

Оглавление

ВВЕДЕНИЕ	2
МАРШРУТ ПРОЕКТИРОВАНИЯ НА ОСНОВЕ ПЛИС	4
СТРУКТУРА ПРОГРАММНЫХ ДИРЕКТОРИЙ	7
ЗАПУСК ПРОГРАММЫ В КОНСОЛЬНОМ РЕЖИМЕ	8
ВХОДНЫЕ ДАННЫЕ ПРОГРАММЫ	13
БИБЛИОТЕКИ ЭЛЕМЕНТОВ ПЛИС	16
ИСПОЛЬЗОВАНИЕ КОНТАКТНЫХ ПЛОЩАДОК	17
ФАЙЛ РАЗМЕЩЕНИЯ КОНТАКТНЫХ ПЛОЩАДОК	21
ИСПОЛЬЗОВАНИЕ ИМЕН КОНТАКТНЫХ ПЛОЩАДОК	24
ВЫПОЛНЕНИЕ СТАТИЧЕСКОГО ВРЕМЕННОГО АНАЛИЗА	25
ВЫХОДНЫЕ ДАННЫЕ ПРОГРАММЫ	41
ПРИМЕРЫ ЗАПУСКА ПРОГРАММЫ В КОНСОЛЬНОМ РЕЖИМЕ	44
TCL ИНТЕРФЕЙС	45
ГРАФИЧЕСКИЙ ИНТЕРФЕЙС X-CAD	48
ИСПОЛЬЗОВАНИЕ ВНЕШНИХ ПРОГРАММ	73
Icarus Verilog	74
GTK Wave	81
Анализатор HDL-описаний	82
ИНТЕРФЕЙС ТОПОЛОГИЧЕСКОГО РЕДАКТИРОВАНИЯ X-PLACE	83
ИНТЕРФЕЙС ПЛАНИРОВКИ КРИСТАЛЛА FLOORPLANNER	99
Использование полученного размещения	110
Лист регистрации изменений	112

ВВЕДЕНИЕ

Программный пакет X-CAD предназначен для автоматизации этапов логического синтеза, размещения элементов и трассировки межсоединений микросхем, проектируемых на базе программируемых логических интегральных схем (ПЛИС).

В состав пакета программных средств входят:

- интерфейсные программы на языке C++, обеспечивающие загрузку схемной и топологической информации из базы данных САПР Cadence в форматах CDL, GDS-II.
- Программы на языке C++, интегрированные с языком управления заданиями Tcl и обеспечивающие отображение проектируемой пользовательской схемы на базис элементов библиотеки ПЛИС, а также реализующие прохождение полного маршрута топологического проектирования, состоящего из автоматических процедур планировки, размещения и трассировки межсоединений.
- Скрипты на языке Tcl для поддержки функционирования интерфейса программы X-CAD и управления процессом конфигурирования ПЛИС.
- Библиотеки стандартных элементов в формате Liberty для автоматического синтеза проектных решений средствами Design Compiler (Synopsys), RTL Compiler (Cadence), Genus (Cadence) или Yosys Open SYnthesis Suite (Yosys).

1. Характеристики ПЛИС

Шифр ПЛИС	5400TC015
Параметры	
Тех. процесс, нм	180
Емкость, ЛЭ	1104
Количество стандартных ЯВВ, шт	93
Количество тактовых ЯВВ, шт	4

2. Совместимость со сторонним программным обеспечением:

- **Cadence Encounter RTL Compiler, Cadence Genus Synthesis Solution, Cadence Spectre Simulation Platform;**
- **Synopsys Design Compiler, Synopsys HSPICE;**
- **Yosys Open SYnthesis Suite;**
- **Icarus Verilog;**
- **OSS CVC Simulator.**

3. Поддерживаемые языки описания аппаратуры:

- **Verilog 2005 (стандарт IEEE 1364-2005).**

4. Системные требования:

- **Операционная система Windows 7 или выше;**
- **Операционная система Red Hat 6 \ Cent OS 6 или выше;**
- **Оперативная память: 8 Гб;**
- **Свободное место на жестком диске: 4 Гб;**
- **Наличие видео драйвера с поддержкой спецификации OpenGL 2.0.**

МАРШРУТ ПРОЕКТИРОВАНИЯ НА ОСНОВЕ ПЛИС

Программные средства X-CAD обеспечивают возможность интеграции с существующим коммерческим ПО (при наличии) для выполнения логического синтеза и моделирования общего назначения от компаний Cadence и Synopsys, а также со свободно распространяемым ПО Yosys Open Synthesis Suite. На рисунке 1 изображен маршрут, обеспечивающий интеграцию средств САПР компании Cadence:

- 1) загрузка файла библиотеки стандартных элементов ПЛИС и HDL описания проектируемой пользовательской схемы в программу логического синтеза RTL Compiler / Genus;
- 2) полученное схемное описание передается в подпрограмму САПР для технологического отображения, размещения и трассировки, после чего происходит проверка проекта на быстродействие с помощью выполнения статического временного анализа (СВА) и моделирования схемы в симуляторе Cadence Spectre;
- 3) в результате работы САПР генерируется файл прошивки ПЛИС.

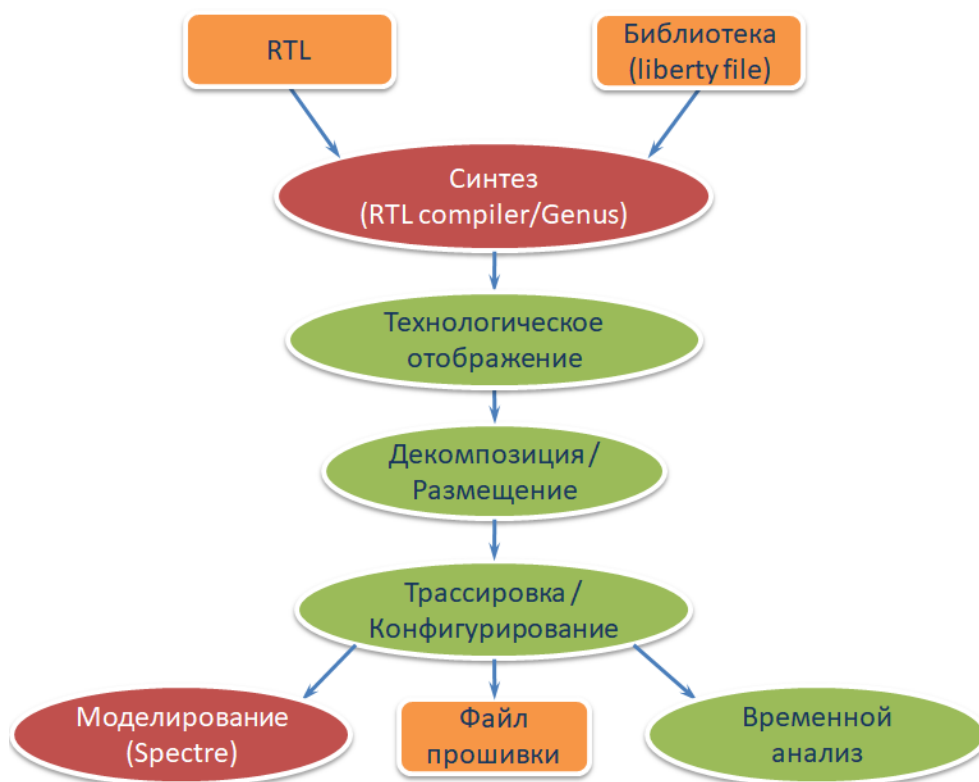


Рисунок 1 – Маршрут проектирования на ПЛИС с использованием ПО Cadence

На рисунке 2 приведен аналогичный маршрут, обеспечивающий интеграцию с САПР компании Synopsys:

- 1) загрузка файла библиотеки стандартных элементов ПЛИС и HDL описания проектируемой пользовательской схемы в программу логического синтеза Synopsys Design Compiler;
- 2) полученное схемное описание передается в подпрограмму САПР для технологического отображения, размещения и трассировки, после чего происходит проверка проекта на быстродействие с помощью выполнения статического временного анализа и моделирования схемы в симуляторе HSpice;
- 3) в результате работы САПР генерируется файл прошивки.

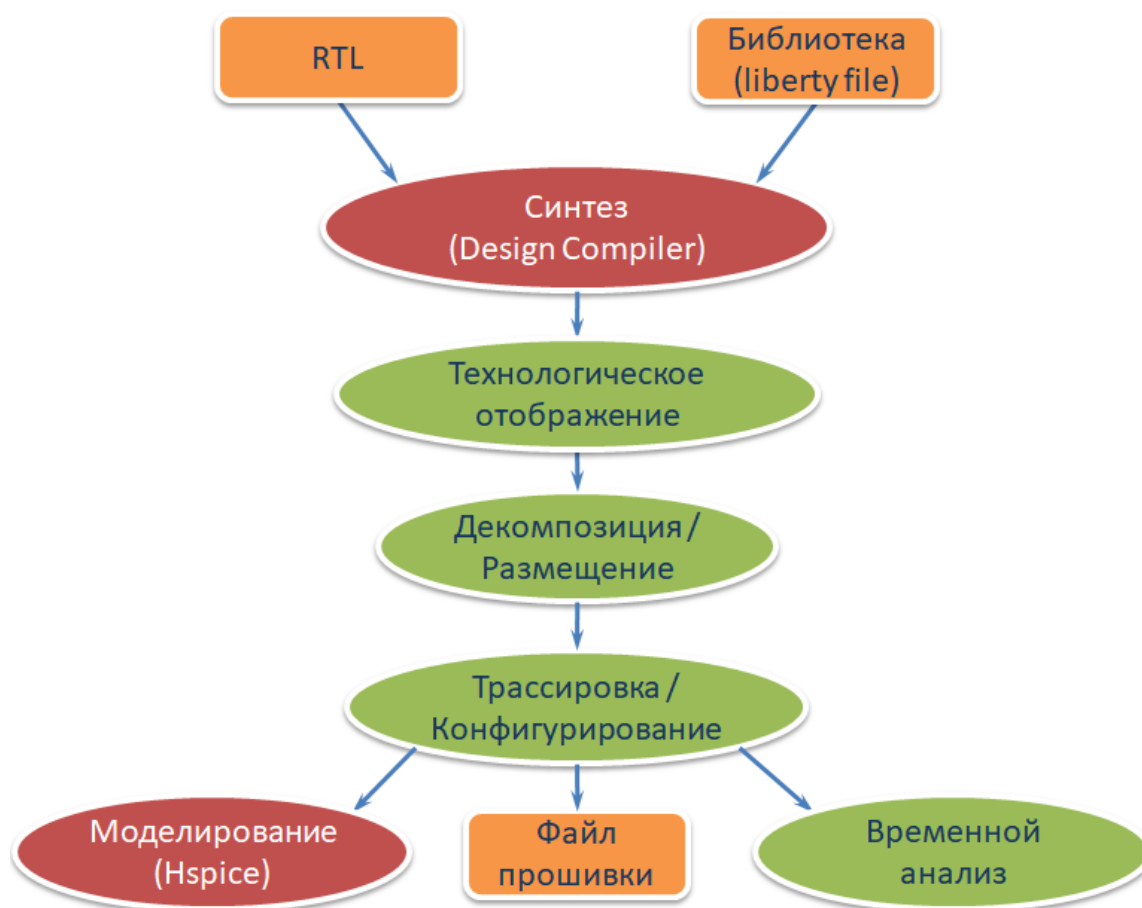


Рисунок 2 – Маршрут проектирования на ПЛИС с использованием ПО Synopsys

На рисунке 3 приведен маршрут, обеспечивающий интеграцию со свободно распространяемым ПО Yosys Open SYnthesis Suite:

- 1) загрузка файла библиотеки стандартных элементов ПЛИС и HDL описания пользовательской схемы в программу логического синтеза Yosys;
- 2) полученное схемное описание передается в подпрограмму САПР для технологического отображения, размещения и трассировки, после чего происходит проверка проекта на быстродействие с помощью выполнения статического временного анализа и моделирования схемы в симуляторе Icarus Verilog;
- 3) в результате работы САПР генерируется файл прошивки.

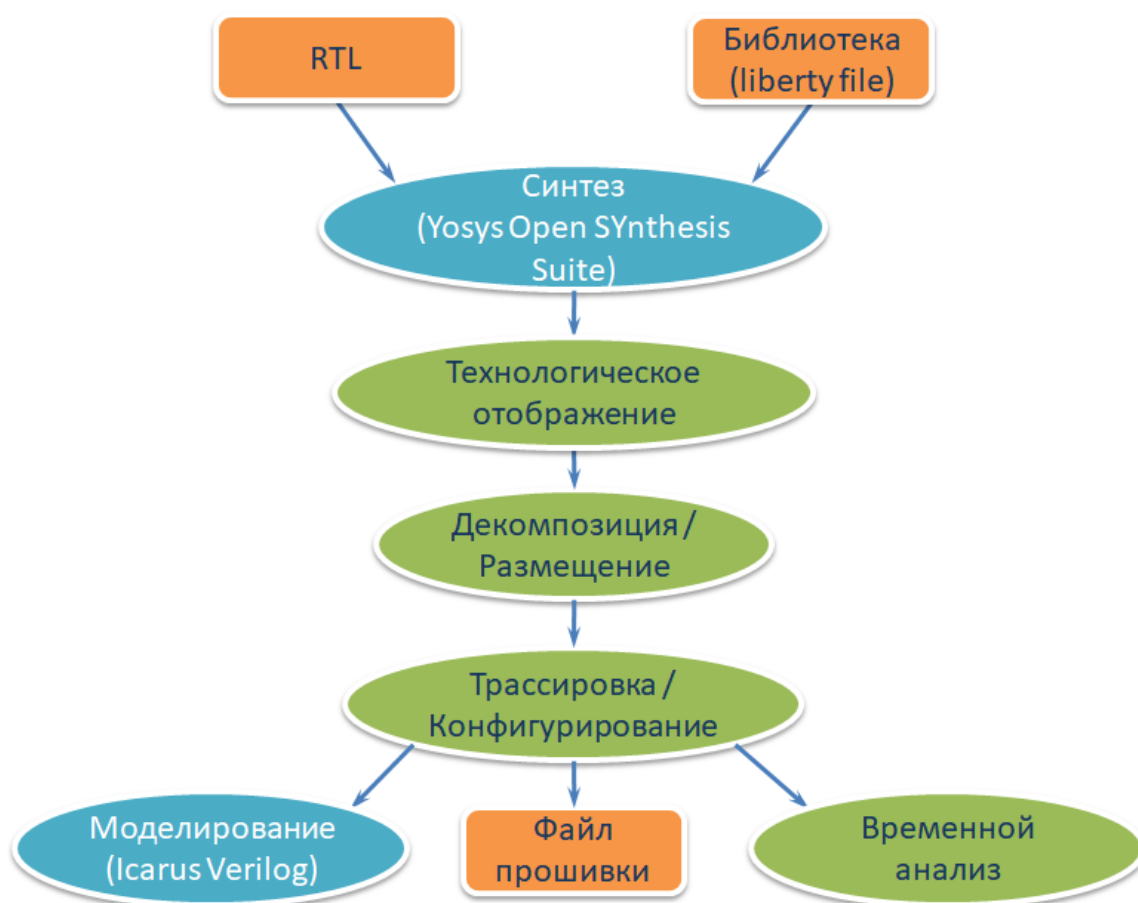


Рисунок 3 – Маршрут проектирования на ПЛИС с использованием ПО Yosys

СТРУКТУРА ПРОГРАММНЫХ ДИРЕКТОРИЙ

Структура директории программы X-CAD:

X-CAD *# домашняя директория*

 bin *# директория исполняемых модулей*

 xcore.exe *# модуль для запуска XCORE*

 xcad.exe *# модуль для запуска XCAD, графического
 интерфейса САПР*

 xplace.exe *# модуль для запуска XPLACE*

 xmap.exe *# модуль для запуска XMAP*

 xsta.exe *# модуль для запуска XSTA*

 src *# директория с исходными файлами*

 /data *# файлы конфигурации ПЛИС*

 /libraries *# библиотеки для работы с ПЛИС*

 /scripts *# управляющие скрипты*

 /docs *# документация*

 lib *# библиотеки tcl/tk*

 tools *# программные модули Yosys, IcarusVerilog, GTK Wave,
 CVC*

ЗАПУСК ПРОГРАММЫ В КОНСОЛЬНОМ РЕЖИМЕ

Консольный режим программы выполняется с помощью управляющего скрипта на языке Tcl. Команда запуска из системной консоли имеет следующий вид:

```
<PATH>/bin/xcore.exe <PATH>/src/scripts/<script> -i <ckt_file> <options> ,
```

где:

<PATH> – домашняя директория САПР X-CAD,

<script> – управляющий скрипт на языке Tcl, обеспечивающий запуск главного исполняемого модуля, согласование входных и вывод выходных данных,

<ckt_file> – полное имя файла с описанием проектируемой схемы,

<options> – опции запуска программы.

Каждая ПЛИС имеет свой управляющий скрипт, адаптированный под ее архитектуру и схемотехнические особенности. Соответствие названия ПЛИС и управляющего скрипта приведено ниже:

5400TC015 – ха3

Основные опции запуска программы:

| | |
|---|--|
| -h | вывод на экран краткой справки о программе X-CAD |
| -i <ckt_file> | имя входного проектного файла формата Verilog или Tcl |
| -o <out_name> | задание имен выходных файлов |
| -top <top_name> | определение имени модуля верхнего уровня. Задается в случае различия имени входного проектного файла и модуля верхнего уровня. |
| Управление процессом логического синтеза | |
| -rtl | синтез с помощью программы Cadence RTL Compiler |
| -gen | синтез с помощью программы Cadence Genus |

| | |
|---|---|
| -dc | синтез с помощью программы Synopsys Design Compiler |
| -y++ | синтез с помощью программы Yosys |
| Управление процессом компоновки и размещения | |
| -p <file> | загрузка размещения элементов проектной схемы из заданного файла; если размещение содержится вместе с нетлистом в Tcl-файле <ckt_file>, следует указать -p "" (пустые кавычки) |
| -inout <file> | загрузка размещения ячеек ввода-вывода (ЯВВ) из заданного файла |
| -macro <file> | загрузка размещения макроячеек из заданного файла |
| -ip_place <file> | загрузка размещения IP блоков из заданного файла |
| -le <file> | загрузка готовой планировки логических элементов из заданного файла |
| -le_union <value> | <p>Управление компоновкой логических элементов:</p> <p>0 - выключение парной компоновки ЛЭ (компоновка только по сигналу SET/RESET);</p> <p>1 - компоновка ЛЭ по входам {a d};</p> <p>2 - компоновка по всем логическим входам ЛЭ;</p> <p>3 - максимальный уровень компоновки, в том числе не связанных друг с другом ЛЭ.</p> |
| -sa_lm <value> | <p>задание множителя внутреннего цикла алгоритма размещения элементов;</p> <p>увеличение значения параметра, увеличивает длительность и качество автоматического размещения;</p> |

| | |
|---|---|
| | <value> - целое число из интервала [1..1000] |
| Управление процессом трассировки | |
| -order <file> | задание пользовательского порядка трассировки цепей проектной схемы, см. описание Tcl команды xc_net_order |
| -sr <value> | <p>-sr – задание метода перетрассировки перегруженных цепей в процессе автоматической трассировки,</p> <p><value> принимает одно из трех значений {0, 1, 2}.</p> <p>-rr – задание метода обхода трассировочных ресурсов ПЛИС,</p> <p><value> принимает одно из двух значений {0, 1}.</p> <p>Возможные комбинации значений опций:</p> <p>-sr 2 -rr 1 – по умолчанию: достигается оптимальный результат по времени и качеству трассировки;</p> <p>-sr 2 -rr 0 – оптимизация времени трассировки;</p> <p>-sr 0 -rr 1 – оптимизация объема трассировочных ресурсов, используемых для имплементации проектной схемы;</p> <p>-sr 0 -rr 0 – отсутствие оптимизации; может быть использована в случае, когда проектная схема не разводится при иных значениях</p> |
| -rr <value> | |
| -v_h <value> | <p>коэффициент учета истории перегруженности коммутационного ресурса при трассировке,</p> <p><value> - любое число из интервала [0..1]</p> |

| | |
|------------------|---|
| -v_p <value> | <p>коэффициент, отвечающий за влияние накопленного и собственного базового веса вершины графа трассировки,</p> <p><value> - любое число из интервала [0..1]</p> |
| -path_w <value> | <p>ограничение суммарной накопленной стоимости прохождения сигнала для межсоединений,</p> <p><value> - любое число из интервала [0..500]</p> |
| -path_l <value> | <p>ограничение максимальной длины межсоединения, т.е. количества пройденных трассировочных элементов,</p> <p><value> - целое число из интервала [1..300]</p> |
| -ro_fast <value> | <p>подключение предустановленных значений параметров трассировки для получения трассировочного решения за кратчайшее время,</p> <p><value> принимает одно из двух значений {0, 1}:</p> <p>-ro_fast 1 - значение “<i>включено</i>” по умолчанию.</p> <p>При включении опции заданные вручную значения параметров -sr, -rr, -v_h, -v_p, -path_w, -path_l игнорируются.</p> <p>Внимание! Опция не может быть включена вместе с опцией -ro_hq.</p> |
| -ro_hq <value> | <p>подключение предустановленных значений параметров трассировки для достижения наилучшего качества трассировочного решения,</p> <p><value> принимает одно из двух значений {0, 1}:</p> <p>-ro_hq 0 - значение “<i>выключено</i>” по умолчанию.</p> |

| | |
|---|--|
| | <p>При включении опции заданные вручную значения параметров -sr, -rr, -v_h, -v_p, -path_w, -path_l игнорируются.</p> <p>Внимание! Опция не может быть включена вместе с опцией -ro_fast.</p> |
| <i>Управление процессом трассировки: поддержка устаревших опций</i> | |
| -reroute | <p>выбор алгоритма трассировки:</p> <p>0 – алгоритм A* без разрыва и перетрассировки неразведенных цепей;</p> <p>1, 2 – алгоритм A* с двумя версиями метода разрыва и перетрассировки;</p> <p>3 – алгоритм PathFinder (включен по умолчанию)</p> |
| -ripup | <p>ограничение на количество разрывов и перетрассировок для каждой цепи (учитывается при -reroute {1,2})</p> |
| -ro_iter | <p>ограничение на суммарное количество разрывов и перетрассировок всего списка цепей (учитывается при -reroute {1,2})</p> |

ВХОДНЫЕ ДАННЫЕ ПРОГРАММЫ

Входными данными для программы может быть описание схемы на разных уровнях представления данных. Так, если **<ckt_name>** – название проектируемой схемы, то:

1. **<ckt_name>.v** – HDL описание схемы на языке Verilog – требует обязательного запуска RTL-compiler, Genus, Design Compiler или Yosys.
2. **<ckt_name>.sv** – HDL описание схемы на языке System Verilog – требует обязательного запуска RTL-compiler, Genus, Design Compiler или Yosys.
3. **<ckt_name>.vhd** – HDL описание схемы на языке VHDL – требует обязательного запуска RTL-compiler, Genus, Design Compiler или Yosys.
4. **<ckt_name>.syn_*.v** – синтезированное Verilog описание схемы с использованием библиотечных и периферийных элементов ПЛИС. Файл может быть создан вручную или получен в результате синтеза с помощью RTL-compiler, Genus, Design Compiler, Yosys.
5. **<ckt_name>.xmap.tcl** – описание схемы на языке Tcl с использованием библиотечных элементов ПЛИС (является также одним из выходных файлов работы программы при использовании HDL описания в качестве исходного), как правило, требует наличия файла **<ckt_name>.xmap.lib.tcl**.

