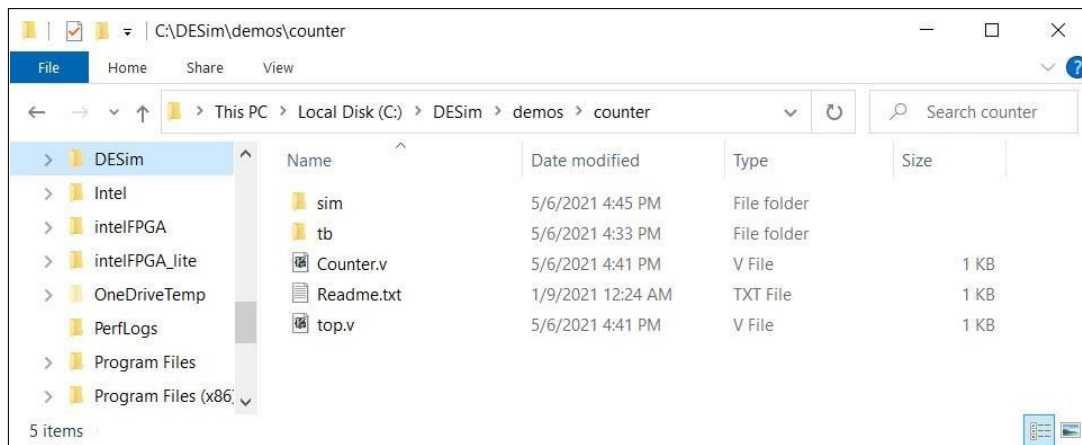


Рисунок 10: Папка *counter*.

Как описано для проекта *addern*, модуль *Counter* инстанцируется в другом модуле Verilog под названием *top*. Этот модуль отражен на Рисунке 12. Он абсолютно такой же, как и на Рисунке 5, за исключением того, что строка 15 инстанцирует модуль с именем *Counter*. Порт *KEY[0]* в модуле *top* подключен к входу сброса счетчика, *CLOCK\_50* – к его тактовому входу, а *LEDR* – к порту выхода счетчика.

Чтобы скомпилировать проект *counter*, в графическом интерфейсе *DESim* нажмите на команду *Compile Testbench*. Эта команда выполняет пакетный файл *run\_compile.bat*, который находится в папке *sim* проекта *counter*. Этот пакетный файл идентичен тому, который был описан ранее для проекта *addern*. Файл тестбенча, который компилируется благодаря *run\_compile.bat* для проекта *counter*, находится в его папке *tb*. Этот тестбенч, *tb.v*, идентичен тому, что был приведен на Рисунке 7.



```

module Counter (KEY, CLOCK_50, LEDR);
    input [0:0] KEY;
    input CLOCK_50;
    output [9:0] LEDR;
    parameter n = 24;

    reg [n-1:0] Count;
    wire Clock, Resetn;

    assign Clock = CLOCK_50;
    assign Resetn = KEY[0];

    // the counter
    always @(posedge Clock)
        if (Resetn == 1'b0)           // synchronous clear
            Count <= 0;
        else
            Count <= Count + 1;

    assign LEDR = Count[n-1:n-10];
endmodule

```

Рисунок 11: Листинг модуля 24-битного счетчика.

Чтобы выполнить тестбенч для проекта *counter*, нажмите на команду Start Simulation в графическом интерфейсе *DESim*. Эта команда запускает пакетный файл *run\_sim.bat*. Он идентичен описанному для проекта *addern* и запускает симулятор *vsim* Verilog.

```

1 module top (CLOCK_50, SW, KEY, LEDR, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
2
3     input CLOCK_50;           // DE-series 50 MHz clock signal
4     input wire [9:0] SW;      // DE-series switches
5     input wire [3:0] KEY;     // DE-series pushbuttons
6     output wire [9:0] LEDR;   // DE-series LEDs
7
8     output wire [6:0] HEX0;    // DE-series HEX displays
9     output wire [6:0] HEX1;
10    output wire [6:0] HEX2;
11    output wire [6:0] HEX3;
12    output wire [6:0] HEX4;
13    output wire [6:0] HEX5;
14
15    Counter U1 (KEY[0], CLOCK_50, LEDR);
16
17 endmodule

```

Рисунок 12: Top модуль для проекта *counter*.