Пример: c17.xmap.tcl, фрагмент

```
#INPUT LIST
```

```
set xc_input { N1 N2 N3 N6 N7 }
```

```
#OUTPUT LIST
```

```
set xc_output { N22 N23 }
```

```
#INPUT IOs
```

```
xc_inst {XC_BUF_N1} ibuf {a=N1 x=XC_I_N1} {inp_io=1}
```

```
xc_inst {XC_BUF_N2} ibuf {a=N2 x=XC_I_N2} {inp_io=1}
```

```
xc_inst {XC_BUF_N3} ibuf {a=N3 x=XC_I_N3} {inp_io=1}
```

```
xc_inst {XC_BUF_N6} ibuf {a=N6 x=XC_I_N6} {inp_io=1}
```

```
xc_inst {XC_BUF_N7} ibuf {a=N7 x=XC_I_N7} {inp_io=1}
```

#OUTPUT IOs

```
xc_inst {XC_BUF_N22} obuf {a=XC_O_N22 x=N22} {out_io=1}
```

```
xc_inst {XC_BUF_N23} obuf {a=XC_O_N23 x=N23} {out_io=1}
```

#LUTs

```
xc_inst {_1_} LE_cell_i3_11 { d=_0_ c=XC_I_N3 b=XC_I_N1  
y=XC_O_N22 }
```

```
xc_inst {_2_} LE_cell_i4_107 { d=XC_I_N7 c=XC_I_N2 b=XC_I_N6  
a=XC_I_N3 y=XC_O_N23 }
```

```
xc_inst {_3_} LE_cell_i3_11 { d=XC_I_N2 c=XC_I_N6 b=XC_I_N3  
y=_0_ }
```

6. <ckt_name>.inout.tcl – файл размещения периферийных площадок ЯВВ.

Подробное описание и примеры использования представлены в разделе
«[Использование файла размещения контактных площадок](#)»

7. <ckt_name>.place.tcl – файл размещения логических элементов, содержащий команды на языке Tcl вида:

```
xc_map_inst <имя элемента> <имя посадочной площадки>
```

Пример:

```
xc_map_inst cell_1 ILAB0101.ILE0701.ILE1
```

```
xc_map_inst cell_2 ILAB0101.ILE0801.ILE1
```

```
...
```

```
xc_map_inst cell_N ILAB0101.ILE1201.ILE2
```

8. **<ckt_name>.le.tcl** – файл компоновки логических ячеек в логические элементы на языке Tcl вида:

```
xc_le <имя ЛЭ> {<имя ячейки> < имя ячейки >}
```

Пример:

```
xc_le LE_0 { cell_1 cell_3 }
```

```
xc_le LE_1 { cell_2 }
```

...

```
xc_le LE_N { cell_N }
```

БИБЛИОТЕКИ ЭЛЕМЕНТОВ ПЛИС

Библиотеки элементов ПЛИС расположены в соответствующих поддиректориях **X-CAD/src/libraries/<chip>/<lib_case>**, где **<chip>** определяется в соответствии с таблицей:

5400TC015 – **ХАЗ**

<lib_case> – один из доступных углов характеристики:

- **PwcV1.65T125** – худший случай (напряжение – 1.65В, температура – 125°C)
- **PtcV1.80T25** – стандартный (напряжение – 1.8В, температура – 25°C)
- **PbcV1.95Tm40** – лучший случай (напряжение – 1.95В, температура – 40°C)

Библиотеки представлены в следующих форматах:

- ***.tcl** – описание конфигурирования элементов ПЛИС.
- ***.lib** – библиотека элементов в стандарте Liberty.
- ***.v** – библиотека элементов в формате Verilog.

Специальные обозначения используемые в названиях библиотек:

- **LE** – библиотека логических элементов ПЛИС.
- **IO** – библиотека элементов ввода-вывода ПЛИС.
- **RE** – библиотека коммутационных элементов ПЛИС.

ИСПОЛЬЗОВАНИЕ КОНТАКТНЫХ ПЛОЩАДОК

Автоматизация представленного маршрута проектирования подразумевает автоматическую генерацию экземпляров модулей ячеек ввода-вывода на основании списков имен входных и выходных портов Verilog-описания схемы.

Уникальное *имя*, присваиваемое *экземпляру ячейки*, имеет вид:

$$XC_BUF_<имя\ порта>$$

У входной ячейки имя внутреннего сигнала, входящего в ПЛИС, принимает вид:

$$XC_I_<имя\ порта>$$

У выходной ячейки имя внутреннего сигнала, приходящего из ПЛИС, принимает вид:

$$XC_O_<имя\ порта>$$

У входной и выходной ячеек, предназначенных для тактового сигнала, имя внутреннего сигнала принимает вид:

$$XC_C_<имя\ порта>$$

Внешние сигналы ячеек ввода/вывода, связанные с периферией, сохраняют имя соответствующего им порта из Verilog-описания устройства.

Пример автоматически сгенерированных экземпляров модулей входной, выходной и тактовых ячеек:

```
xa_ib XC_BUF_G0 (.a(XC_I_G0), .x(G0));  
xa_ob XC_BUF_G17 (.a(G17), .x(XC_O_G17));  
xa_clk_ib XC_BUF_CK (.a(XC_C_CK), .x(CK));  
xa_clk_ob XC_BUF_CK_out (.a(CK_out),  
.x(XC_C_CK_out));
```

Экземпляр модуля ячейки ввода/вывода может быть добавлен вручную в соответствии с тем стандартом языка аппаратуры, который поддерживается

используемым программным обеспечением для выполнения логического синтеза.

Пример добавленных вручную экземпляров модулей входной и выходной ячеек:

```
xa_ib u1 (.a(G0), .x(G0_int));  
xa_ob u2 (.a(G17_int), .x(G17));
```

Пример добавленных вручную экземпляров модулей входной и выходной тактовых ячеек:

```
xa_clk_ib clk1 (.a(CK), .x(CK_int));  
xa_clk_ob clk2 (.a(CK_int), .x(CK_out));
```

Использование выходной тактовой ячейки (как автоматически сгенерированной, так и добавленной пользователем) возможно только в том случае, когда выходной тактовый сигнал повторяет входной (когда выходному тактовому сигналу назначается входной).

Данная ситуация показана на следующем **примере**:

```
module clk_in_out(clk_in, clk_out, rst_n, data, out);  
  input clk_in, rst_n, data;  
  output clk_out, out;  
  wire clk_in_int, clk_out_int;  
  
  xa_clk_ib clk1 (.a(clk_in), .x(clk_in_int));  
  xa_clk_ob clk2 (.a(clk_out_int), .x(clk_out));  
  
  always @(posedge clk_in_int or negedge rst_n)  
  begin  
    if(~rst_n) begin  
      out <= 0;  
    end else begin
```

```
        out <= data;  
    end  
end  
  
    assign clk_out_int = clk_in_int;  
  
endmodule
```

Также пользователю доступно использование специализированной контактной площадки PAD_RST, позволяющей глобально осуществить сброс всех триггеров на кристалле ПЛИС.

Для назначения сигнала сброса на данную контактную площадку необходимо при разработке Verilog-описания задать этому сигналу имя «GRESET».

Пример использования контактной площадки глобального сброса:

```
module greset(clk_in, GRESET, data, out);  
    input clk_in, GRESET, data;  
    output out;  
  
    always @(posedge clk_in or posedge GRESET) begin  
        if(GRESET) begin  
            out <= 0;  
        end else begin  
            out <= data;  
        end  
    end  
end
```

```
endmodule
```

ФАЙЛ РАЗМЕЩЕНИЯ КОНТАКТНЫХ ПЛОЩАДОК

<ckt_name>.inout.tcl – файл размещения периферийных площадок ПЛИС.

Файл имеет следующий синтаксис описания:

```
set xc(inout) {  
  {<имя порта > <имя контактной площадки>}  
  ...  
  {<имя порта > <имя контактной площадки>}  
}  
set xc(inout_type) {  
  {<имя порта > <тип площадки > <пин площадки>}  
  ...  
  {<имя порта > <тип площадки > <пин площадки>}  
},
```

где *<имя порта>* - имя входного/выходного порта RTL описания схемы; *<имя контактной площадки>* - имя контактной площадки ПЛИС, на которой размещен порт, *<тип площадки>* - тип контактной площадки / имя библиотечного элемента контактной площадки; *<пин площадки>* - пин контактной площадки, к которому подключен пор

Входные/выходные порты RTL описания схемы, отсутствующие в файле **<ckt_name>.inout.tcl**, будут размещены на контактные площадки автоматически.

Пример Verilog-описания и соответствующего ему файла размещения контактных площадок:

```
module fa ( input a, b, c, output sum, carry );
```

```
wire d,e,f;
wire a_in;

xa_ib m0 (.a(a), .x(a_in));

xor(sum,a_in,b,c);
and(d,a_in,b);
and(e,b,c);
and(f,a_in,c);
or(carry,d,e,f);

endmodule
```

m0 – пример определенного вручную экземпляра ячейки ввода/вывода. Экземпляры ячеек для сигналов b, c, carry будут добавлены автоматически. Площадка для сигналов a и sum назначаются в соответствии с inout-файлом. Площадки для b, c, carry будут назначены автоматически.

```
set xc(inout) {
    { a FROM_PAD1}
    { sum TO_PAD205}
}
set xc(inout_type) {
    { a xa_ib a}
    { sum xa_ob x}
}
```

Файл <ckt_name>.inout.tcl может быть получен при помощи интерфейса планировки кристалла FloorPlanner. Подробное описание работы с данной

программой представлено в разделе [ИНТЕРФЕЙС ПЛАНИРОВКИ КРИСТАЛЛА FLOORPLANNER](#).

Файл с размещением периферийных площадок должен быть подключен к проекту с помощью опции *-inout* в [консольном](#) ИЛИ в [графическом](#) режиме.

ИСПОЛЬЗОВАНИЕ ИМЕН КОНТАКТНЫХ ПЛОЩАДОК

Автоматическую генерацию и размещение ЯВВ можно выполнить, используя в Verilog-описании схемы названия портов, совпадающие с именем контактной площадки. Имена контактных площадок заданы в соответствии с их назначением и именами внешних портов ПЛИС. Доступные контактные площадки представлены в таблицах ниже.

| Название кристалла | Имя контактной площадки | Специализированное имя контактной площадки | Тип контактной площадки |
|--------------------|-------------------------|--|-------------------------|
| 5400TC015 | IO[190:0] | | xa_ib,
xa_ob |

ВЫПОЛНЕНИЕ СТАТИЧЕСКОГО ВРЕМЕННОГО АНАЛИЗА

Запуск выполнения статического временного анализа (СВА) проектируемой схемы производится в автоматическом режиме. Для установки тактовых сигналов, определения временных ограничений и изменения других настроек СВА следует использовать опции запуска.

Основные опции запуска СВА:

| Управление процессом статического временного анализа | |
|--|---|
| -clk <list> | – определение списка тактовых сигналов в проекте |
| -per <list> | – определение значений периодов тактовых сигналов |
| -sta_rise <value> | – определение времени нарастания входного сигнала |
| -sta_fall <value> | – определение времени спада входного сигнала |
| -sta_tran <value> | – определение времени нарастания и спада входного сигнала |
| -sta_max_delay
<value> | – определение значения максимальной задержки на главных выходах схемы |
| -sta_num_crit
<value> | – определение количества критических путей, выведенных в отчет |
| -sta_sdc <file_path> | – определение полного пути до файла временными ограничениями проекта |

Результаты СВА генерируются по завершению работы программы в папку **<top_name>.STA.reports.**

Результатом проведенного СВА схемы до этапа трассировки является следующий набор файлов:

- **<top_name>.gate.nworst.txt** – временной отчет для заданного количества наиболее медленных критических путей;

- **<top_name>.gate.sdf** – файл с задержками и заданными временными ограничениями.

Результатом проведенного СВА схемы после этапа трассировки является следующий набор файлов:

- **<top_name>.routed.nworst. txt** – временной отчет для заданного количества наиболее медленных критических путей;
- **<top_name>.routed.sdf** – файл с задержками и заданными временными ограничениями.

Доступные временные ограничения для использования в SDC файле:

| | |
|------------------------|----------------------|
| create_clock | set_multicycle_path |
| set_clock_latency | set_input_transition |
| set_clock_uncertainty | set_input_delay |
| set_clock_transition | set_output_delay |
| set_propagated_clock | set_max_delay |
| create_generated_clock | set_min_delay |
| set_false_path | set_load |
| set_disable_timing | set_case_analysis |

- **create_clock** – добавить к анализу тактовый сигнал, его источник, период, время нарастания и спада сигнала.

create_clock -period <T_per> [-waveform <T_rise> <T_fall>]
<source>,

-period <T_per> – задать период задаваемого тактового сигнала.

<T_per> – время периода тактового сигнала,
задается в нс или МГц;

- `-waveform` – задать коэффициент заполнения тактового сигнала. По умолчанию, коэффициент заполнения равен 50%, что эквивалентно $\langle T_{rise} \rangle = 0$ нс, $\langle T_{fall} \rangle = (\langle T_{per} \rangle / 2)$ нс.
- $\langle T_{rise} \rangle$ $\langle T_{fall} \rangle$ – время начала нарастания и начала спада сигнала.
- `<source>` – источник создаваемого синхросигнала (порт, пин). По умолчанию тактовый сигнал будет назван по имени источника. При желании можно указать имя тактового с помощью опции `-name..`

Пример использования команды:

create_clock -period 5 [get_ports CLK1]

create_clock -period 5 -waveform {1 2}[get_ports CLK2]

Результат определения коэффициента заполнения тактового сигнала по умолчанию и с использованием опции `-waveform` показан на рисунке 4.

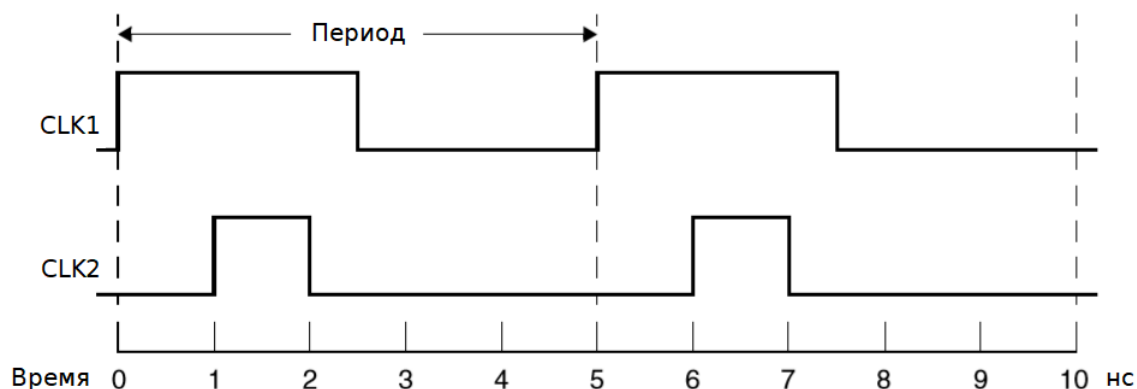


Рисунок 4 – Диаграммы тактовых сигналов

- **set_clock_latency** – установить время задержки тактового сигнала.

set_clock_latency [-rise/-fall] [-min][-max] [-source]
 [-late] [-early] <delay_value> <name>,

| | |
|---------------|---|
| -rise | – задать задержку только при нарастании тактового сигнала; |
| -fall | – задать задержку только при убывании тактового сигнала; |
| -min | – задать только минимальную задержку тактового сигнала; |
| -max | – задать только максимальную задержку тактового сигнала; |
| -source | – задать только задержку задержка источника; |
| -late | – задать только позднее время прибытия от источника; |
| -early | – задать только раннее время прибытия от источника; |
| <delay_value> | – значение времени задержки; |
| <name> | – установить имя тактового сигнала, для которого задается задержка. |

Задержка `clock_latency` состоит из двух:

- `source latency` – задержка источника – время, требуемое синхросигналу, для распространения от источника до точки указанной в проекте.
- `network_latency` – задержка сети – время, требуемое синхросигналу, чтобы дойти от точки, указанной в проекте, до тактового вывода последовательностного элемента.

В данном случае задержкой источника является задержка ЯВВ, которая добавляется в проект автоматически. Её задержка определяется программой из характеризованной библиотеки.

Задержка сети определяется программой за счет оценки реальных трассировочных элементов, используемых в схеме для коммутации

межсоединений. Их задержка также передается из характеризованной библиотеки.

Пример использования команды:

```
set_clock_latency -rise -max 0.34 CLK1
```

- **set_clock_uncertainty** – установить время, на которое отклоняется тактовый сигнал от идеального при распространении на различные приемники после трассировки.

```
set_clock_uncertainty [-setup/-hold] [-rise/-fall]
```

```
-from <clk_names> -to <clk_names> <uncertainty>,
```

```
-setup           – задать погрешность только для времени  
                  установки;
```

```
-hold           – задать погрешность только для времени  
                  удержания;
```

```
-rise           – задать погрешность во время нарастания  
                  тактового сигнала;
```

```
-fall           – задать погрешность во время спада тактового  
                  сигнала;
```

```
-from           – задать список тактовых сигналов, являющихся  
<clk_names>      начальными точками анализируемого пути;
```

```
-to <clk_names> – задать список тактовых сигналов, являющихся  
                  конечными точками анализируемого пути;
```

```
<uncertainty>  – задать значение погрешности.
```

В данном случае данная погрешность определяется программой за счет оценки задержки реальных трассировочных элементов из характеризованной библиотеки.

Пример использования команды:

```
set_clock_uncertainty -rise -from CLK1 -to CLK2 0.34
```

- **set_clock_transition** – установить время переключения тактового сигнала.

```
set_clock_transition [-rise/-fall] <transition_time>,
```

```
<clock_name>
```

```
-rise                – задать только время нарастания тактового  
                      сигнала;
```

```
-fall                – задать только время спада тактового сигнала;
```

```
<transition_time>   – задать значение времени переключения;
```

```
e>
```

```
<clock_name>        – задать список тактовых сигналов, которым будет  
                      установлено заданное значение времени  
                      переключения.
```

Пример использования команды:

```
set_clock_transition -rise 0.15 CLK1
```

- **set_propagated_clock** – распространить задержки тактового сигнала и автоматически определить задержку на каждом тактовом пине регистра с учетом добавленных паразитных элементов.

```
set_propagated_clock <clock_name>,
```

```
<clock_name>        – задать список тактовых сигналов..
```

Пример использования команды:

```
set_propagated_clock CLK1
```

- **create_generated_clock** – добавить к анализу тактовый сигнал, производный от основного тактового сигнала, деленный, помноженный или инвертированный внутренним элементом схемы.

```
create_generated_clock    -name      <clock_name>    -source  
<master_name>
```

```
[-divide_by divide_factor | -multiply_by multiply_factor]
```

```
[-duty_cycle percent] [-invert] <source>,
```

-name <clk_name> – имя создаваемого синхросигнала;

-source
<master_name> – имя источника основного синхросигнала
(порт, пин);

-divide_by factor – коэффициент деления сигнала;

-multiply_by
factor – коэффициент умножения сигнала;

-duty_cycle
percent – коэффициент заполнения в процентах;

-invert – инверсия, создаваемого синхросигнала;

<source> – источник создаваемого синхросигнала (порт,
пин);

Например, рассмотрим схему изображенную на рисунке 5, выполняющая деление тактового сигнала на 3.

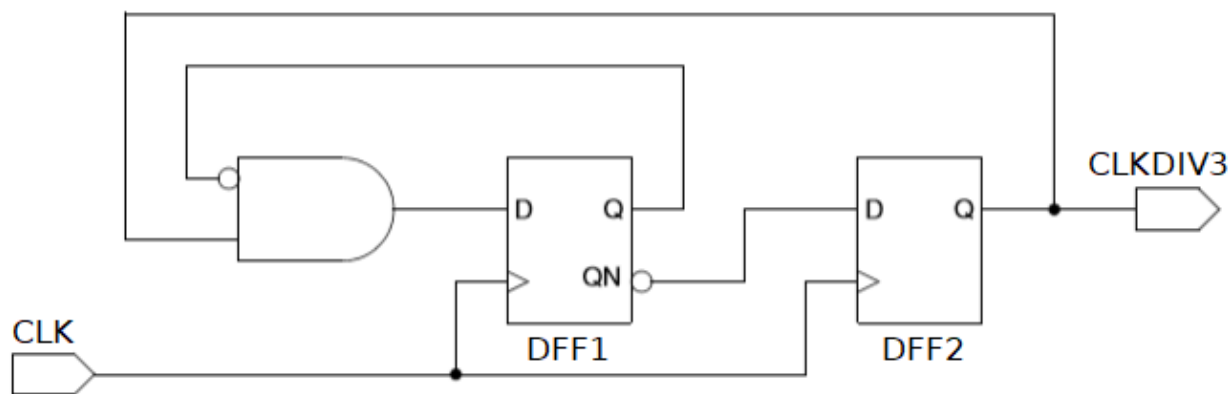


Рисунок 5 – Схема деления тактового сигнала

Основной тактовый сигнал CLK и производный от него CLKDIV3 задаются программе следующим образом:

```
create_clock period 2 -name CLK [get_ports CLK]
```

```
create_generated_clock -name CLKDIV3 -source [get_ports CLK]
```

```
-divide_by 3 -duty_cycle 67 [get_pins DFF2/Q]
```

Формы заданных тактовых сигналов представлены на рисунке 6.

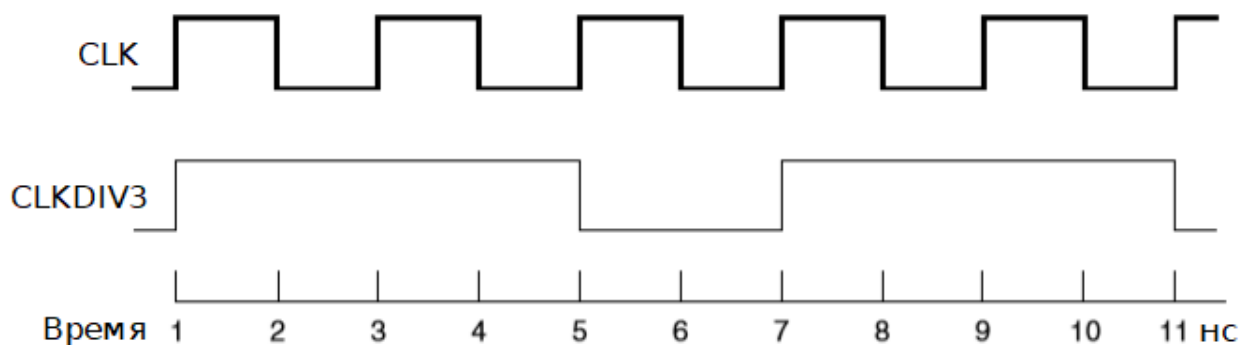


Рисунок 6 – Временные диаграммы основного и производного тактовых сигналов

- **set_false_path** – исключить заданный путь из временного анализа. Команда определяет заданные пути ложными и удаляет временные ограничения на этих путях.

Ложные временные пути - это пути, по которым не распространяется изменение логического уровня.

set_false_path [-setup/-hold] [-rise/-fall] -through {node list}

-from {node list} -to {node list},

-setup – исключить из временного анализа проверку времени установки заданного сигнала;

-hold – исключить из временного анализа проверку времени удержания заданного сигнала;

-rise – исключить из временного анализа элементы, тактирующиеся нарастающим фронтом заданного тактового сигнала;

-fall – исключить из временного анализа элементы, тактирующиеся убывающим фронтом заданного тактового сигнала;

-through {list} – задать список пинов, портов или ячеек, через которые должны проходить ложные пути;

-from {list} – задать список начальных точек исключаемого временного пути – тактовые сигналы, входные порты, входные тактовые пины последовательностных ячеек;

-to {list} – задать список конечных точек исключаемого временного пути – тактовые сигналы, выходные порты, входные пины данных последовательностных ячеек;

Пример использования команды:

set_false_path -to [get_clocks CLK]

Данная команда объявляет пути, тактируемые в конечных точках сигналом CLK, на рисунке 7 ложными, т.е. исключает из временного анализа все пути.

```
set false path -fall -to [get clocks CLK]
```

Данная команда объявляет пути, тактируемые в конечных точках убывающим фронтом сигнала CLK, на рисунке 7 ложными, т.е. исключает из временного анализа пути В и С.

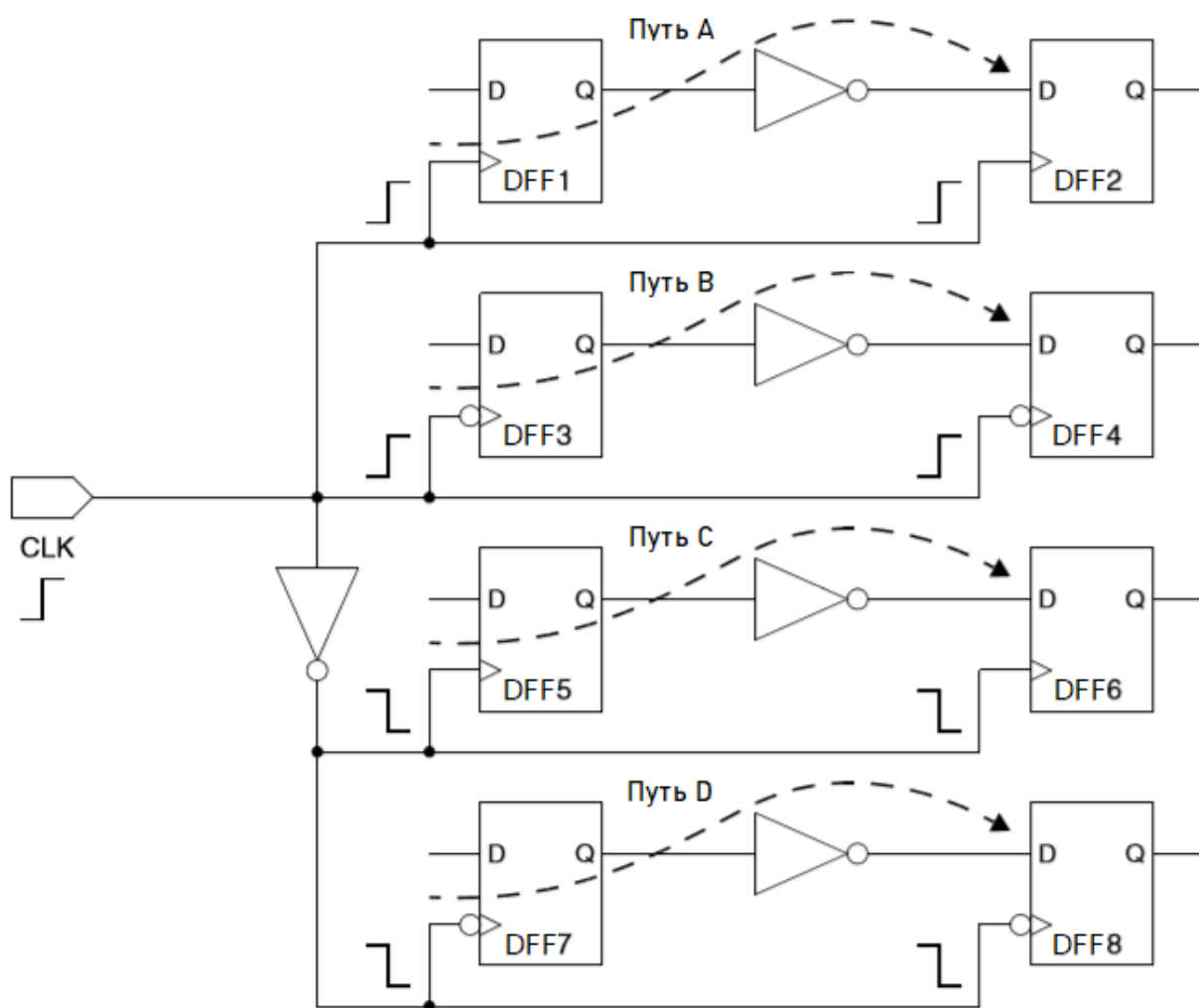


Рисунок 7 – Схема, поясняющая использование ложных путей

- **set_disable_timing** – удалить из временного анализа все объекты (ячейки, пины или порты) на заданном пути.

Эта команда полностью удаляет из временного анализа встретившиеся на пути объекты, в отличие от команды `set_false_path`, которая исключает только конкретный путь, а не вычисление задержки объектов этого пути. Если все пути через определенный пин – ложные, то применение `set_disable_timing` более эффективно, чем `set_false_path`.

set_disable_timing `{[-from pin_name] | [-to pin_name]}`

`<cell_list>`,

`-from pin_name` – задать начальный пин исключаемого временного пути;

`-to pin_name` – задать конечный пин исключаемого временного пути;

Для работы команды необходимо задать одновременно как пин начала пути, так и пин, показывающий конец пути.

`<cell_list>` – задать список ячеек, удаляемых из временного анализа для заданного пути;

Пример использования команды:

set_disable_timing `-from DFF1/D -to DFF2/Q {DFF1 DFF2}`

- **set_multicycle_path** – установить путь, занимающий несколько тактов, для проверки времени установки и времени удержания.

set_multicycle_path ncycles [-setup/-hold] [-rise/-fall]

-through {node list} -from {node list} -to {node list},

ncycles – количество циклов тактового сигнала, которое требуется данным для проверки времени установки и удержания;

-setup – изменить количество циклов тактового сигнала только для проверки времени установки;

-hold – изменить количество циклов тактового сигнала только для проверки времени удержания. По умолчанию время удержания проверяется на ncycles - 1 такте;

-rise – изменить количество циклов нарастания тактового сигнала;

-fall – изменить количество циклов убывания тактового сигнала;

-through {list} – задать список пинов, портов или ячеек, через которые проходит устанавливаемый путь;

-from {list} – задать список начальных точек устанавливаемого пути – тактовые сигналы, входные порты, входные тактовые пины последовательностных ячеек;

-to {list} – задать список конечных точек устанавливаемого пути – тактовые сигналы,

выходные порты, входные пины данных
последовательностных ячеек;

Пример использования команды:

```
set_multicycle_path 4 -setup -from [get_ports CLK]
```

- **set_input_transition** – установить время переключения на входах схемы.

```
set_input_transition <value> <input name list> [r / f],
```

<value> – значение времени переключения;

<input name list> – список входов, которым будет присвоено
заданное значение;

r – установить время нарастания сигнала на
заданных входах;

f – установить время спада сигнала на
заданных входах;

По умолчанию, если не заданы опции r/f,
заданное время переключения установится
как для нарастания, так и для спада.

Пример использования команды:

```
set_input_transition 1.5 [get_inputs]
```

- **set_input_delay** – установить время задержки прибытия входного сигнала
относительно основного тактового сигнала.

```
set_input_delay <input_delay value> <net name list>
```

```
[-clock <related clock>],
```

<input_delay value> – значение времени задержки;

<net name list> – задать список входов, которым будет присвоено заданное значение;

-clock <related clock> – установить имя тактового сигнала, относительно которого приходит входной сигнал;

Пример использования команды:

```
set_input_delay 0.344 {IN1 IN2} [-clock CLK]
```

- **set_output_delay** – установить время задержки прибытия выходного сигнала до последовательностной логики вне проекта относительно основного тактового сигнала.

```
set_output_delay <output delay value> <net name list>,  
[<related clock>],
```

<output_delay value> – значение времени задержки;

<net name list> – задать список входов, которым будет присвоено заданное значение;

-clock <related clock> – установить имя тактового сигнала, относительно которого приходит выходной сигнал.

Пример использования команды:

```
set_output_delay 0.7 {OUT} [-clock CLK]
```

- **set_max_delay** – задать максимальное время задержки сигнала для временных путей, заканчивающихся в заданном списке выходов.

```
set_max_delay <delay_value> <name list> [r / f],
```

<delay_value> – значение времени задержки;

- | | |
|--------------------------------|--|
| <code><name list></code> | – список выходов, которым будет присвоено заданное значение; |
| <code>r</code> | – установить максимальную задержку для нарастающего сигнала; |
| <code>f</code> | – установить максимальную задержку для убывающего сигнала; |
- По умолчанию, если не заданы опции `r/f`, заданное время задержки установится как для нарастающего, так и для убывающего сигнала.

Пример использования команды:

```
set_max_delay 10.5 [get_outputs] r
```

- **set_min_delay** – задать минимальное время задержки сигнала для временных путей, заканчивающихся в заданном списке выходов.

```
set_min_delay <delay value> <name list> [r / f],
```

- | | |
|----------------------------------|--|
| <code><delay_value></code> | – значение времени задержки; |
| <code><name list></code> | – список выходов, которым будет присвоено заданное значение; |
| <code>r</code> | – установить максимальную задержку для нарастающего сигнала; |
| <code>f</code> | – установить максимальную задержку для убывающего сигнала; |
- По умолчанию, если не заданы опции `r/f`, заданное время задержки установится как для нарастающего, так и для убывающего сигнала.

Пример использования команды:

```
set_min_delay 6.5 [get_outputs] r
```

- **set_load** – задать дополнительную емкость списку цепей.

```
set_load <cap_value> <names_list> replace,
```

<cap_value> – значение емкости;

<names_list> – задать список цепей, которым будет установлена
 дополнительная емкость;

replace – заменить существующую на заданных цепях
 емкость. По умолчанию, значение, установленное
 данной командой, добавляется к существующей
 емкости, а не заменяет её;

Пример использования команды:

```
set_load 0.5 [get_outputs] replace
```

- **set_case_analysis** – задать постоянное логическое значение сигнала или нарастающий/убывающий фронт и исключить из временного анализа связанный с ними путь.

```
set_case_analysis {0 | 1 | zero | one | rise | fall} <objects_list>,
```

0 / zero – задать порту или пину значение «0»;

1 / one – задать порту или пину значение «1»;

rise – задать порту или пину постоянный
 нарастающий сигнал;

fall – задать порту или пину постоянный
 убывающий сигнал;

`<objects_list>` – задать список портов или пинов, для которых будет задано постоянное значение;

Функции, позволяющие получить списки имен по требуемому критерию

`get_inputs` – получить список входов;
`get_outputs` – получить список выходов;
`get_nodes` – получить список всех цепей;
`get_nodes <internal>` – получить список внутренних цепей;
`get_cells` – получить список всех ячеек;

Функции, возвращающие переданные ей значения. Необходимы для совместимости с Synopsys SDC.

`get_clocks` – получить список созданных тактовых сигналов;
`get_pins` – получить список всех пинов;
`get_ports` – получить список всех портов;
`get_lib_cells` – получить список всех библиотечных ячеек;

ВЫХОДНЫЕ ДАННЫЕ ПРОГРАММЫ

- `<ckt_name>.log` – протокол выполнения программы.
- `<ckt_name>.run_rtl.tcl` – файл с командами выполнения логического синтеза.
- `<ckt_name>.syn_y.v` – синтезированное описание схемы на языке Verilog с использованием библиотечных элементов ПЛИС, полученное с помощью Yosys.
- `<ckt_name>.xmap.tcl` – описание схемы на языке Tcl с использованием библиотечных и периферийных элементов. В схему добавляются

периферийные элементы: *ха_ib*, *ха_ob*. К именам входов добавляется префикс *XC_I_<input_name>*. К именам выходов добавляется префикс *XC_O_<output_name>*, составляются списки входных и выходных узлов *xc_inputs*, *xc_outputs*, а также список тактовых узлов *xc_clocks*.

- **<ckt_name>.xmap.lib.tcl** – вывод библиотеки элементов в Tcl-формате;
- **<ckt_name>.xmap.glib.v** – вывод библиотеки дополнительных элементов в Verilog-формате.
- **<ckt_name>.gate.v** – вывод списка соединений в терминах элементов библиотеки ПЛИС в Verilog-формате.
- **<ckt_name>.gate.flat.v** – вывод списка соединений в терминах элементов библиотеки ПЛИС с раскрытием иерархии в Verilog-формате.
- **<ckt_name>.routed.v** – вывод списка соединений с результатами трассировки в терминах элементов библиотеки ПЛИС в Verilog-формате.
- **<ckt_name>.routed.flat.v** – вывод списка соединений с результатами трассировки в терминах элементов библиотеки ПЛИС с раскрытием иерархии в Verilog-формате.
- **<ckt_name>.le.tcl** – вывод результатов парной компоновки логических элементов на языке Tcl, можно использовать как входной файл.
- **<ckt_name>.map.tcl** – вывод результатов размещения и трассировки в Tcl-формате;
- **<ckt_name>.place.tcl** вывод результатов размещения на языке Tcl, можно использовать как входной файл при повторном запуске программы.
- **<ckt_name>.ro.map.tcl** – вывод результатов трассировки в Tcl-формате с детализацией отображения цепей в списки соединений элементов ПЛИС;
- **<ckt_name>.geom.gds** – вывод результатов размещения и трассировки в формате GDS-II;
- **<ckt_name>.il** – вывод результатов трассировки в Skill-формате, для использования в редакторе Virtuoso Cadence.
- **<ckt_name>.mem** – вывод результатов конфигурирования ПЛИС в терминах конфигурации памяти;

- **<ckt_name>.bitnum** – прошивка в терминах программируемых бит памяти.
- **<ckt_name>.scs** – вывод результатов конфигурирования ПЛИС в Spice-формате для моделирования Spectra.
- **<ckt_name>.xsp** – вывод результатов конфигурирования ПЛИС в Spice-формате для моделирования в Xspice.
- **<ckt_name>.xspc** – вывод результатов конфигурирования ПЛИС в Spice-формате с добавлением паразитных конденсаторов для моделирования в Xspice.
- **<ckt_name>.sta.results/** – директория с результатами статического временного анализа.

ПРИМЕРЫ ЗАПУСКА ПРОГРАММЫ В КОНСОЛЬНОМ РЕЖИМЕ

Далее описаны разные варианты запуска управляющего скрипта:

1. Переход к рабочей директории

```
cd X-CAD/test
```

2. Запуск с использованием Yosys

```
../bin/xcore.exe ../src/scripts/xa -i <ckt_name>.v -y++
```

Например:

```
../bin/xcore.exe ../src/scripts/xa -i ../src/test_isc3/c432.v -y++
```

3. Синтезированное Verilog описание схемы

```
../bin/xcore.exe ../src/scripts/xa -i <ckt_name>.syn_y.v
```

Например:

```
../bin/xcore.exe ../src/scripts/xa -i ./c432.syn_y.v
```

4. Tcl описание схемы

```
../bin/xcore.exe ../src/scripts/xa -i <ckt_name>.tcl
```

Например:

```
../bin/xcore.exe ../src/scripts/xa -i c432.xmap.tcl
```

5. С использованием готового размещения

```
../bin/xcore.exe ../src/scripts/xa -i <ckt_name>.v  
-p <ckt_name>.place.tcl
```

Например:

```
../bin/xcore.exe ../src/scripts/xa -i c432.syn_y.v  
-p ./c432.place.tcl
```

6. Описание схемы на языке Tcl и размещение находятся в одном файле

```
../bin/xcore.exe ../src/scripts/xa -i <ckt_name>.tcl -p ""
```

Например:

```
../bin/xcore.exe ../src/scripts/xa -i c432.syn_y.tcl -p ""
```

TCL ИНТЕРФЕЙС

Tcl интерфейс управления программой

xc_init – инициализация работы с программой.

xc_quit – завершение работы с программой.

xc_set_param – установка параметров программы, синтаксис:

xc_set_param <имя параметра> <значение параметра>

xc_ckt_init – инициализация новой схемы, синтаксис:

xc_ckt_init <имя схемы> PROJECT

xc_ckt_set_gnd – назначение узла земли схемы, синтаксис:

xc_ckt_set_gnd <узел земли>

xc_ckt_set_vdd – назначение узла питания схемы, синтаксис:

xc_ckt_set_vdd <узел питания>

xc_net_order high – Задает порядок трассировки узлов с высоким приоритетом, синтаксис:

xc_net_order high {<list_nodes_names>}

где *<list_nodes_names>* — список узлов для трассировки в порядке убывания приоритета.