В файле *Readme.txt* для проекта счетчика указывается:

To use this demo, reset the circuit by pressing and releasing KEY[0].

В графическом интерфейсе *DESim*, когда на кнопках стоит галочка, они установлены в значение 1. Для сброса цепи счетчика кликните на кнопку KEY<sub>0</sub> один раз, чтобы сначала ее нажать (это установит соответствующий сигнал для этой кнопки в 0), а затем кликните на кнопку еще раз, чтобы ее отжать. 24-битный счетчик начнет работать, и десять наиболее значимых выходов счетчика будут выводиться на светодиоды LEDs. Снимок экрана графического интерфейса *DESim* во время моделирования проекта счетчика приведен на Рисунке 13.

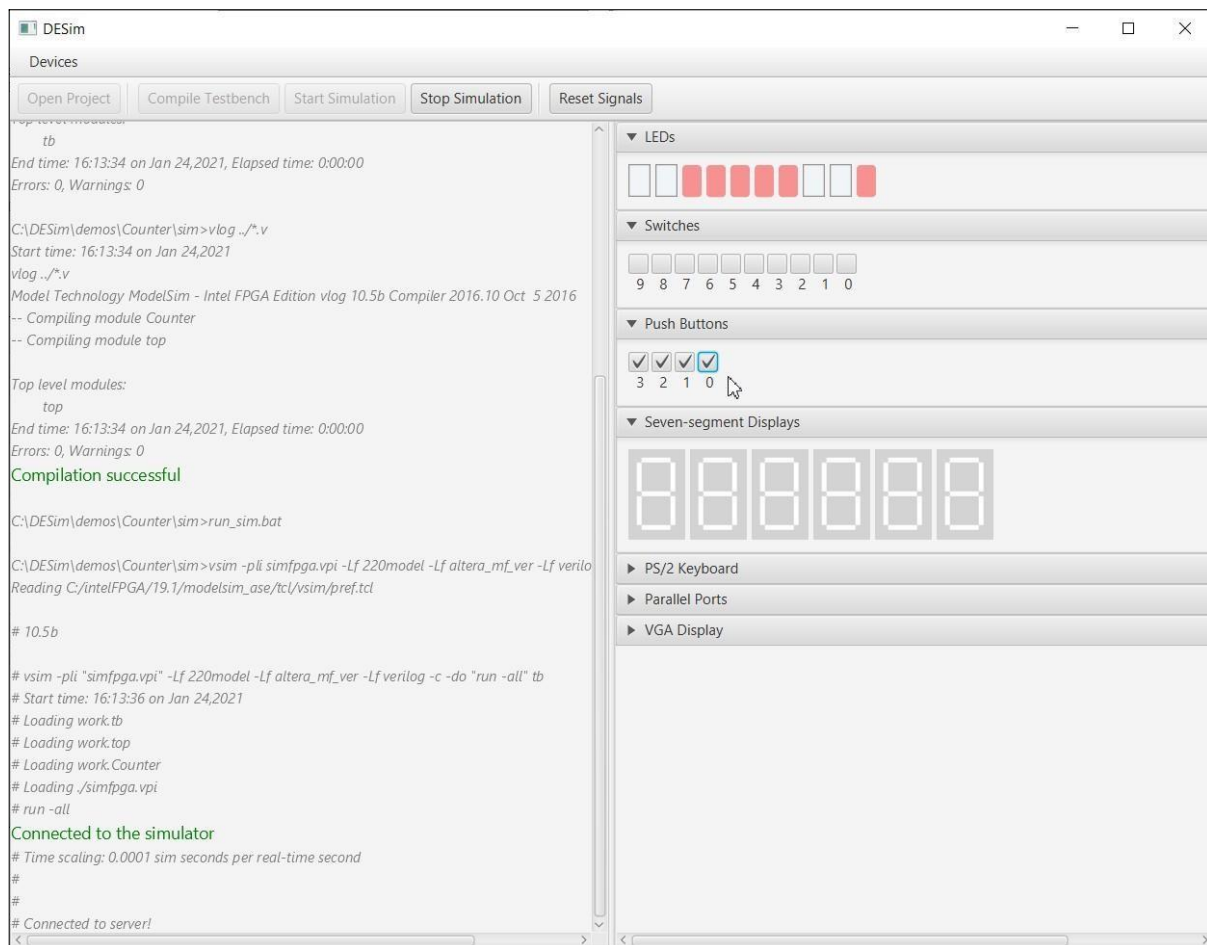


Рисунок 13: Моделирование проекта *counter*.

## Моделирование схемы, включающей модуль памяти

В папке *DESim* demos находится проект под названием *display*. Он показывает, как кодом Verilog инстанцировать модуль памяти и как инициализировать хранимое в ней содержимое во время моделирования *DESim*. Используйте команду *Open Project*, чтобы открыть этот пример проекта. Как проиллюстрировано на Рисунке 14, содержимое папки файловой системы этого проекта похоже на предыдущие, но есть еще два дополнительных файла: *inst\_mem.v* и *inst\_mem.mif*. Эти файлы используются для модуля памяти в схеме, который будет описан далее.