Пример:

xc_net_order high { net1 net2 input_net }

Файл, содержащий данную команду может быть загружен при запуске исполняемого скрипта с помощью опции **-order “filename”**.

Tcl интерфейс для отображения результатов RTL синтеза в базис ПЛИС

xc_lib_cell – добавление экземпляра модели элемента в библиотеку.

Синтаксис команды **xc_lib_cell**:

xc_lib_cell <lib_cell_name> <subckt_name> {<input nodes>}

{<control_nodes>} {<function>},

где:

| | |
|-----------------|---|
| <lib_cell_name> | - имя библиотечной ячейки; |
| <subckt_name> | - имя подсхемы из spice описания ПЛИС (формат cdl); |
| <input_nodes> | - список входных и выходных узлов в формате:
<io_pin_name>=<port_list>, где: <ul style="list-style-type: none"> • <io_pin_name> — имя контакта в соответствии с .lib библиотекой • <port_list> — список узлов контакта в соответствии со spice описанием схемы ПЛИС (знак & вместо = позволяет объединять узлы для реализации элементов xor или mux) |
| <control_nodes> | - список управляющих узлов с установленными значениями программирования; |
| <function> | - логическая функция библиотечной ячейки. |

xc_pin – добавление экземпляра контакта ввода – вывода.

Синтаксис команды **xc_pin**:

xc_pin <pinName> <direction> <nodeName>,

где:

| | |
|-------------|---|
| <pinName> | - имя добавляемого контакта |
| <direction> | - направление контакта:

i – вход

o – выход

b - двунаправленный |
| <nodeName> | - имя узла проектной схемы |

xc_inst – добавление экземпляра логического элемента;

Синтаксис команды **xc_inst**:

xc_inst <inst_name> <lib_cell_name> {<list nodes>} {<inout>}

где:

| | |
|-----------------|---|
| <inst_name> | - имя добавляемого элемента |
| <lib_cell_name> | - имя библиотечной ячейки |
| <list nodes> | - список входных и выходных узлов |
| <inout> | - идентификация IO-элемента
(inp_io=1 или out_io=1). |

xc_le – группировка логических ячеек в логические элементы. Схемотехника ПЛИС предусматривает быструю локальную связь с выхода первой ячейки на любой из входов второй, таким образом позволяя объединять элементы проектируемой схемы в пары для более быстрой передачи данных.

Синтаксис команды **xc_le**:

xc_le <LE_name> { <inst_name> <inst_name> }

где:

| | |
|-------------|---|
| <LE_name> | - имя добавляемого логического элемента |
| <inst_name> | - имя добавляемого элемента типа xc_inst |

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС X-CAD

Диалоговый интерфейс представляет собой набор программных окон и интерактивных графических элементов, позволяющих осуществлять работу с набором программ, входящих в состав пакета, поставляемого вместе с данной интерфейсной программой. Интерфейс призван наладить взаимодействие между компонентами этого пакета программ тем самым упростив работу с ним.

Данный интерфейс позволяет просто и быстро провести потоковую работу с пакетом программ для генерации прошивок целевых ПЛИС.

Программа разработана с применением кроссплатформенных средств разработки (библиотека Qt) и распространяется на платформах семейств Windows и Unix.

Запуск программы

Запуск программы может быть осуществлен как посредством активации исполняемого файла из файлового менеджера, так и из командной строки.

Запуск через обозревателя файлов

Для запуска через обозревателя файлов необходимо открыть папку с исполняемым файлом «xcad.exe» (с соответствующей иконкой ) , который расположен в директории «bin» пакета программ, и активировать двойным щелчком левой кнопки мыши или вызвать с помощью правой кнопки мыши контекстное меню и выбрать соответствующий пункт для запуска приложения.

Запуск через командную строку

Чтобы запустить программу посредством командной строки, необходимо перейти в папку «bin» пакета программ и вызвать на исполнение файл «xcad.exe», используя следующие команды:

- Windows: “start xcad.exe”
- Unix: “./xcad”

Стартовое окно

При первом запуске программа создаст в домашней директории поддиректорию «XCAD», а также следующие файлы и директории:

- *history.hst* – файл для хранения истории открытых проектов;
- *user_config.cfg* – файл пользовательских настроек программы;
- *Projects* – директория для пользовательских проектов.

Позже в директории «XCAD», также, будет создан следующий файл:

- *win_state.ws* – файл, хранящий состояние графического интерфейса на момент закрытия программы.

После того, как файлы и директории были созданы, пользователя встречает стартовое окно менеджера проектов (рисунок 8) со списком последних открытых проектов (изначально пуст), который впоследствии будет отображать список проектов со следующей информацией: «имя проекта», «последний запуск», «кристалл», «ядро», «полный путь» (до файла проекта).

Перед открытием окна программа проверяет существование файлов проектов по указанным расположениям. Если файл проекта не был найден, то соответствующая этому проекту строка в списке будет выделяться красным цветом, а иконка проекта сменится иконкой с восклицательным знаком в красном круге (). При выборе такого элемента списка, кнопки открытия проекта будут заблокированы.

Также в этом окне имеются следующие кнопки:

- «Открыть выбранный проект» – открывает выбранный из списка проект (если список пуст, то кнопка будет заблокирована);
- «Открыть существующий проект» – открывает окно обозревателя файлов для выбора файла проекта (с расширением “*.xcprj”);
- «Новый проект» – открывает окно мастера создания проекта;
- «Удалить проект» – удаляет директорию с проектом и соответствующую запись в истории (если проект не выбран, то кнопка будет заблокирована);
- Переключатель языка (в правом нижнем углу).

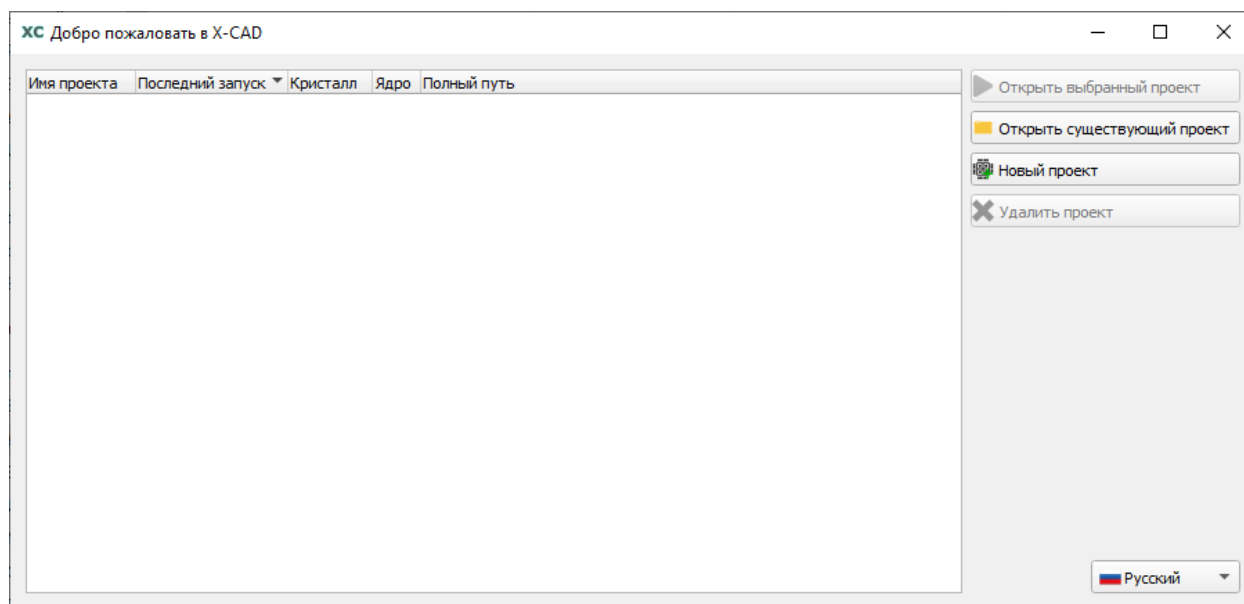


Рисунок 8 – Стартовое окно программы

Список проектов поддерживает, также, контекстное меню (рисунок 9), которое вызывается посредством щелчка правой кнопкой мыши (ПКМ) на элементе списка. Меню содержит следующие возможности:

- «Открыть» – открывает выбранный проект;
- «Открыть расположение» – открывает директорию, в которой расположен файл проекта;
- «Удалить из истории» – удаляет запись о проекте из этого окна и истории;
- «Удалить проект» – удаляет директорию с проектом и соответствующую запись в истории.

Открытие существующего проекта

Если программа запускается не первый раз и ранее были созданы/открыты проекты, то список будет содержать все из них (рисунок 9). Открыть существующий проект можно следующими способами:

- выбрать проект из списка (нажав на него ЛКМ) и нажать кнопку «Открыть выбранный проект»;
- двойным щелчком ЛКМ на проекте из списка;

- нажать на кнопку «Открыть существующий проект», после чего через открывшееся окно обозревателя файлов найти файл проекта с расширением «*.xcprj», выбрать его (ЛКМ) и нажать «Открыть», или нажать на него двойным щелчком ЛКМ.

При открытии несуществующего проекта (например, двойным щелчком ЛКМ) будет показано сообщение о том, что файл проекта (рисунок 10) больше не существует.

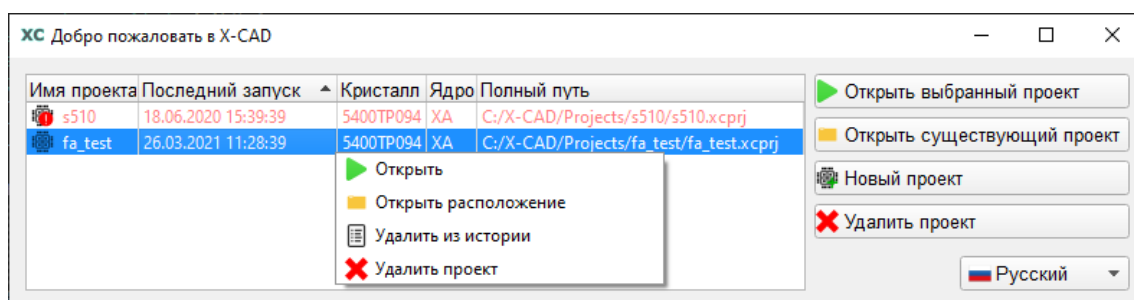


Рисунок 9 – Стартовое окно программы с элементами списка

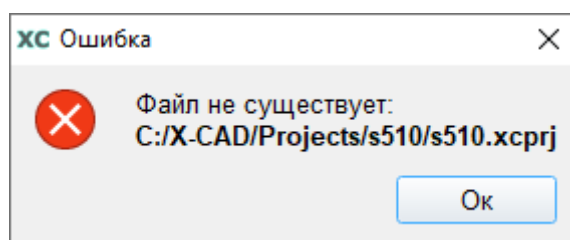


Рисунок 10 – Сообщение об ошибке при открытии несуществующего проекта

После загрузки проекта откроется главное окно программы для дальнейшей работы с проектом (рисунок 11).

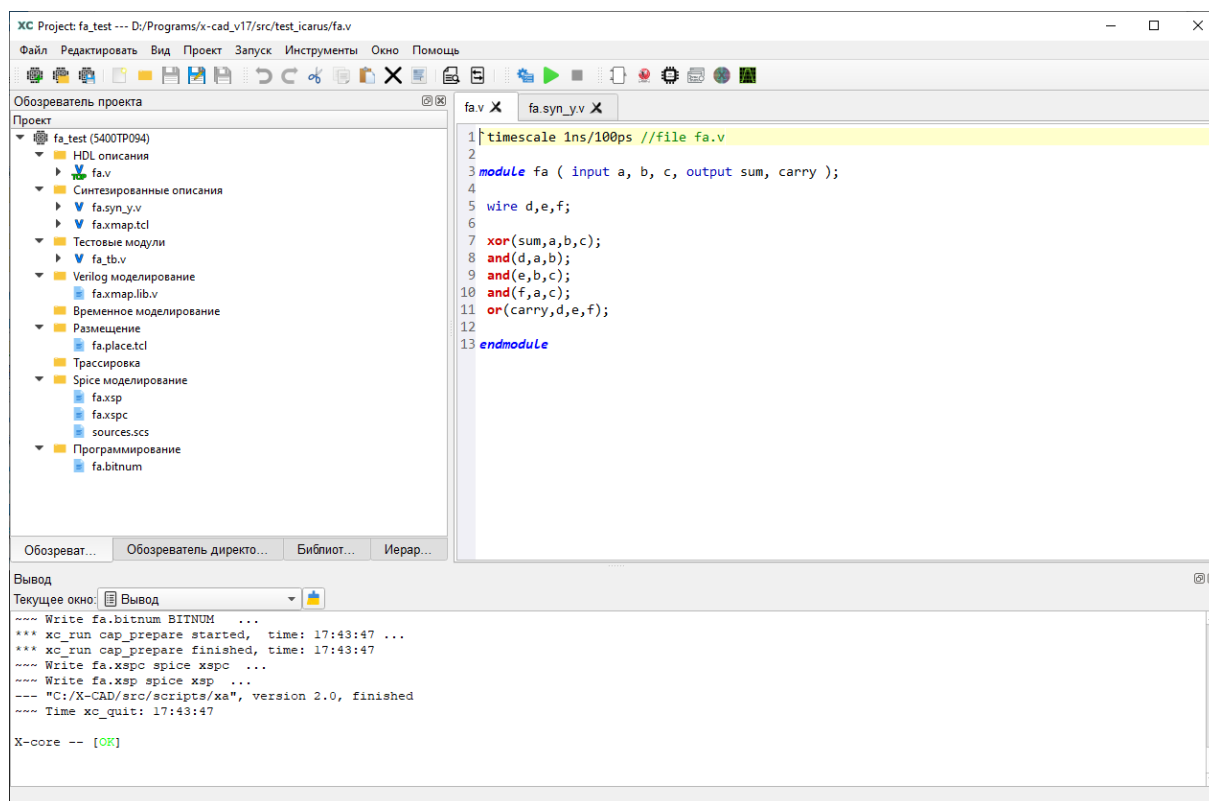


Рисунок 11 – Главное окно программы

Мастер создания проекта

Мастер создания проекта помогает пользователю создать проект. Его можно запустить, нажав на кнопку «Новый проект» в стартовом окне менеджера проектов, или выбрав пункт меню «Файл – Новый проект» в главном окне, или нажав соответствующую кнопку на панели инструментов главного окна. Окно мастера (рисунок 12) содержит следующие элементы интерфейса:

- строка «Целевой кристалл», отражающую текущий активный кристалл, который можно изменить посредством кнопки «Выбрать» с символом кристалла ПЛИС;
- поле «Директория», где можно вручную или посредством обозревателя (кнопка «Открыть» с изображением папки) указать путь до директории, где будут храниться все файлы, связанные с проектом (или дочерняя директория с именем проекта, где они будут храниться, если установлен флаг «Создать директорию проекта»); для совместимости с внешними инструментами, путь

может содержать только латиницу, цифры и знаки: «:», «/», «_», «.», «-»;

- поле «Имя проекта», где вручную указывается имя проекта (может содержать только латиницу, цифры и знаки «_» и «-»);
- поле «Полный путь» (нередатируемое), где отражается полный путь, по которому будет создан файл проекта;
- флаг «Создать директорию проекта», выставление которого приведет к созданию по пути, указанному в поле «Директория проекта», дочерней директории с именем проекта, где впоследствии будет создан файл проекта, и где будут располагаться все связанные с ним файлы;
- интерфейс управления проектными файлами со следующими элементами:
 - кнопка «Добавить» (с изображением папки) для добавления файлов в проект;
 - кнопка «Удалить» (с изображением красного креста) для удаления добавленных файлов, если необходимо;
 - поле «Главный модуль» для указания модуля верхнего уровня (можно выбрать только после добавления файлов в проект);
 - окно со списком добавленных файлов, где отражается имя и полный путь до файла, а также, можно выбрать в какую виртуальную папку будет перемещен файл после создания проекта;
- кнопка «Создать проект», которая запускает процесс создания проекта и открывает главное окно программы.

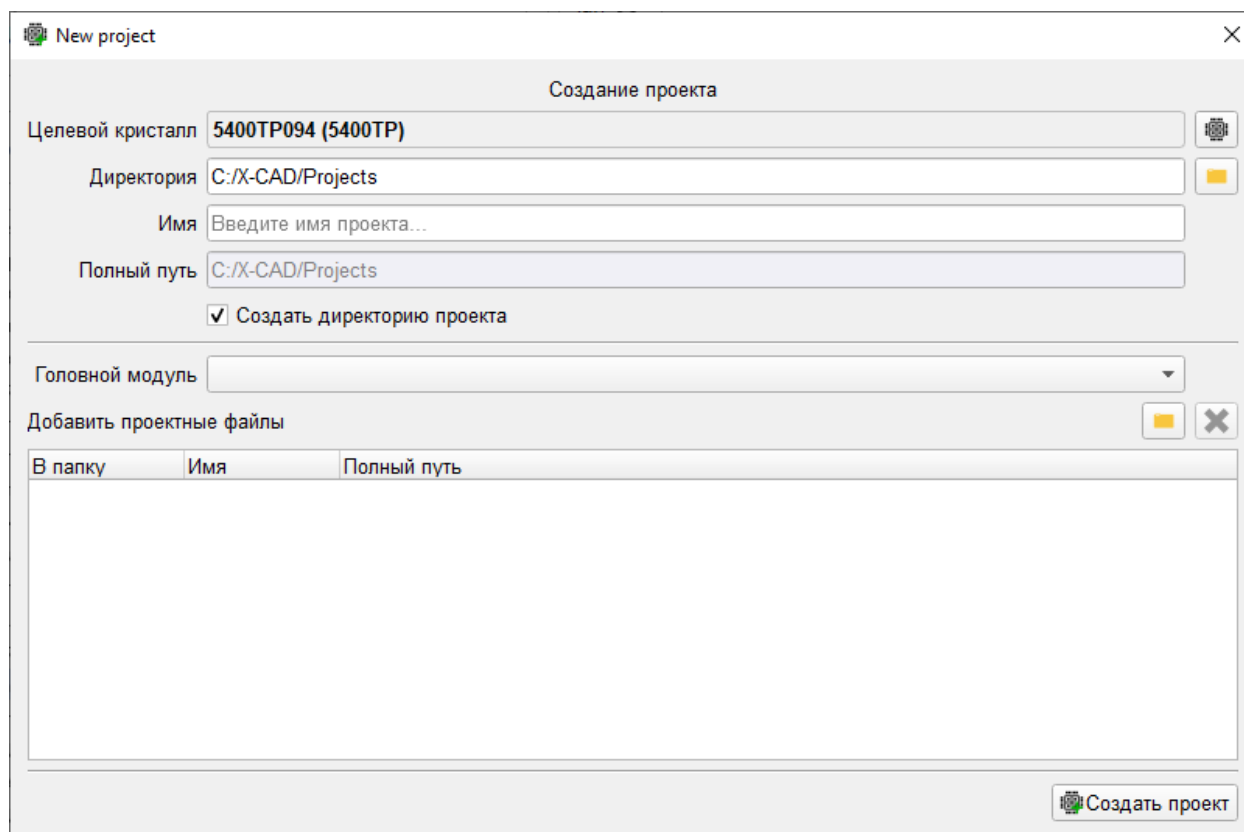


Рисунок 12 – Окно создания проекта

Если при создании проекта в пути или имени проекта были указаны недопустимые символы, то программа выдаст ошибку (рисунок 13).

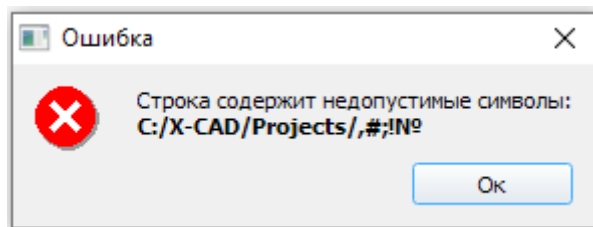


Рисунок 13 – Сообщение об ошибке в пути до директории проекта

Если же все данные были введены корректно, то после нажатия на кнопку «Создать проект» по пути, указанному в поле «Полный путь», будет создан файл проекта и сопутствующие файлы и директории. После этого окно мастера будет закрыто и запустится главное окно программы.

Также программа запомнит путь из поля «Директория» и в следующий раз, при создании проекта, по умолчанию укажет его. Изначально, путь указан до

директории «XCAD/Project», созданную в домашнем каталоге при первом запуске программы.

Окно выбора кристалла

Окно позволяет выбрать целевой кристалл (рисунок 14) и открывается через мастера создания проекта (кнопка «Выбрать» с изображением кристалла ПЛИС) или через главное окно программы («Проект – Выбор кристалла»). Окно содержит следующие элементы:

- фильтр «Семейство» в виде выпадающего списка с выбором целевого семейства кристаллов;
- окно со списком кристаллов (для указанного семейства) и дополнительной информацией по каждому кристаллу: «имя», «семейство», «ядро», «логические ячейки», «периферийные ячейки», «напряжение», «ток», «температура», «потребляемая мощность» и «производитель»;
- кнопка «Ок», подтверждающая выбор кристалла (для выбора необходимо выделить кристалл в списке нажатием на нем ЛКМ);
- кнопка «Отмена», отменяющая выбор и возвращающая к предыдущему выбранному кристаллу.

Подтвердить выбор также можно двойным нажатием ЛКМ по строке с нужным кристаллом.

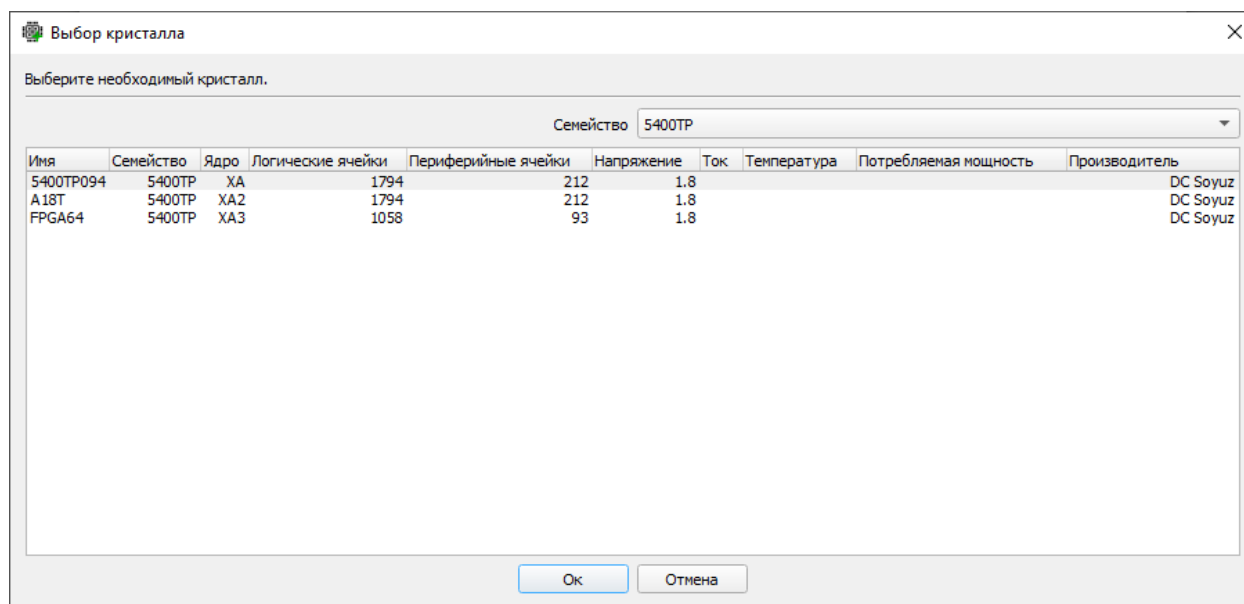


Рисунок 14 – Окно выбора кристалла

Главное окно для работы с проектом

Главное окно – основное пространство для работы с проектом, содержащее следующие элементы интерфейса:

- *Главное меню* (рисунок 15-1);
- *Панели инструментов* (рисунок 15-2);
- *Обозреватель проекта* (рисунок 15-3);
- *Обозреватель проектной директории* (рисунок 15-3);
- *Дерево иерархии проекта* (рисунок 15-3);
- *Окно библиотек проекта* (рисунок 15-3);
- *Текстовый редактор* (рисунок 15-4);
- *Окно «Вывод»* (рисунок 15-5);
- *Статусная строка* (рисунок 15-6).

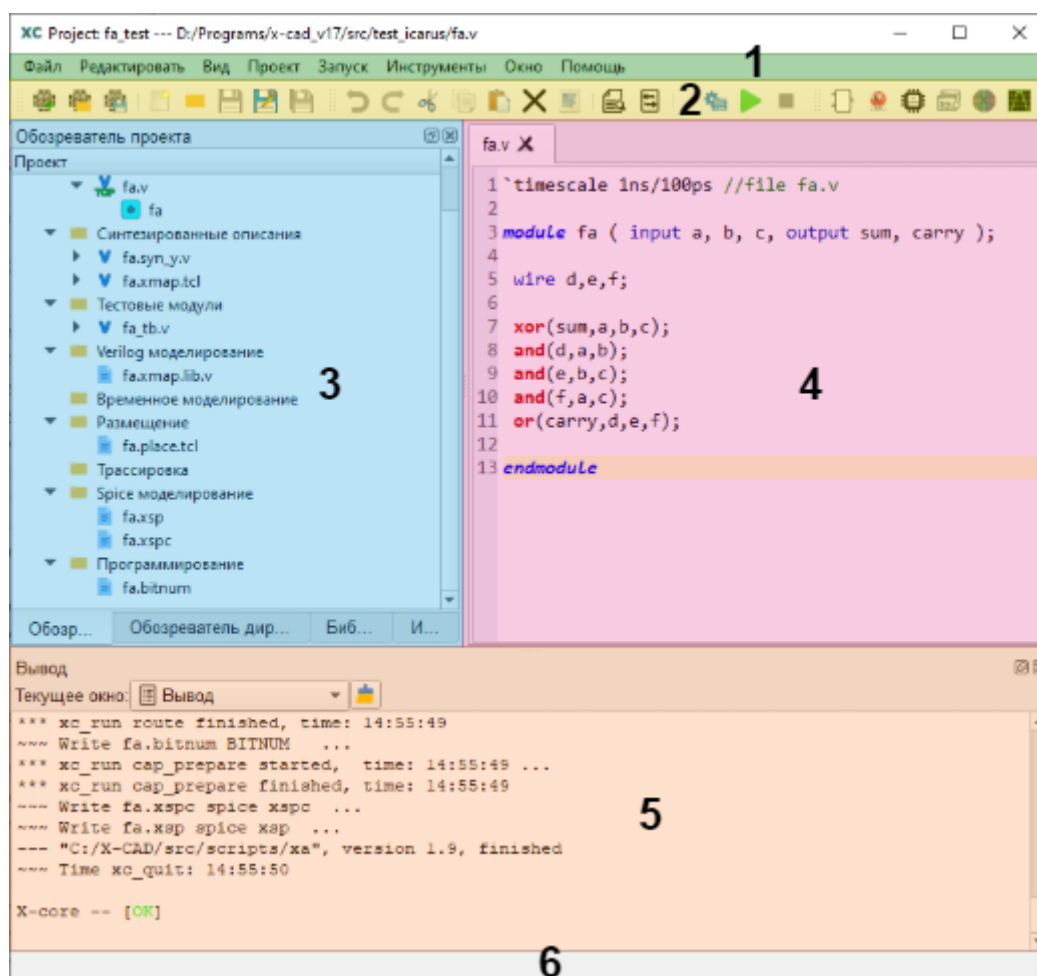


Рисунок 15 – Главное окно программы: 1 – главное меню;

2 – панели инструментов; 3 – обозреватель проекта, директории проекта, дерева иерархии и библиотек проекта; 4 – текстовый редактор;

5 – окно вывода; 6 – статусная строка

Главное меню рабочего окна

Главное меню (рисунок 15-1) – панель с дочерними меню, содержащими основные действия для работы с программой. Меню содержит следующие дочерние меню:

- «Файл» – создание, загрузка и сохранение проектов и файлов, завершение работы программы;
- «Редактировать» – инструменты текстового редактора;
- «Вид» – настройки видимости элементов интерфейса главного окна;

- «Проект» – выбор целевого кристалла, обзор директории проекта;
- «Запуск» – настройки, запуск и прерывание работы маршрута;
- «Инструменты» – запуск дополнительных инструментов и сторонних программ, а также настройка путей до программ (рисунок 16);
- «Окно» – изменение шрифта приложения, стиля окон и языка интерфейса (русский/английский);
- «Помощь» – краткая информация о программе.

Если после внесения изменений в окно настроек путей оно будет закрыто посредством системной кнопки закрытия окна, всплывет сообщение с предложением сохранить изменения, отменить изменения или отменить выход из окна.

Также, в окне изменения путей можно сбросить настройки до значений по умолчанию (кнопка «По умолчанию»).

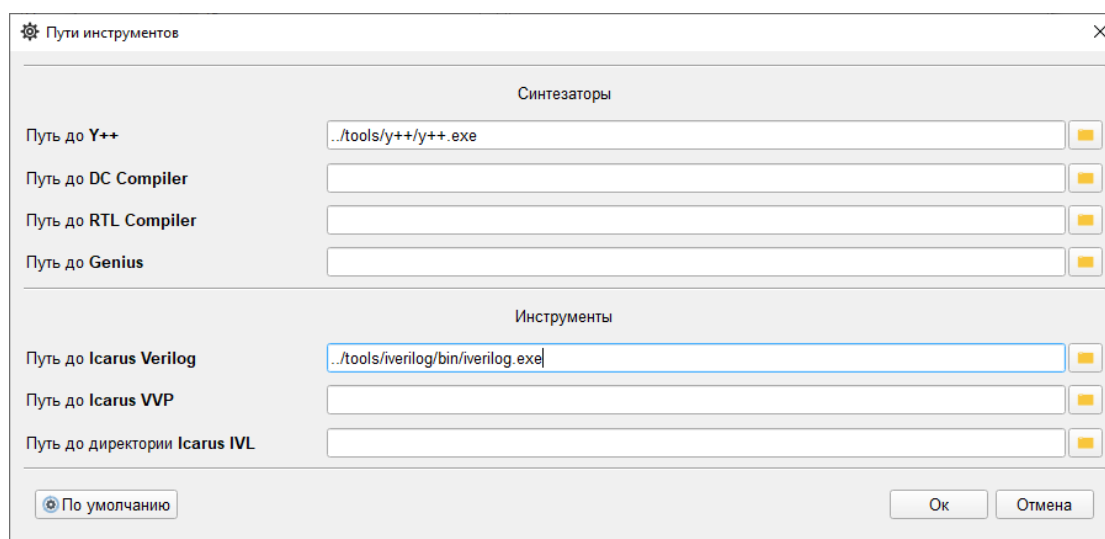


Рисунок 16 – Окно настроек путей расположения до программ

Панели инструментов

Панели инструментов (рисунок 15-2) повторяют наиболее востребованные в процессе работы действия, содержащиеся в меню, для быстрого доступа. Сюда включены следующие панели:

- Главная панель инструментов (повторяет меню «Файл»);
- Инструменты для работы с текстовым редактором (повторяет меню «Редактировать»);
- Инструменты для настройки и управления запуском (повторяет меню «Запуск»);
- Панель внешних инструментов (повторяет меню «Инструменты»);

Панели могут быть передвинуты пользователем в любую часть окна.

Обозреватель проекта

Обозреватель проекта (рисунок 15-3, 17) – окно в котором отображается структура проекта. Оно содержит корневой элемент с именем проекта и дочерние элементы-папки, в которых могут находиться файлы.

- «HDL описания» – для файлов с описанием схем на языках: Verilog, System Verilog, VHDL;
- «Синтезированные описания» – для синтезированных Verilog-файлов и файлов с описанием схемы в базисе ПЛИС;
- «Тестовые файлы» – для файлов с тестовыми модулями, содержащими входные воздействия;
- «Verilog моделирование» - для Verilog-файлов, необходимых для поведенческого моделирования;
- «Временное моделирование» - для файлов, содержащих отчеты временного анализа;
- «Размещение» - для файлов, содержащих результаты размещения элементов;
- «Трассировка» - для файлов, содержащих результаты трассировки межсоединений;
- «Spice-моделирование» - для входных и выходных файлов Spice-моделирования;
- «Программирование» - для файлов прошивки.

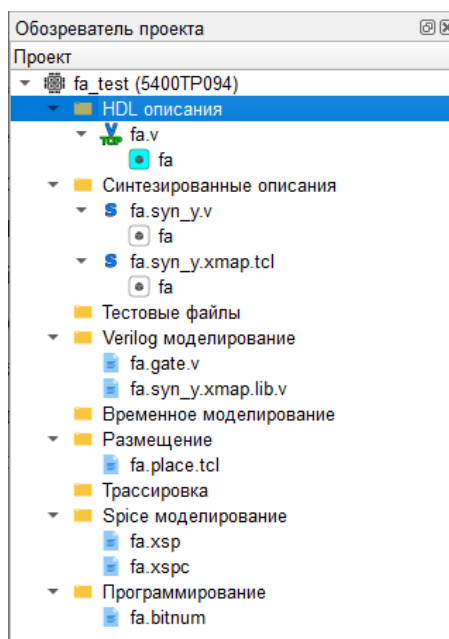


Рисунок 17 – Обозреватель проекта

Добавить файлы в папку можно как с помощью контекстного меню, так и посредством *механизма захвата и перетаскивания (Drag'n'Drop)*.

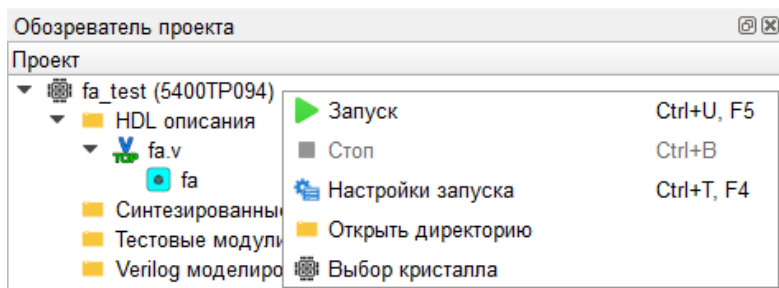
При добавлении в проект, *файлы анализируются*, и в элементах с именами файлов создаются дочерние элементы – модули, а также формируются списки входных/выходных цепей для окна настроек запуска.

Также *для всех файлов создаются резервные копии*, которые располагаются в директории «Backup» в директории проекта.

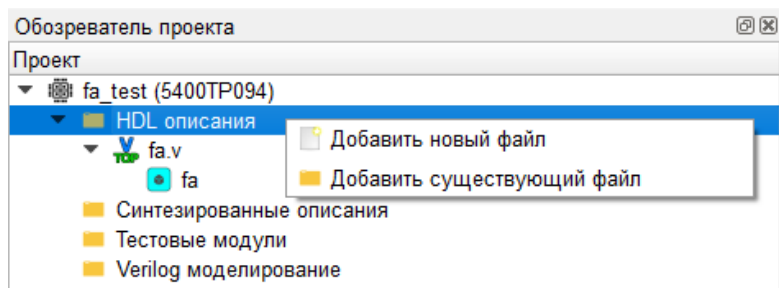
При добавлении описания схемы в пустой проект, программа предпринимает попытку *автоматически определить модуль верхнего уровня* (определяется наивысшим уровнем иерархии). Если таких модулей в файле несколько, то программа выставит первый найденный модуль в качестве модуля верхнего уровня. Если в файле не описаны модули, то дочерние элементы с именами модулей созданы не будут.

Поддерживается, также, перемещение файлов между виртуальными папками посредством *механизма захвата и перетаскивания*.

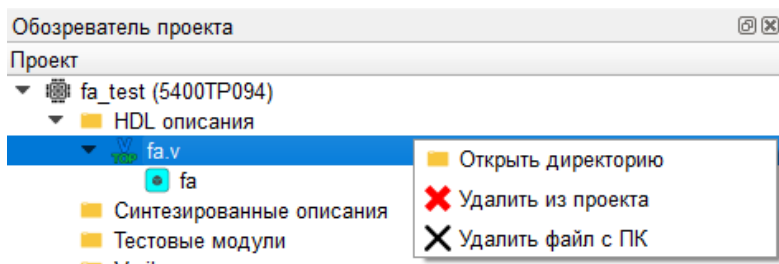
Для каждого из уровней структуры проекта существует контекстное меню, вызываемое ПКМ (рисунок 18).



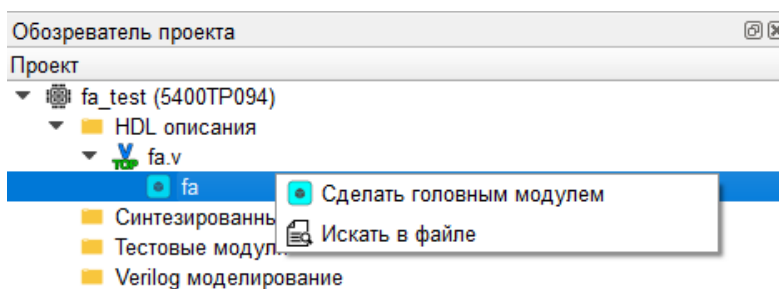
а)



б)



в)



г)

Рисунок 18 – Контекстные меню обозревателя проекта для:

а – проекта; б – папки; в – файла; г – модуля

Меню проекта (рисунок 18-а):

- «Запуск» – запускает выполнение маршрута;

- «Стоп» – останавливает выполнение маршрута;
- «Настройки запуска» – открывает окно настроек запуска;
- «Открыть директорию» – открывает расположение файла проекта;
- «Выбор кристалла» – открывает окно выбора кристалла.

Меню папки (рисунок 18-б):

- «Добавить новый файл» – открывает окно создания нового файла;
- «Добавить существующий файл» – открывает окно выбора существующего файла.

Меню файла (рисунок 18-в):

- «Открыть директорию» – открывает расположение файла;
- «Удалить из проекта» – исключает файл из проекта.
- «Удалить файл с ПК» - безвозвратно удаляет файл с ПК.

Меню модуля (рисунок 18-г):

- «Сделать головным модулем» – устанавливает выбранный модуль в качестве модуля верхнего уровня иерархии;
- «Искать в файле» – открывает расположение модуля в файле.

Двойной щелчок ЛКМ по элементу файл открывает его в текстовом редакторе.

Двойной щелчок ЛКМ по модулю делает его модулем верхнего уровня.

Обозреватель проектной директории

Обозреватель проектной директории (рисунок 15-3, 19) отображает файлы, содержащиеся в рабочей директории проекта. Двойной щелчок ЛКМ по файлу откроет его в текстовом редакторе программы.

Обозреватель директории проекта

| Имя файла | Размер | Тип файла | Изменён |
|--------------------------|------------|-------------|------------------|
| Backup | | File Folder | 17.06.2020 10:49 |
| fa_syn_y.v | 357 байты | v File | 17.06.2020 10:49 |
| fa_syn_y.xmap.tcl | 1,23 KiB | tcl File | 17.06.2020 10:49 |
| fa.v | 181 байты | v File | 17.06.2020 10:37 |
| fa_STA.reports | | File Folder | 18.06.2020 9:53 |
| fa_gate.nworst.txt | 15,11 KiB | txt File | 18.06.2020 9:53 |
| fa_gate.nworst_slack.tcl | 87 байты | tcl File | 18.06.2020 9:53 |
| fa_gate.sdf | 2,18 KiB | sdf File | 18.06.2020 9:53 |
| fa_gate.sl_table.tcl | 733 байты | tcl File | 18.06.2020 9:53 |
| fa_gate.sta.log | 5,54 KiB | log File | 18.06.2020 9:53 |
| fa_gate.sta_run.log | 5,44 KiB | log File | 18.06.2020 9:53 |
| Y_WORK | | File Folder | 17.06.2020 10:49 |
| fa.bitnum | 43,91 KiB | bitnum File | 18.06.2020 9:53 |
| fa.flat.log | 3,67 KiB | log File | 18.06.2020 9:53 |
| fa.flat_sta.v | 771 байты | v File | 18.06.2020 9:53 |
| fa_gate.v | 1,04 KiB | v File | 18.06.2020 9:53 |
| fa.log | 121,93 KiB | log File | 18.06.2020 9:53 |
| fa.place.tcl | 500 байты | tcl File | 18.06.2020 9:53 |
| fa_syn_y.v | 357 байты | v File | 18.06.2020 9:53 |
| fa_syn_y.xmap.lib.tcl | 90,66 KiB | tcl File | 18.06.2020 9:53 |
| fa_syn_y.xmap.lib.v | 349 байты | v File | 18.06.2020 9:53 |
| fa_syn_y.xmap.log | 869 байты | log File | 18.06.2020 9:53 |
| fa_syn_y.xmap.tcl | 1,23 KiB | tcl File | 18.06.2020 9:53 |
| fa_syn_y.xmap.v | 626 байты | v File | 18.06.2020 9:53 |
| fa.xsp | 64,54 KiB | xsp File | 18.06.2020 9:53 |
| fa.xspc | 109,33 KiB | xspc File | 18.06.2020 9:53 |
| fa.test.log | 21,48 KiB | log File | 18.06.2020 9:53 |

Рисунок 19 – обозреватель директории проекта

Окно иерархии проекта

Окно иерархии проекта (рисунок 15-3, 20) содержит иерархию модулей, описанных в HDL-файлах проекта, начиная от модуля верхнего уровня. Если модуль верхнего уровня не назначен, окно будет пустым.

Двойной щелчок ЛКМ по модулю открывает в текстовом редакторе файл, где данный модуль вызывается модулем более высокого уровня. Щелчок ПКМ по модулю вызывает контекстное меню, в котором доступен переход к месту объявления данного модуля.