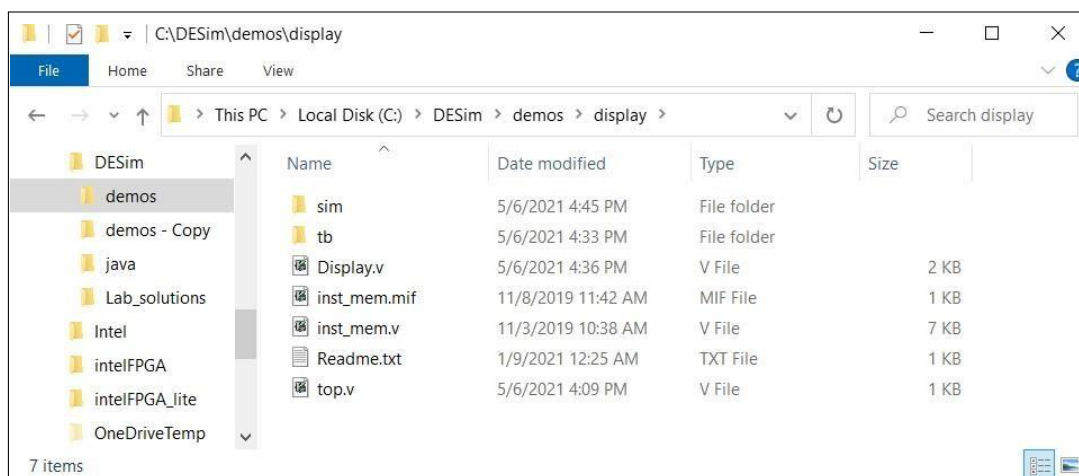


Рисунок 14: Папка *display*.

Рисунок 15 отражает Verilog-код для *Display.v*, который имеет порты с названиями *KEY*, *SW*, *HEX0* и *LEDR*. На Рисунке 16а приведена логическая схема, соответствующая листингу на Рисунке 15. Схема содержит счетчик, который используется для считывания содержимого последовательных мест в памяти. Эта память представляет коды в формате ASCII для некоторых букв верхнего и нижнего регистров, которые подаются на вход модуля дешифратора. Счетчик и модули памяти имеют общий тактовый сигнал, а счетчик имеет синхронный вход очистки. Каждый следующий тактовый цикл продвигает счетчик и считывает новый код ASCII из памяти. Поскольку счетчик имеет ширину три бита, считываются только первые восемь мест в памяти (старшие два бита адреса памяти установлены в 00), и они обеспечивают коды ASCII для букв A, b, C, d, E, F, g и h. Дешифратор приводит в действие соответствующий код для отображения каждой буквы на семисегментном дисплее.