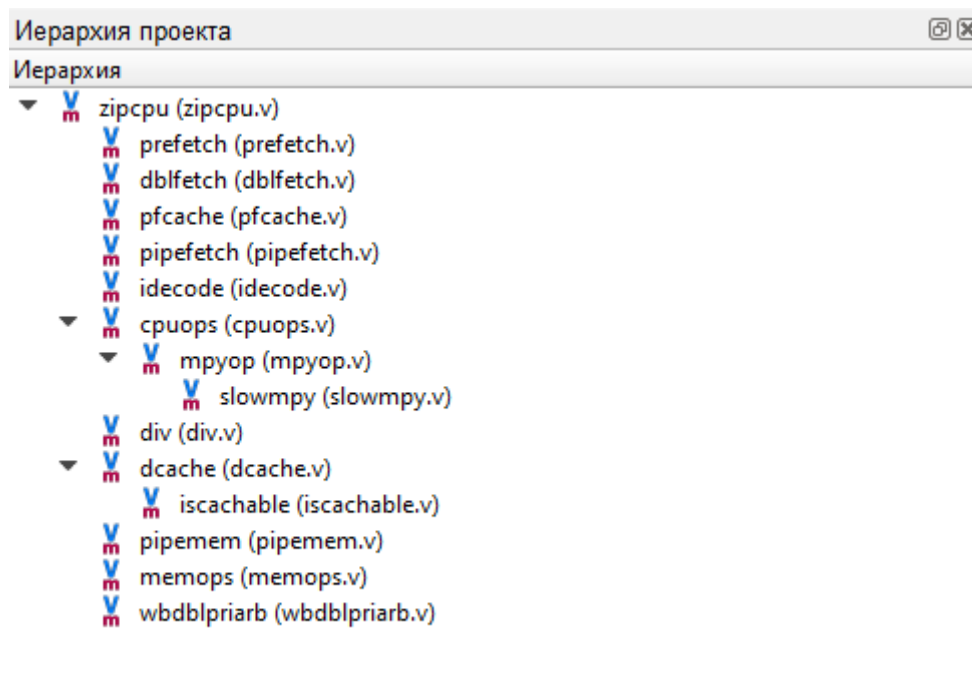


Рисунок 20 – окно иерархии проекта

Окно библиотек проекта

Окно библиотек проекта (рисунок 15-3, 21) структурирует HDL-описания проекта в виде библиотек. Данная особенность необходима для некоторых логических синтезаторов. По умолчанию окно содержит единственную библиотеку «work». Для добавления новых библиотек или перемещения файлов в уже существующие, необходимо вызвать контекстное меню и выбрать пункт «Задать библиотеку». В контекстном меню библиотеки (вызывается щелчком ПКМ) доступна опция удаления библиотеки. После удаления библиотеки, все файлы, находящиеся в ней, будут перемещены в библиотеку по умолчанию.

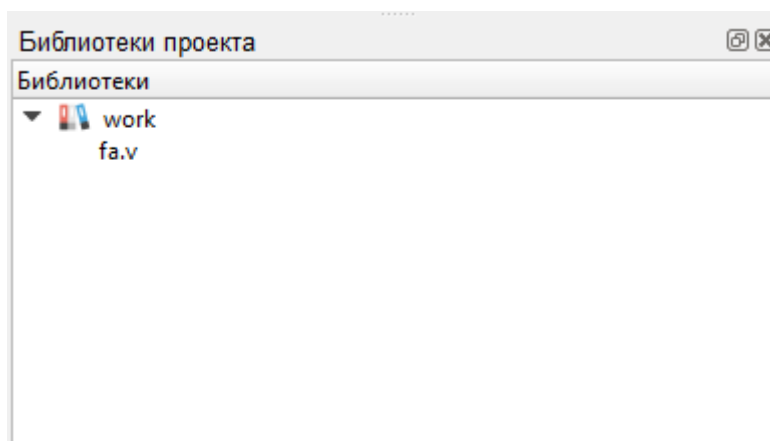


Рисунок 21 – окно библиотек проекта

Текстовый редактор

Редактор текстовых файлов (рисунок 15-4, 22), который обладает следующими возможностями:

- Подсветка синтаксиса для языка Verilog и Tcl;
- Автодобавление отступа равного отступу предыдущей строки;
- Контекстное меню (повторяет меню «Редактировать»);
- Номера строк;
- Выделение текущей строки;
- Поддержка горячих клавиш;
- Поиск и замена в файле (чувствительность к регистру, поиск по регулярному выражению, замена одного или всех вхождений).
- Поддержка режима захвата и перетаскивания файлов (*Drag 'n 'Drop*).

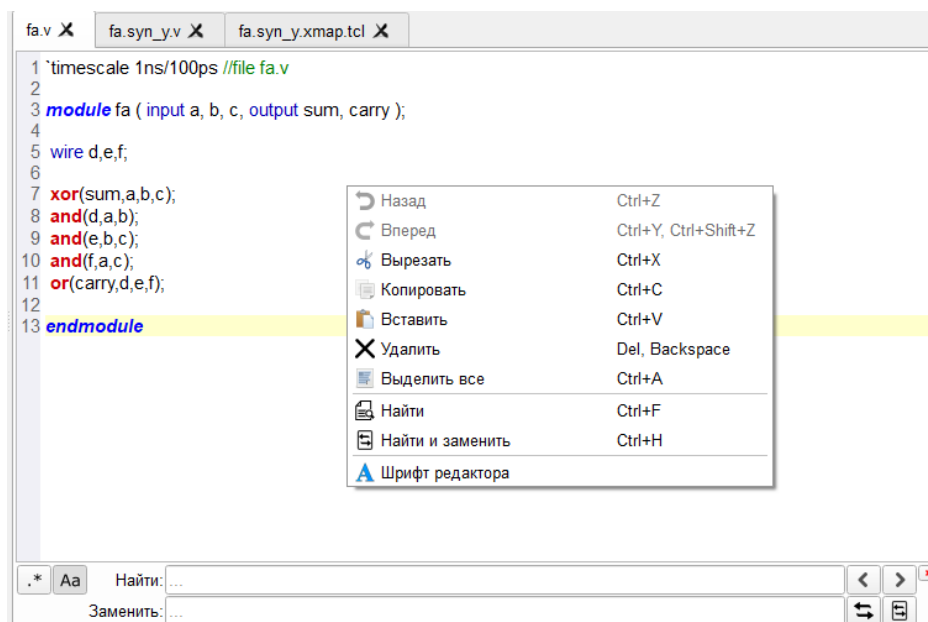


Рисунок 22 – Текстовый редактор

Окно «Вывод»

Окно «Вывод» (рисунок 15-5, 23) – окно с тремя режимами работы: «Вывод», «Ошибки и предупреждения» и список ошибок. Выбор режима осуществляется через выпадающий список «Текущее окно».

В режиме «Вывод» в окно перенаправляется весь текстовый вывод программ, которые были запущены в процессе работы маршрута, а также информация о начале и завершении работы программ, их пути и строки вызова, а также коды завершения. В данном режиме поддерживается подсветка строк с ошибками и предупреждениями, а также с ключевыми словами «ОК», «STOPPED», «ERROR», «WARNING», и строкой с именем созданного файла прошивки. Весь вывод, содержащийся в окне в режиме «Вывод», сохраняется в файл «имя_проекта.log» в директории проекта.

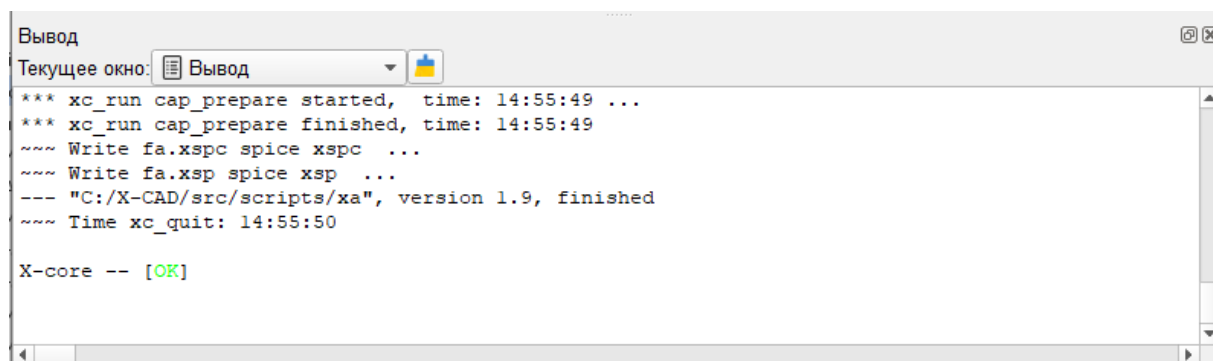


Рисунок 23 – Окно «Выход»

Режим «Ошибки и предупреждения» предназначен только для вывода строк с ошибками и предупреждениями программ. Весь остальной вывод здесь отображен не будет. Подсветка строк аналогична режиму «Выход».

В режиме «Список ошибок» окно, также, как и в режиме «Ошибки и предупреждения», содержит только ошибки и предупреждения полученные в результате работы программ. Окно поддерживает (по двойному щелчку ЛКМ) переход к строке, вызвавшей ошибку или предупреждение. Однако здесь будут отражены сообщения только тех программ, которые поддерживаются программой X-CAD. На данный момент поддерживаются следующие программы: Yosys и Icarus Verilog.

В окне также имеется кнопка очистки окон вывода, расположенная рядом с элементом выбора поля.

Статусная строка

Статусная строка (рисунок 15-6) служит для вывода информационных сообщений. Например, сообщение о готовности проекта, строка вызова последней программы, подсказка именовании пунктов меню и др.

Настройки запуска

Активация пункта «Настройки запуска» меню «Запуск» (или соответствующей кнопки на панели инструментов, или пункта контекстного

меню проекта) открывает окно настроек запуска (рисунок 24), где содержатся настройки параметров синтеза и маршрута в целом.

Включение/отключение опции происходит путем активации кнопки «Да/Нет». Если выбран вариант «Нет» (кнопка окрашена серым цветом) - опция отключена и не повлияет на работу маршрута; а если выбран вариант «Да» (кнопка окрашена зеленым цветом) - опция будет активна и внесет соответствующие изменения в работу маршрута.

Если опция требует ввода дополнительной информации, но в поле не указано значение, поле будет подсвечено красным цветом.

Если после внесения изменений окно будет закрыто посредством системной кнопки закрытия окна, всплывет сообщение с предложением сохранить изменения, отменить изменения или отменить закрытие окна.

Также, в окне настроек запуска можно сбросить настройки до значений по умолчанию.

Настройки запуска

Инструмент синтеза Y++

Маршрут LIB

Условия работы PtcV1.80T25

Использовать собственный скрипт синтеза Нет

Общие настройки

Задать имена выходных файлов [-o] Нет

Кластеризация элементов

Отключить объединение логики и триггеров [-nojoin] Нет

Использовать файл кластеризации LE [-le] Нет

Использовать файл кластеризации LAB [-lab] Нет

Отключить парное объединение LE [-le_union] Нет

Задать максимальное количество LE в LAB [-cl_lab_size] Нет 1

Задать максимальное количество LAB [-cl_no_labs] Нет 1

Размещение элементов

Использовать файл размещения IO ячеек [-inout] Нет

Использовать файл размещения LAB [-lab_place] Нет

Использовать файл размещения макро-блоков [-macro] Нет

Использовать файл размещения всех элементов [-p] Нет

Отключить алгоритм моделирования отжига [-sa] Нет

Задать множитель циклов моделирования отжига [-sa_lm] Нет 1

Трассировка межсоединений

Использовать файл порядка трассировки цепей [-order] Нет

PathFinder: Фактор истории поиска пути [-v_h] Нет 0,10

PathFinder: Фактор штрафа поиска пути [-v_p] Нет 0,50

PathFinder: Максимальный вес пути [-path_wl] Нет 300

Значения по умолчанию

Ок Отмена

Рисунок 24 – Окно настроек запуска

Пример работы программы

Запустите программу и активируйте кнопку «Создать проект». После этого откроется окно с параметрами создания нового проекта.

Заполните все необходимые поля. На данном этапе можно добавить необходимые файлы в проект. Для этого необходимо нажать на кнопку «Добавить» и выбрать файлы. Файлы также можно будет добавить и после создания проекта. Для создания проекта необходимо нажать на кнопку «Создать».

После этого откроется главное окно. Для примера был создан проект с именем «fa_test» (рисунок 25). После создания проекта откроется главное окно программы.

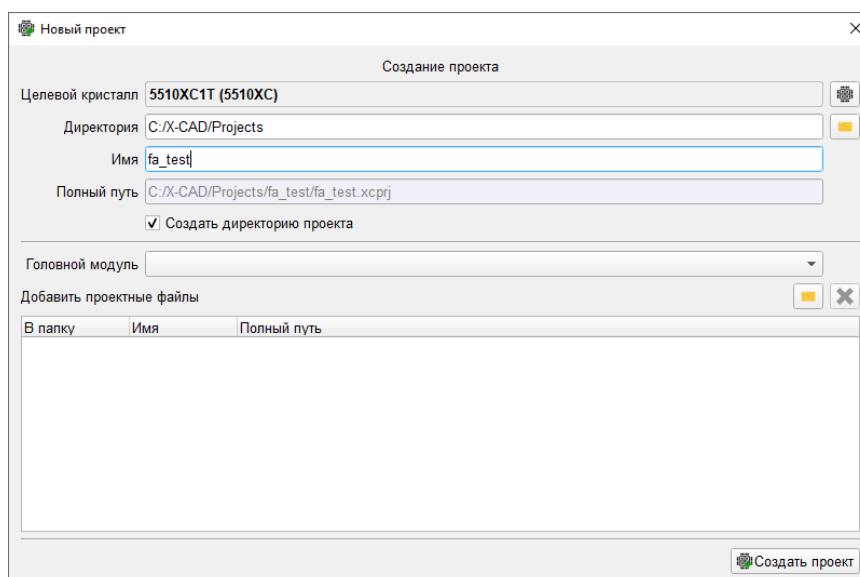


Рисунок 25– Создание тестового проекта

После создания проекта, можно добавить необходимые файлы описания схем на языке Verilog. Для этого примера была взята схема «fa_test» из набора тестовых схем Icarus Verilog (рис. 26).

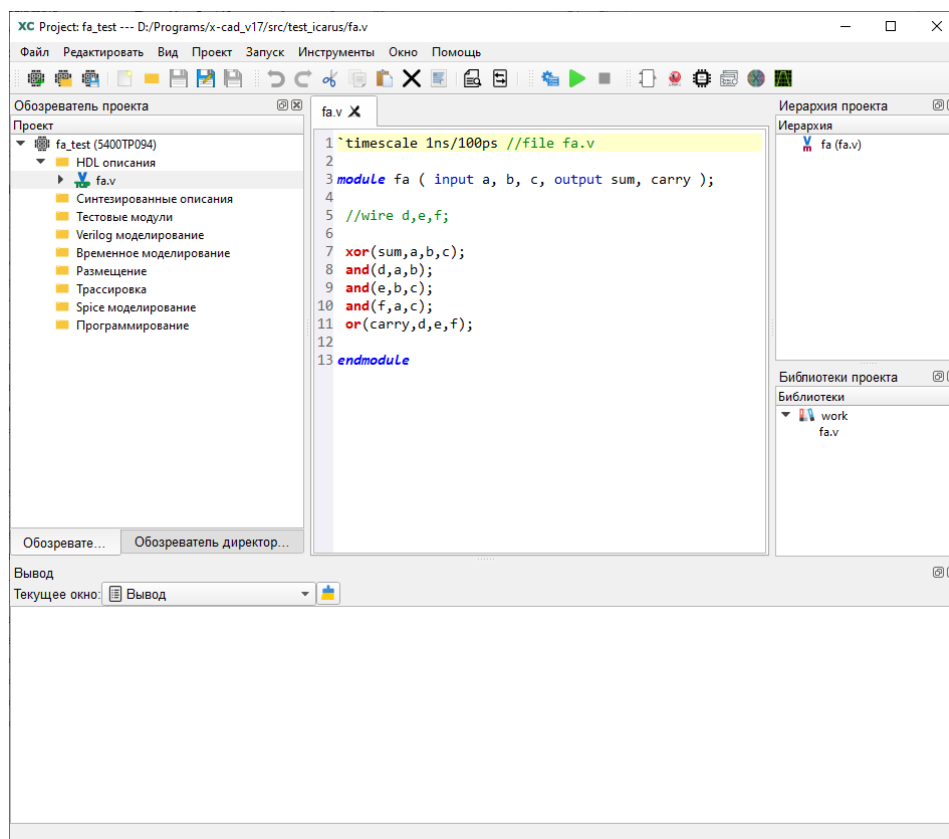


Рисунок 26 – Окно проекта с загруженным файлом схемы «fa»

На данном этапе можно запускать проект с параметрами по умолчанию, нажав на кнопку «Запуск» (с зеленой стрелкой), либо предварительно изменить настройки в соответствующем окне («Настройки запуска»). После этого начнется выполнение маршрута. Если в результате работы не было ошибок, в папке проекта будут сгенерированы такие файлы как: промежуточные описания, файлы размещения элементов на кристалле и прошивки (рисунок 27).

Для выполнения СВА необходимо включить его в настройках опцией «Запустить статический временной анализ» и задать список тактовых сигналов схемы, список значений периодов тактовых сигналов, максимальную время прибытия сигнала на выходы схемы и другие опции СВА.

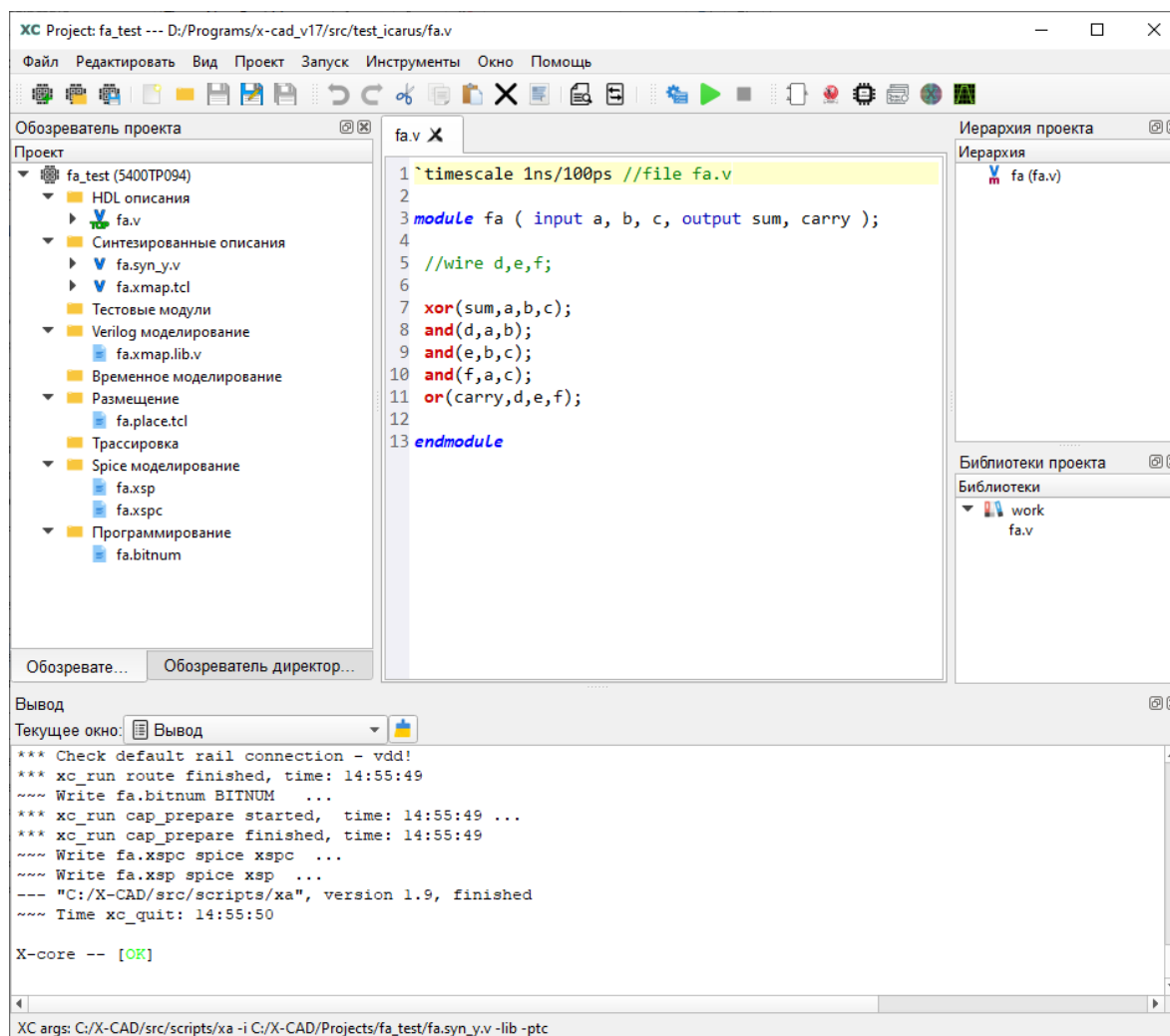


Рисунок 27 – Окно проекта после окончания всех процессов

После того, как завершится этап синтеза и сгенерируется синтезированное описание проектируемой схемы, можно установить данный файл в качестве главного («Синтезированные описания» – «Установить как текущий»). В этом случае, при запуске маршрута, этап синтеза будет пропущен, и сразу будет осуществлен запуск программы ХА. Это может быть удобно при необходимости сравнить результаты работы маршрута с разными параметрами ядра.