Память, используемая в логической схеме, изображена в части *b* Рисунка 16. Это синхронная 32x8 память только для чтения (ROM), которая имеет регистр для хранения значений адреса. Память задана в файле *inst\_mem.v*, и инициализируется содержимым файла *inst\_mem.mif*, который проиллюстрирован Рисунком 17. Этот файл содержит коды ASCII для восьми букв, отображаемых схемой.

```

module Display (KEY, SW, HEX0, LEDR);
    input [0:0] KEY;
    input [0:0] SW;
    output reg [6:0] HEX0;
    output [9:0] LEDR;

    parameter A = 8'd65, b = 8'd98, C = 8'd67, d = 8'd100, E = 8'd69,
        F = 8'd70, g = 8'd103, h = 8'd104;
    wire Resetn, Clock;
    wire [2:0] Count;
    wire [7:0] char;

    assign Resetn = SW[0];
    assign Clock = KEY[0];

    count3 U1 (Resetn, Clock, Count);
    inst_mem U2 ({2'b0, Count}, Clock, char);
    assign LEDR = {2'b0, char};

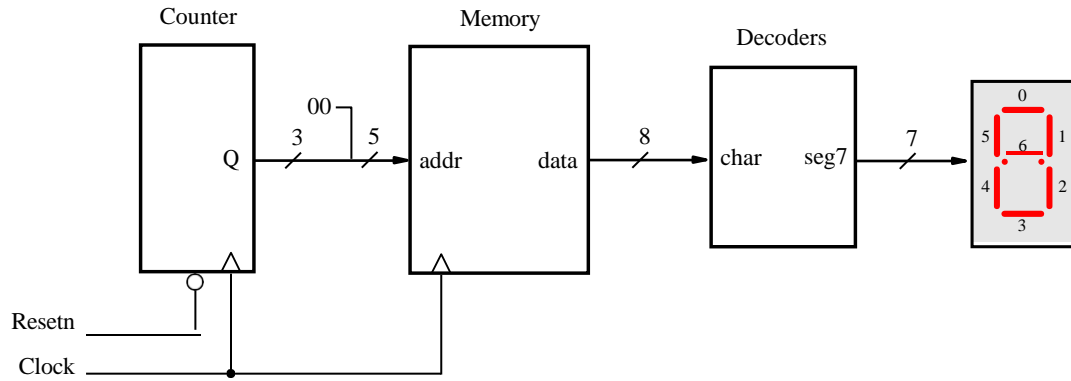
    always @(*)
        case (char)
            A: HEX0 = 7'b0001000;
            b: HEX0 = 7'b0000011;
            C: HEX0 = 7'b1000110;
            d: HEX0 = 7'b0100001;
            E: HEX0 = 7'b0000110;
            F: HEX0 = 7'b0001110;
            g: HEX0 = 7'b0010000;
            h: HEX0 = 7'b0001011;
            default HEX0 = 7'b1111111;
        endcase
endmodule

module count3 (Resetn, Clock, Q);
    input Resetn, Clock;
    output reg [2:0] Q;

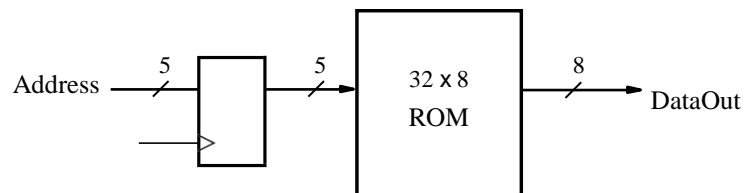
    always @ (posedge Clock)
        if (Resetn == 0)
            Q <= 3'b000;
        else
            Q <= Q + 1'b1;
endmodule

```

Рисунок 15: Verilog-код для проекта *display*.



a) схема



b) модуль памяти

Рисунок 16: Схема, представляющая проект *display*.

```

DEPTH = 32;
WIDTH = 8;
ADDRESS_RADIX = HEX;
DATA_RADIX = DEC;
CONTENT
BEGIN
    00 : 65;          % A %
    01 : 98;          % b %
    02 : 67;          % C %
    03 : 100;         % d %
    04 : 69;          % E %
    05 : 70;          % F %
    06 : 103;         % g %
    07 : 104;         % h %
END;
```

Рисунок 17: Файл инициализации памяти *inst\_mem.mif*.

Чтобы скомпилировать проект *display*, в графическом интерфейсе *DESim* нажмите команду Compile Testbench. Эта команда выполняет скрипт *run\_compile.bat* проекта, который находится в его папке *sim*. Этот пакетный файл показан ниже:

```
1      if exist ..\inst_mem.mif (
2          copy /Y ..\inst_mem.mif .
3      )
4      if exist ..\inst_mem_bb.v (
5          del ..\inst_mem_bb.v
6      )
7      if exist work rmdir /S /Q work
8
9      vlib work
10     vlog ../tb/*.v
11     vlog ../*.v
```

Строки 1-3 используются для копирования файла инициализации памяти *inst\_mem.mif* из папки проекта *display* в папку *sim*. Это делается по двум причинам: 1. *ModelSim* требует, чтобы файл находился в папке *sim* для правильной инициализации модуля памяти во время моделирования, и 2. Если файл изменен в папке *display*, то при запуске моделирования всегда будет использоваться последняя версия файла. Строки 4-6 удаляют файл *inst\_mem\_bb.mif*, который иногда ассоциируется с модулем памяти; если бы он присутствовал, то вызвал бы ошибку *ModelSim*. Остальная часть пакетного файла, который компилирует Verilog-код, такая же, как и для ранее описанных проектов *DESim*.

Файл testbench *tb.v*, который компилируется благодаря *run\_compile.bat* для проекта *display*, идентичен тому, который используется для проекта *addern* и был показан на Рисунке 7. Чтобы выполнить тестбенч для проекта *display*, нажмите на Start Simulation. Его скрипт *run\_sim.bat* идентичен скрипту, используемому для проектов *addern* и *counter*.

Файл *Readme.txt* для проекта *display* показан на Рисунке 18. Вы можете следовать его инструкциям для считывания последовательных мест памяти и отображения соответствующих символов на HEX0. Пример результата моделирования после первого сброса схемы и последующего создания нескольких тактовых сигналов с помощью KEY[0] отражен на Рисунке 19.

To use this demo:

```
-- The clock input is created by toggling KEY[0]
-- The active-low synchronous reset input is SW[0]
```

The circuit displays "characters" stored in a ROM on HEX0.  
To use the circuit:

1. Set SW[0] to 0 to allow the circuit to be reset
2. pulse KEY[0] down/up to make a clock cycle  
-- the character 'A', the first character stored in the ROM should be displayed on HEX0
3. Set SW[0] to 1 so that the reset is not active
4. pulse KEY[0] down/up to make a clock cycle
5. pulse KEY[0] down/up to make a clock cycle  
-- HEX0 should now show 'b', the next character stored in the ROM
6. pulse KEY[0] down/up to make a clock cycle  
-- HEX0 should now show 'C', the next character stored in the ROM
7. etc (there are eight characters stored in the ROM)

Рисунок 18: Файл *Readme.txt* для проекта *display*.