ИСПОЛЬЗОВАНИЕ ВНЕШНИХ ПРОГРАММ

Программа X-CAD предоставляет возможности для работы с внешними программами: X-Place, Icarus Verilog и GTK Wave.

Для каждой внешней программы, на панели инструментов и в меню «Инструменты», есть соответствующая кнопка, которая открывает окно с настройками запуска этой программы. В таком окне присутствуют следующие элементы: имя программы и соответствующая иконка, краткое описание, набор настроек запуска, поле дополнительных опций (на случай, если пользователю нужны другие опции, не представленные в окне), поле с результирующей строкой вызова программы и кнопки:

« Значения по умолчанию» - возвращает все настройки к значениям по умолчанию;

« Чистый запуск» - программа будет вызвана без аргументов;

« Запуск» - запуск программы со всеми текущими настройками запуска.

Icarus Verilog

Icarus Verilog - компилятор и симулятор языка описания аппаратуры Verilog стандарта IEEE-1364. Icarus Verilog имеет ограниченную поддержку временного моделирования и временной информации о схеме в формате Standard Delay Format (SDF). Формат SDF — это стандарт IEEE для представления и интерпретации временных данных для использования на любом этапе процесса проектирования электроники. Временное моделирование может быть полезно для выявления ложных путей и их проверки.

Для запуска моделирования Verilog-описания с помощью Icarus Verilog необходимо выполнить следующие действия:

- 1) добавить в папку «Тестовые модули» тестовый модуль (testbench) с описанием входных воздействий для тестируемой схемы;
- 2) установить добавленный тестовый модуль с входными воздействиями модулем верхнего уровня (рисунок 28);

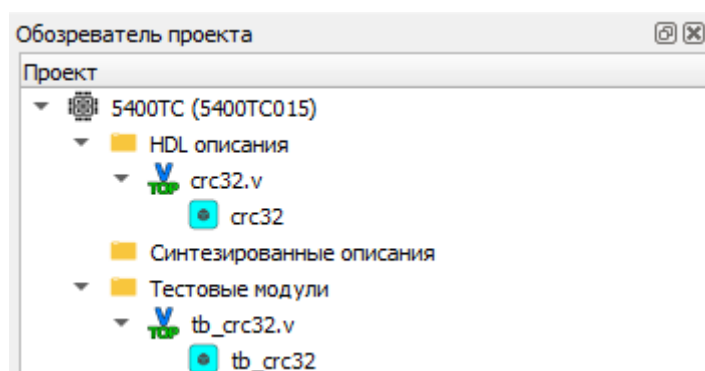


Рисунок 28 – Определение модуля верхнего уровня в окне обозревателя проекта

- 3) добавить в исходный код тестового модуля (testbench) строки для формирования временных диаграмм в формате VCD:

```
initial begin
    $dumpfile("<vcd_file_name>.vcd");
    $dumpvars(0,<tb_module_name>);
end
```

где:

<vcd_file_name> – имя выходного файла;

<tb_module_name> – имя тестового модуля верхнего уровня;

- 4) открыть окно запуска Icarus Verilog («Инструменты» – « Icarus Verilog»), выбрать нужный файл из выпадающего списка «Тестовый файл» и выбрать нужный модуль из списка «Тестовый модуль»;
- 5) выбрать необходимые файлы с описанием проектной схемы для моделирования;
- 6) нажать на кнопку «Запустить».

Моделирование Verilog описания проектируемого устройства с помощью Icarus Verilog может быть выполнено в четырех вариациях, после прохождения различных этапов проектирования в ПЛИС. Варианты моделирования:

- 1) Моделирование RTL описания. Выполняется для поведенческого моделирования и формальной верификации исходного описания проекта на уровне RTL.
- 2) Моделирование синтезированного описания. Выполняется для верификации результатов логического синтеза проектируемой схемы.
- 3) Моделирование структурного описания. Выполняется для верификации результатов технологического отображения проектируемой схемы в базис элементов ПЛИС.
- 4) Моделирование структурного описания после трассировки. Выполняется для финальной верификации проектируемой схемы, после топологического синтеза.

Моделирование RTL описания

Для проведения поведенческого моделирования RTL описания необходимо выбрать следующий набор файлов в окне «Входные файлы» для запуска Icarus Verilog (см. рисунок 30):

- Файл с тестовым модулем (testbench);
- Исходные файлы с описанием проекта.

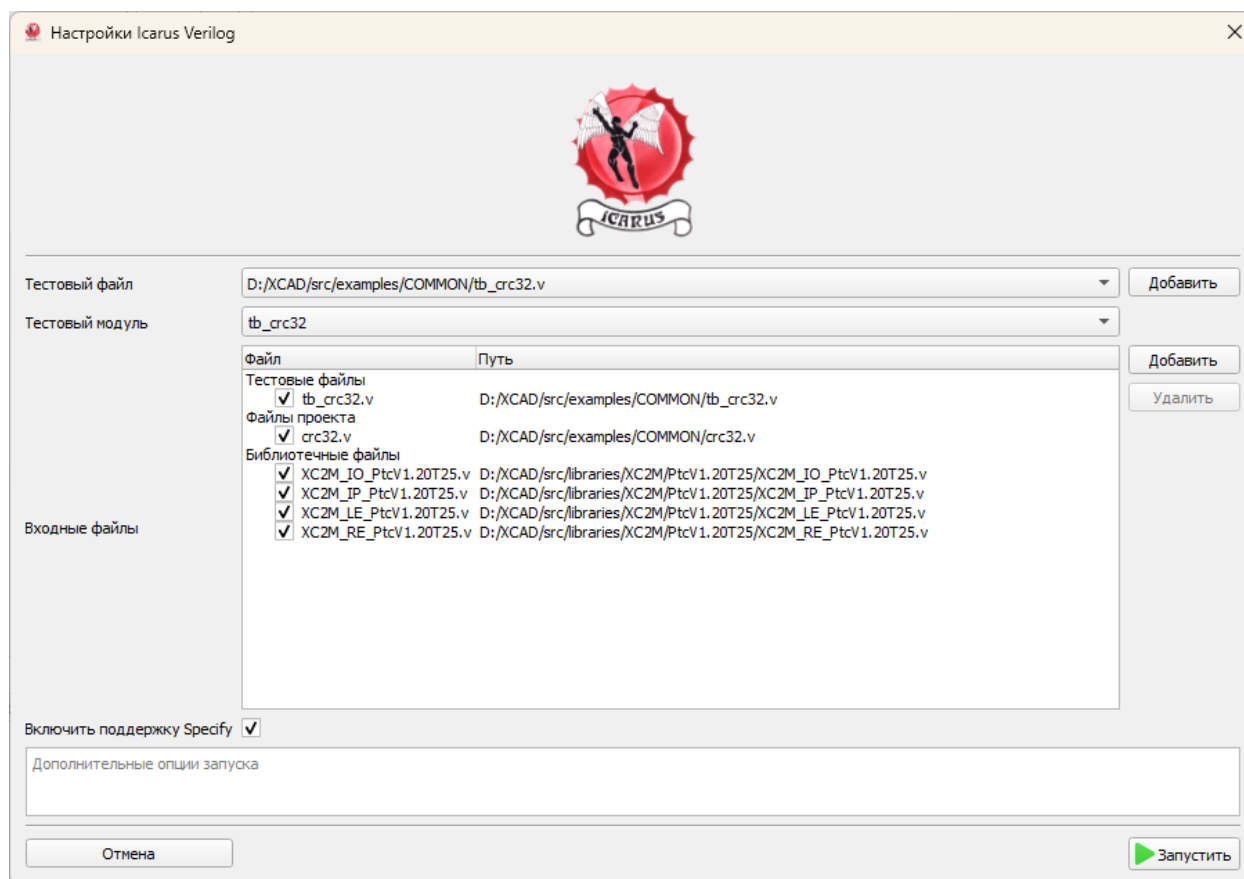


Рисунок 30 – Запуск моделирования RTL описания

Моделирование синтезированного описания

Для проведения поведенческого моделирования синтезированного описания необходимо выбрать следующий набор файлов в окне «Входный файлы» для запуска Icarus Verilog (см. рисунок 31):

- Файл с тестовым модулем (testbench);
- Синтезированные файлы с описанием проекта.

Синтезированные файлы могут иметь различный суффикс, в зависимости от использованного ПО для логического синтеза:

- ***.syn_y.v** - файл с результатами логического синтеза Yosys;
- ***.syn_gen.v** - файл с результатами логического синтеза Cadance Genus;
- ***.syn_dc.v** - файл с результатами логического синтеза Design Compiler.

При запуске моделирования синтезированного описания будут использованы усредненные задержки из конструкций specify, заданные в Verilog библиотеках использованных элементов. Для проведения моделирования без

учета этих задержек необходимо снять галочку с опции «Включить поддержку Specify».

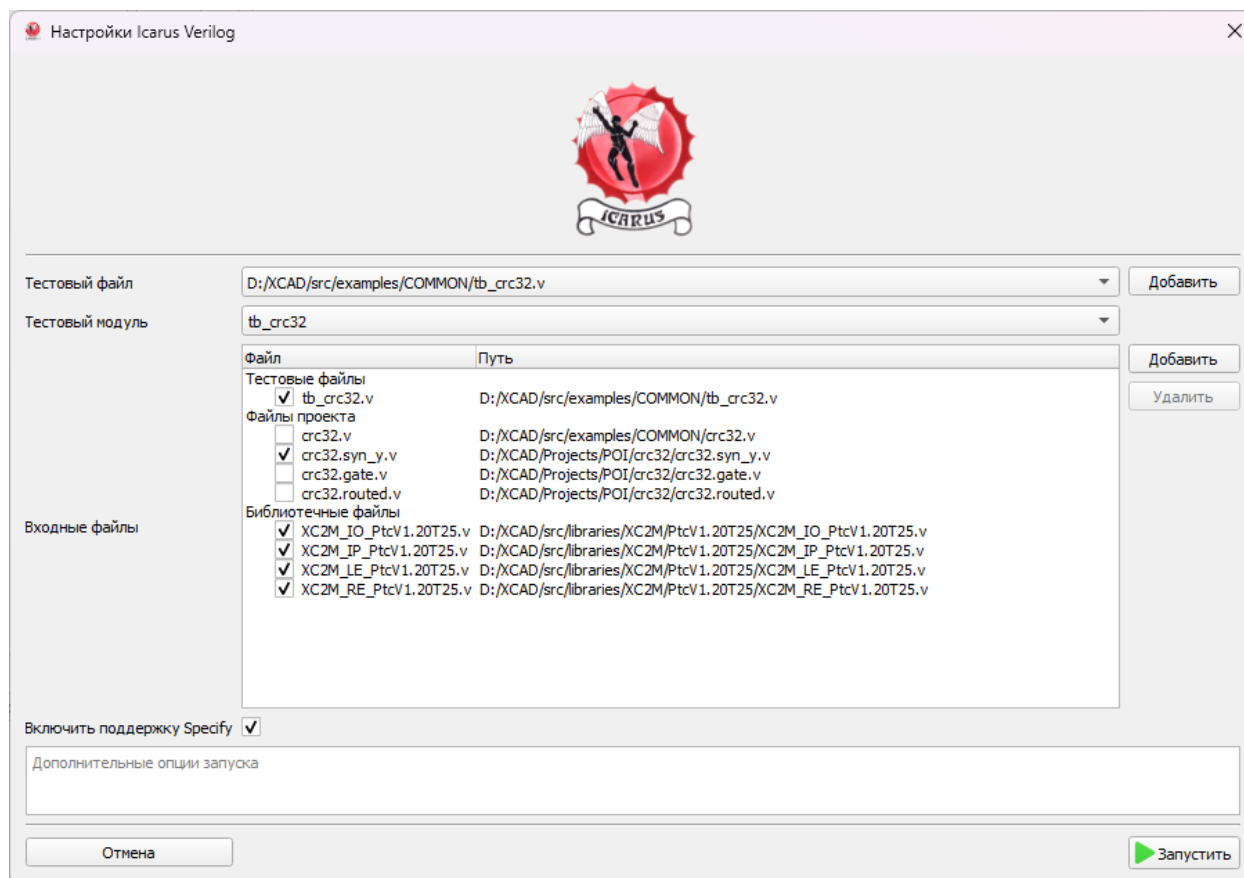


Рисунок 31 – Запуск моделирования синтезированного описания

Моделирование структурного описания

Для проведения поведенческого моделирования структурного описания после этапа технологического отображения необходимо выбрать следующий набор файлов в окне «Входный файлы» для запуска Icarus Verilog (см. рисунок 32):

- Файл с тестовым модулем (testbench);
- Синтезированные файлы с описанием проекта.

К синтезированным файлам для данного вида моделирования относятся:

- ***.gate.v** - файл со структурным описанием проекта на вентиляльном уровне в иерархическом виде после этапа технологического отображения в термины элементов ПЛИС;

- ***.flat.gate.v** - файл со структурным описанием проекта на вентиляльном уровне в плоском виде после этапа технологического отображения в терминах элементов ПЛИС;
- ***.xmap.glib.v** - опциональный файл с библиотекой дополнительных элементов, задействованных при технологическом отображении.

При запуске моделирования описания в иерархическом виде необходимо *всегда* подключать библиотеку дополнительных элементов, в то время как для моделирования описания в плоском виде этого делать не требуется.

Данный вид моделирования может быть запущен с учетом задержек в формате SDF, полученных в результате статического временного анализа. Для этого необходимо убедиться в подключении соответствующего SDF файла в структурном описании с помощью конструкции *initial*:

```
initial $sdf_annotate("<file_name>.gate.sdf");
```

Для проведения моделирования без учета этих задержек необходимо снять галочку с опции «Включить поддержку Specify», либо закомментировать конструкцию *initial* в Verilog описании.

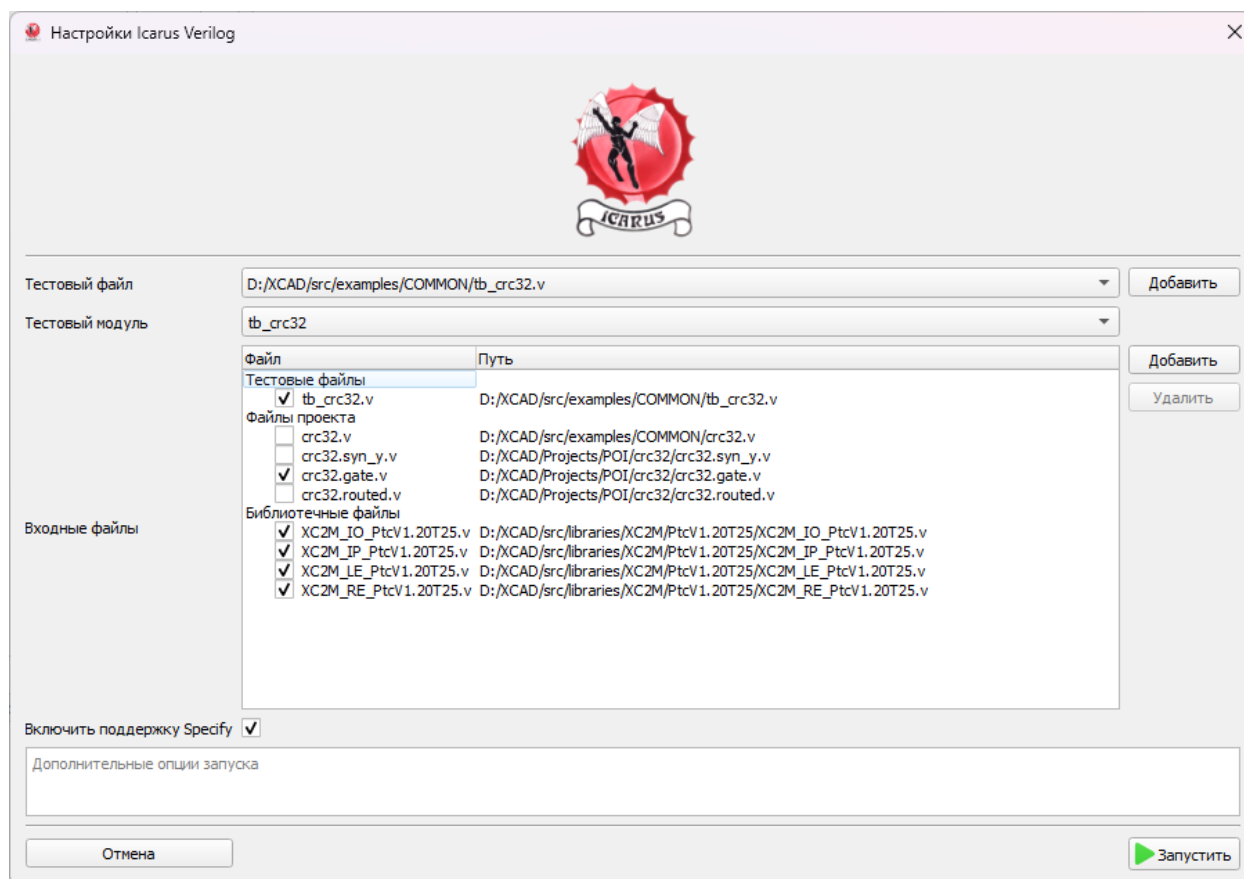


Рисунок 32– Запуск моделирования структурного описания

Моделирование структурного описания после трассировки

Для проведения поведенческого моделирования структурного описания после этапа трассировки в ПЛИС необходимо выбрать следующий набор файлов в окне «Входный файлы» для запуска Icarus Verilog (см. рисунок 33):

- Файл с тестовым модулем (testbench);
- Синтезированные файлы с описанием проекта.

К синтезированным файлам для данного вида моделирования относятся:

- ***.routed.v** - файл со структурным описанием проекта на вентиляном уровне в иерархическом виде после этапа трассировки в ПЛИС;
- ***.flat.routed.v** - файл со структурным описанием проекта на вентиляном уровне в плоском виде после этапа трассировки в ПЛИС;
- ***.xmap.glib.v** - опциональный файл с библиотекой дополнительных элементов, задействованных при технологическом отображении.

При запуске моделирования описания в иерархическом виде необходимо **всегда** подключать библиотеку дополнительных элементов, в то время как для моделирования описания в плоском виде этого делать не требуется.

Данный вид моделирования может быть запущен с учетом задержек в формате SDF, полученных в результате статического временного анализа. Для этого необходимо убедиться в подключении соответствующего SDF файла в структурном описании с помощью конструкции ***initial***:

```
initial $sdf_annotate("<file_name>.routed.sdf");
```

Для проведения моделирования без учета этих задержек необходимо снять галочку с опции «Включить поддержку Specify», либо закомментировать конструкцию ***initial*** в Verilog описании.

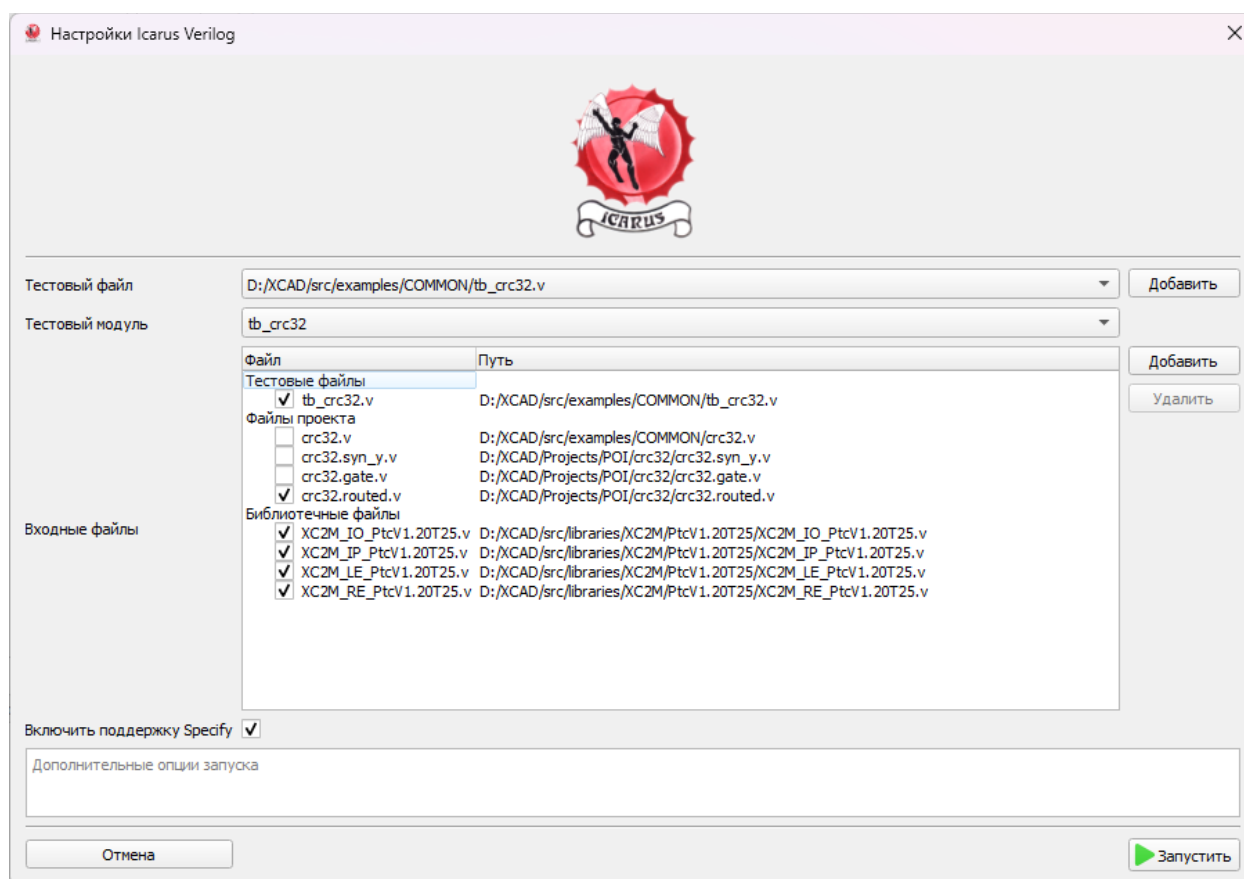


Рисунок 33 – Запуск моделирования структурного описания после трассировки

GTK Wave

По завершении работы Icarus Verilog образуется «*.vcd» файл, который можно открыть программой GTK Wave («Инструменты» – « GTK Wave») (рисунок 34) через соответствующее окно программы X-CAD для просмотра результатов моделирования (рисунок 35).