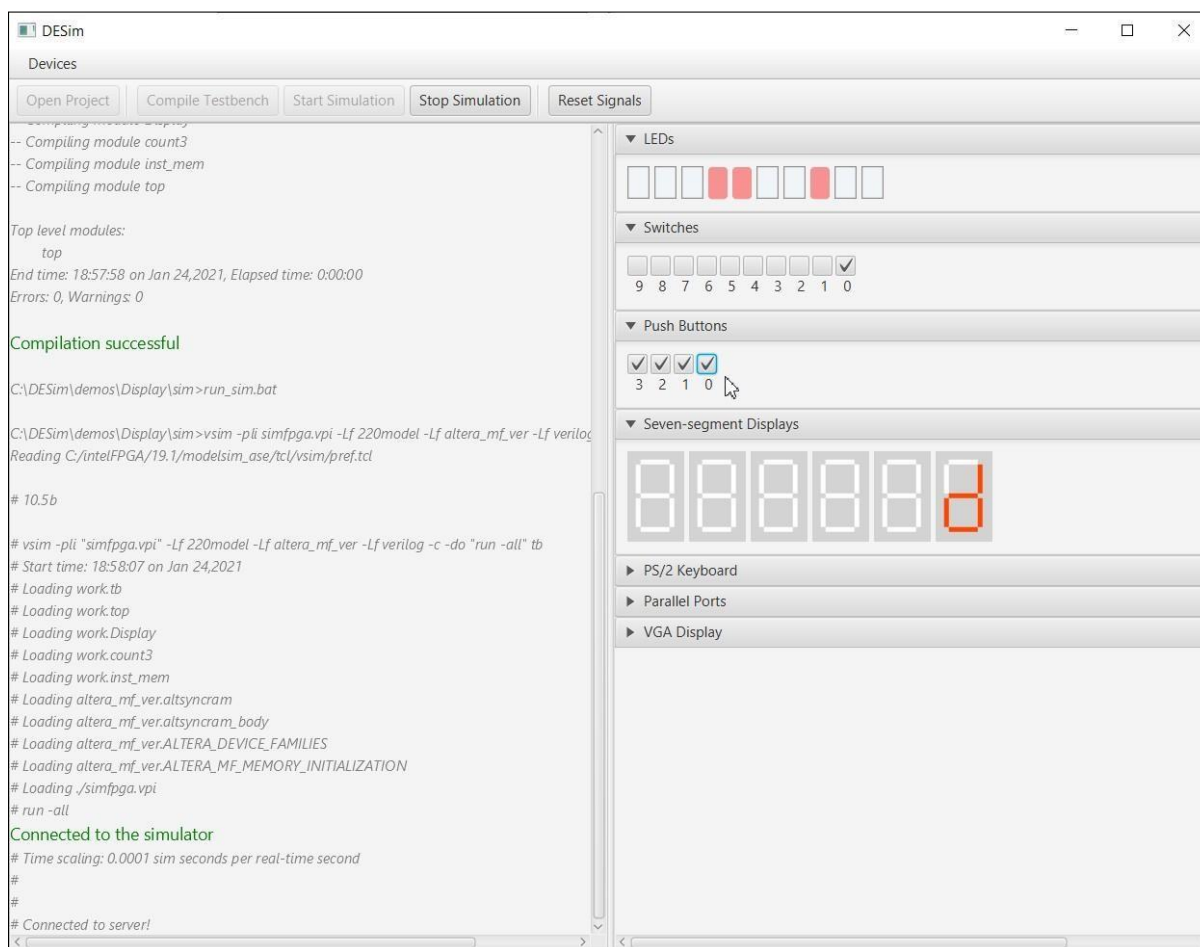


Рисунок 19: Моделирование проекта *display*.

## Настройка проекта *DESIm*

Простой способ создать свой собственный проект *DESIm* – использовать один из примеров проектов в папке *DESIm demos* в качестве основы. Вы должны выбрать конкретный demo проект, ориентируясь на его особенности. Например, если Ваш проект включает модуль памяти, то можно начать с копии проекта *display*. Если же модуль памяти не требуется, то можно начать с копии любого из других проектов. Вдобавок, стоит следить, чтобы в "top"-модуле проекта присутствовали все нужные Вам порты. Все примеры проектов, описанные в данном документе, используют одинаковый набор портов в своих top модулях, но в некоторых других demo проектах используемые порты могут иметь некоторые отличия.

Взяв за основу подходящий проект из папки *demos*, скопируйте его содержимое в новую папку на своем ПК. Например, можно сделать копию папки *demos\display* и назвать новую папку *my\_folder*. Затем в папке *my\_folder* нужно будет заменить файл *Display.v* на собственный файл с исходным кодом, скажем *my\_source.v*. Далее Вам предстоит изменить файл *top.v* в папке *my\_folder* так, чтобы он инстанцировал Ваш Verilog-модуль, скажем *my\_module* (который будет находиться в файле *my\_source.v*). После этого подключите сигналы в файле *top.v*, такие как *CLOCK\_50*, *KEY* и т.д., к портам модуля *my\_module*, как того требует Ваш код. Если какие-то порты, необходимые для *my\_module*, отсутствуют в модуле *top*, то следует использовать другой пример проекта из папки *demos*, который все-таки будет иметь необходимые порты в своем модуле *top*.

Не требуется вносить никаких изменений в файлы папок *sim* или *tb* для нового проекта в *my\_folder*. Теперь Вы можете открыть свой новый проект в *DESIm* и приступить к его компиляции/моделированию.



## Устранение проблем при работе с приложением *DESim*

В этом разделе рассматриваются некоторые потенциальные проблемы, которые могут возникнуть при использовании программного обеспечения *DESim*, и предлагаются их решения.

1. После запуска программного обеспечения *DESim* Вы должны увидеть сообщение **The server is running...** в верхней части панели сообщений графического интерфейса пользователя. Если Вы его не видите, а вместо этого появляется сообщение **Server setup failed**, значит, программное обеспечение *DESim* работает неправильно и его следует закрыть. Это может произойти в том случае, если Вы запустили *DESim* в еще одном окне. Программа *DESim* не может быть запущена более одного раза одновременно на Вашем компьютере.
2. Если нажать на команду **Compile Project** в графическом интерфейсе *DESim*, можно увидеть сообщение об ошибке типа **'vlib' is not recognized as an internal or external command**. Эта ошибка означает, что *DESim* попытался выполнить программу *vlib*, которая является частью *ModelSim* software, но программа была не найдена операционной системой. Такая ошибка возникает, либо если программа *ModelSim* не установлена на компьютере, либо если она установлена, но не может быть найдена. Существуют два способа решения второй упомянутой проблемы: 1) можно обновить переменную окружения Microsoft Windows Path так, чтобы она включала расположение программы *ModelSim*, или 2) можно явно указать расположение программы *ModelSim* в пакетном файле, который запускает *DESim*. Этот пакетный файл называется *DESim\_run.bat* и находится в папке файловой системы, где установлен *DESim*. Это второе решение используется в системах DESL и ECF. Поэтому если Вы столкнетесь с данной ошибкой на одной из этих компьютерных систем, убедитесь, что Вы используете правильную версию *DESim* software; для DESL и для ECF используются разные версии. Обратите внимание, что Вы можете определить, какая система используется, посмотрев на имя диска C: на компьютере.
3. Иногда при компиляции или моделировании проекта в программе *DESim* Вы можете увидеть предупреждающее сообщение, в котором говорится, что *ModelSim* не может "разъединить" ("unlink") файл. Например, если Ваш проект *DESim* хранится в папке *C:\DESim\demos\addern*, то это сообщение выведет:

```
** Warning: (vlog-31) Unable to unlink file "C:/DESim/demos/addern/sim/work/_lock"
```

Данное предупреждение возникает по неизвестным причинам и вызвано проблемами с самим *ModelSim* software (это происходит и при непосредственном использовании GUI *ModelSim*, а не только при использовании *DESim*). Если проблема с "unlink" сохраняется (иногда она решается автоматически), то решением является поиск в Проводнике Microsoft Windows папки файловой системы *C:\DESim\demos\addern\sim\work* и удаления файла с именем *\_lock* вручную.