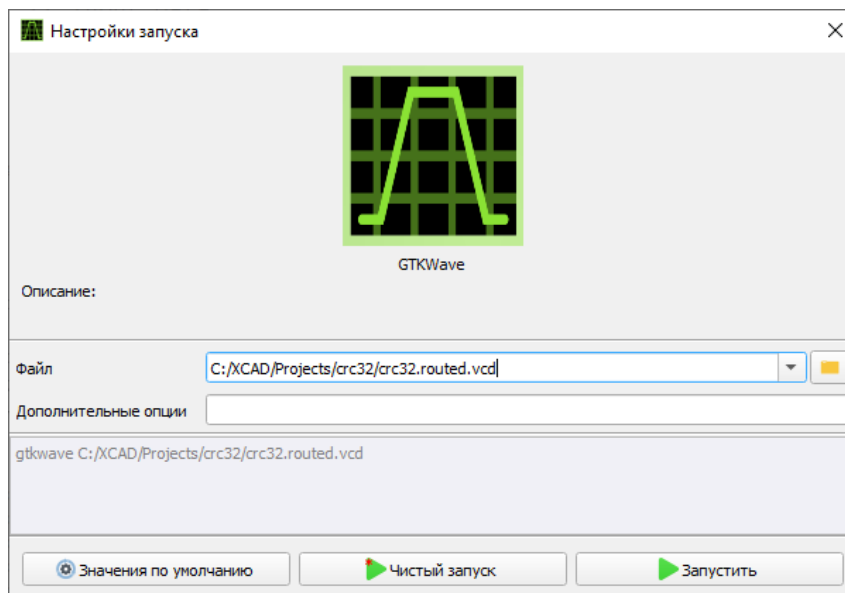


Рисунок 34 – Окно запуска GTK Wave

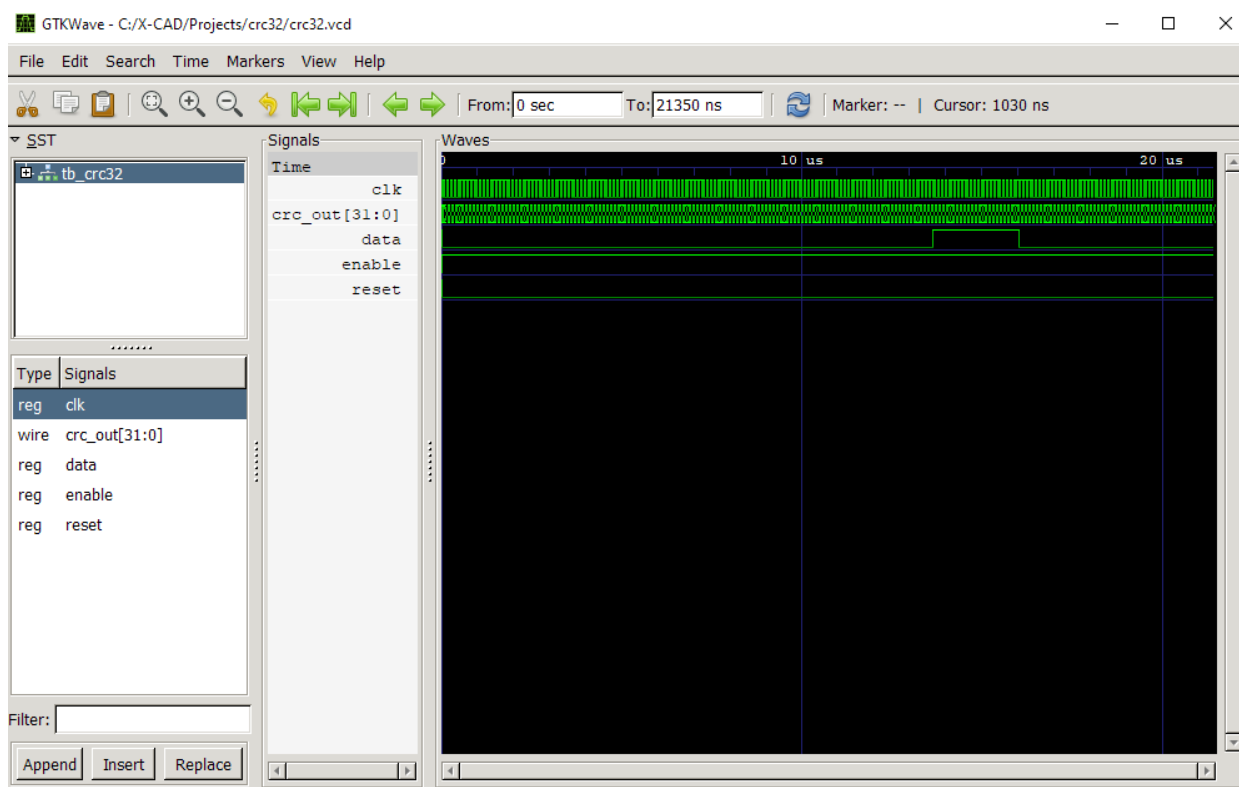


Рисунок 35 – Окно программы GTK Wave с результатами моделирования

Анализатор HDL-описаний

HDL-описания проектных схем на языке Verilog могут быть подвергнуты синтаксическому анализу. Для этого необходимо:

- 1) Открыть окно анализа («Инструменты» – «Анализ кода»):
- 2) В появившемся окне выбрать необходимый язык описания и нажать на кнопку «Ок» (рисунок 36);

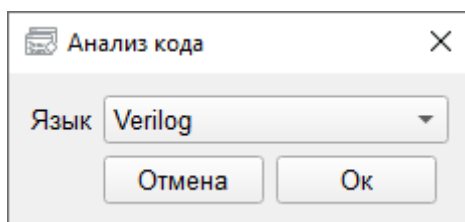


Рисунок 36 – Окно запуска анализа HDL-описаний

После завершения анализа появится окно с сообщением о количестве ошибок и предупреждений, либо их отсутствии. Найденные ошибки и предупреждения можно просмотреть в окне «Список ошибок» (рисунок 37). Двойной клик левой кнопкой мыши по строке данного списка приведет к открытию файла, в котором возникла проблема, и установке указателя на необходимую строку.

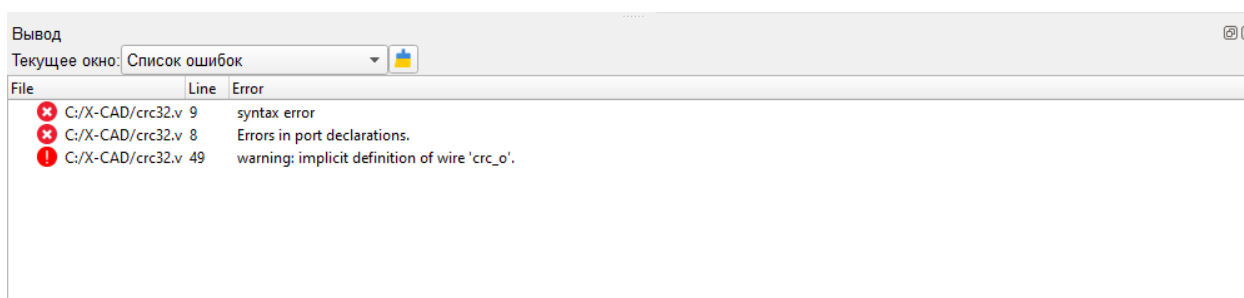


Рисунок 37 – Список ошибок

ИНТЕРФЕЙС ТОПОЛОГИЧЕСКОГО РЕДАКТИРОВАНИЯ X-PLACE

X-Place – графический интерфейс, предназначенный для просмотра и редактирования размещения функциональных и сложных функциональных блоков (СФ-блоков), периферийных блоков и макроблоков на кристалле ПЛИС.

Программа разработана с применением кроссплатформенных средств разработки и распространяется на платформах семейств Windows и Unix. Средства разработки включают библиотеки: Qt, OpenGL.

Запуск программы

Запуск программы может быть осуществлен посредством запуска исполняемого файла через обозреватель файлов, запуска из командной строки или через соответствующее окно программы X-CAD.

Запуск через обозреватель файлов

Для запуска через обозреватель файлов необходимо открыть папку с исполняемым файлом «xplace.exe», который расположен в директории «bin» пакета программ, и активировать двойным щелчком ЛКМ или вызвать с помощью правой кнопки мыши контекстное меню и выбрать соответствующий пункт для запуска приложения.

Запуск через командную строку

Чтобы запустить программу посредством командной строки, необходимо перейти в папку «bin» пакета программ и вызвать на исполнение файл «xplace.exe», используя следующие команды:

- Windows: “start xplace.exe <аргументы_и_значения>”
- Unix: “./xplace <аргументы_и_значения>”

Программа поддерживает следующие аргументы командной строки:

- -i «файл» – описание схемы (на языке Tcl в стандартном режиме);
- -l «файл» – библиотека элементов схемы (на языке Tcl);
- -p «файл» – файл размещения элементов (на языке Tcl);

- *-с «имя_кристалла»* – данные кристалла по имени, которое соответствует имени кристалла в файле «ха_config.ini».
- *-pp* – запуск программы в режиме планировки кристалла.

Запуск через интерфейс X-CAD

Для запуска программы следует активировать соответствующий пункт меню программы X-CAD («Инструменты» – «X-Place») (рисунок 38) и в открывшемся окне нажать кнопку «Чистый запуск», или установить необходимые значения параметров запуска и нажать «Запустить».

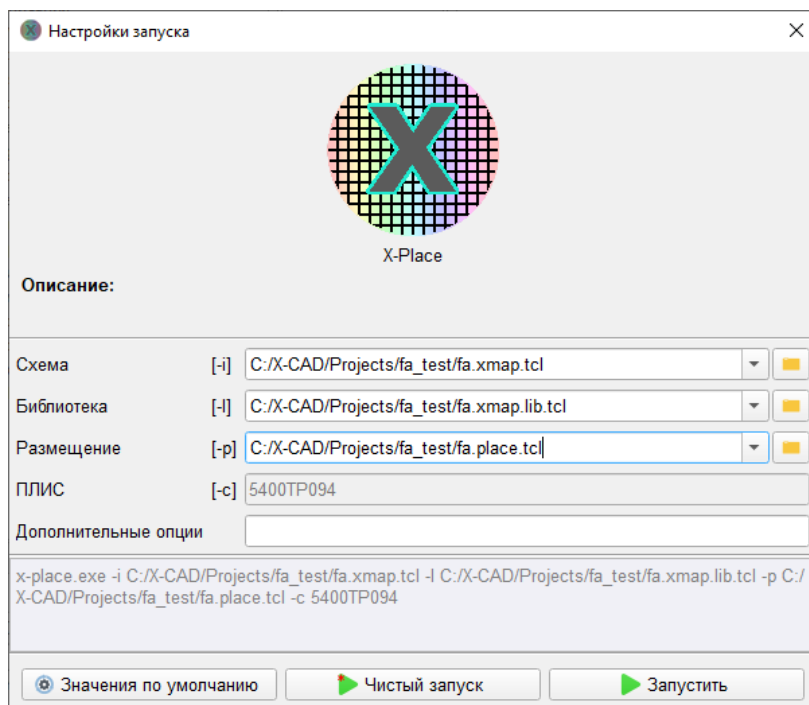


Рисунок 38 – Окно запуска X-Place

Главное окно для работы с проектом

Главное окно – основное пространство для работы с проектом, содержащее следующие элементы интерфейса:

- *Главное меню* (рисунок 38-1);
- *Панели инструментов* (рисунок 38-2);
- *Обозреватель структуры схемы* (рисунок 38-3);

- *Обозреватель списка цепей* (рисунок 38-4);
- *Рабочее графическое поле* (рисунок 38-5);
- *Окно «Tcl консоль»* (рисунок 38-6);
- *Статусная строка* (рисунок 38-7).

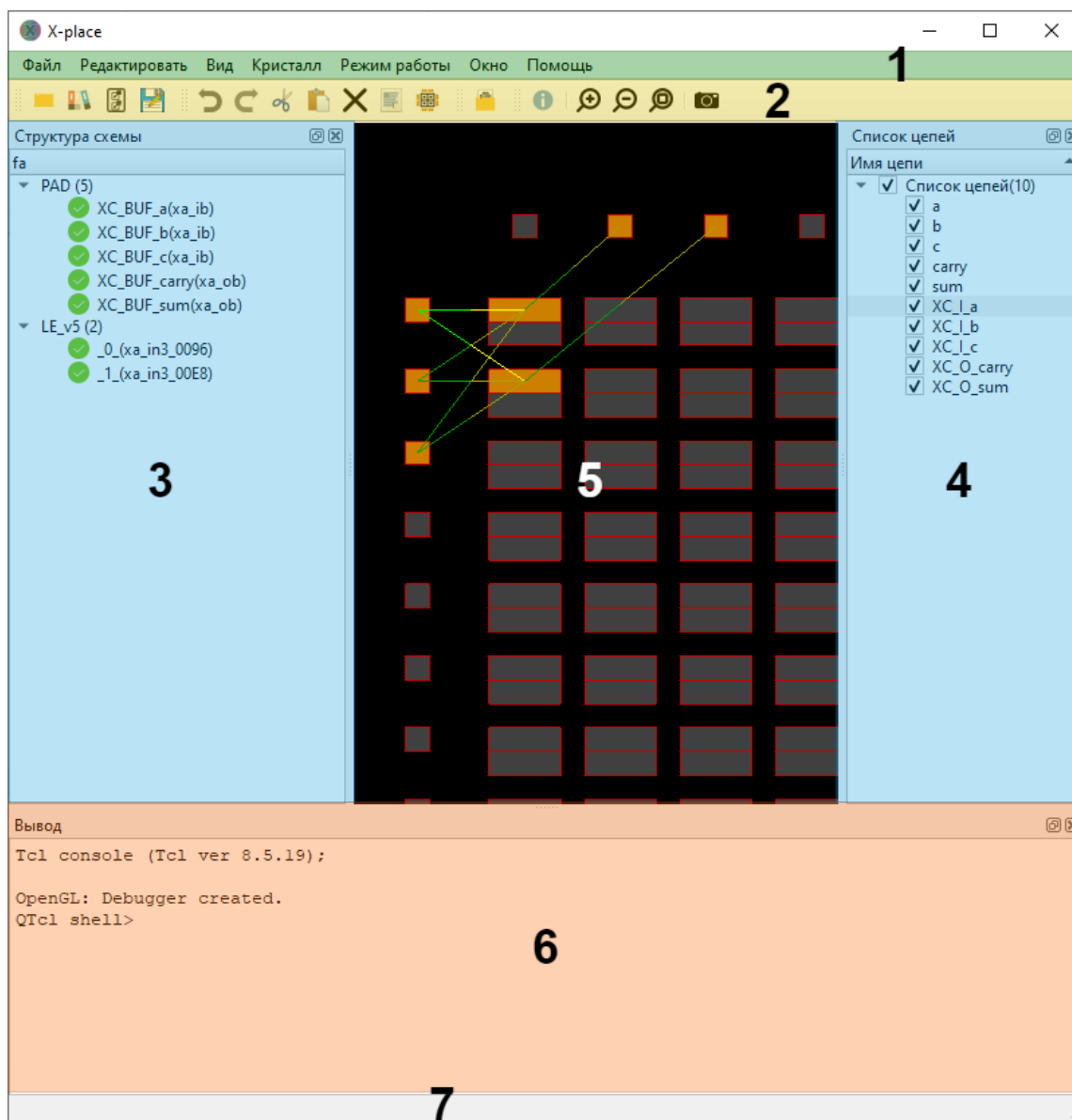


Рисунок 38 - Структура главного окна программы:

- 1 – главное меню; 2 – панели инструментов; 3 – обозреватель структуры схемы;
4 – обозреватель списка цепей;
5 – рабочее окно; 6 – окно «Tcl консоль»; 7 – статусная строка

Главное меню

Главное меню (рисунок 35-1) – панель с дочерними меню, содержащими основные действия для работы с программой. Меню содержит следующие дочерние меню:

- «Файл» – открытие описания схемы на языке Tcl в терминах ХА, сохранение текущего размещения, загрузка размещения, завершение работы программы;
- «Редактировать» – удаление выбранных размещенных элементов, выделение всех элементов;
- «Вид» – позволяет:
 - управлять графическим отображением кристалла;
 - Устанавливать режим отображения межсоединений;
 - Устанавливать режим отображения межсоединений выделенных объектов;
 - Включать/отключать динамическое выделение объектов на кристалле;
- «Кристалл» – выбор целевого кристалла, загрузка кристалла из index.tcl файла;
- «Режим работы» – переключение режимов работы: стандартный или FloorPlanner;
- «Окно» – настройка шрифта приложения, выбор языка;
- «Помощь» – позволяет получить вспомогательную информацию (находится в разработке).

Панели инструментов

Панели инструментов (рисунок 38-2) повторяют наиболее востребованные в процессе работы действия, содержащиеся в меню, для быстрого доступа. Сюда включены следующие панели:

- Главная панель инструментов (повторяет меню «Файл»);

- Инструменты для работы с размещением (повторяет меню «Редактировать»);
- Некоторые другие инструменты.

Панели могут быть передвинуты пользователем в любую часть окна.

Обозреватель структуры схемы

Обозреватель структуры схемы (рисунок 38-3, 39) – окно, в котором отображается структура схемы, представленная в виде списка элементов, разделенных по типам целевых ячеек кристалла.

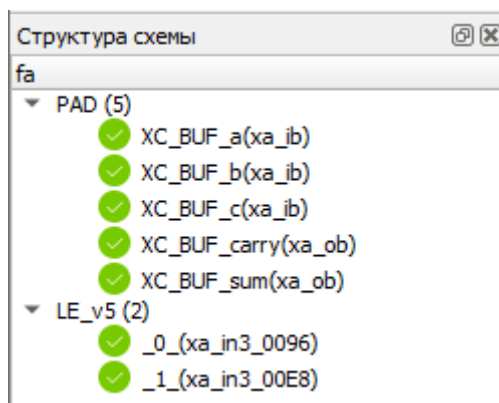


Рисунок 39 – Обозреватель структуры схемы

Слева от каждого элемента есть иконка, которая отражает состояние элемента:

| Иконка | Описание |
|--------|---|
| | Библиотечные данные для элемента загружены не были. Перед размещением следует сначала подгрузить данные библиотеки. |
| | Библиотечные данные для элемента загружены. Элемент готов к размещению. |
| | Библиотечные данные загружены. Элемент размещен |

Для каждого элемента схемы с помощью ПКМ может быть вызвано контекстное меню. Меню имеет следующие элементы:

- «Информация» – открывает окно с информацией об элементе и ячейке, к которой он привязан (если есть);
- «Отменить размещение» – отменяет назначение элемента ячейке;
- «Установить фокус» – устанавливает фокус на ячейке, в которую размещен элемент (если есть).

Также обозреватель поддерживает *механизм захвата и перетаскивания (Drag'n'Drop)*, с помощью которого можно назначить элемент на нужную ячейку.

Двойное нажатие ЛКМ на элементе схемы установит фокус подобно пункту меню «Установить фокус».

Обозреватель списка цепей

Обозреватель списка цепей (рисунок 38-4, 40) – окно, в котором отображается список цепей и откуда можно управлять их отображением.

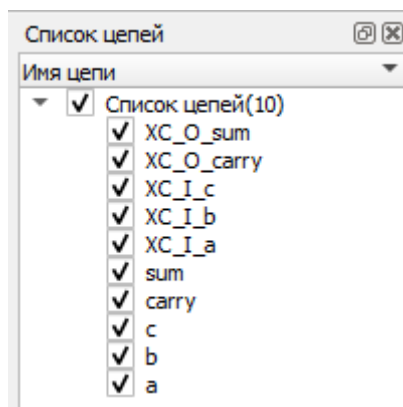


Рисунок 40 - Обозреватель списка цепей

Флаг рядом с цепью отвечает за ее *отображение в графическом окне*. Если флаг установлен, то цепь отображается, иначе – скрывается.

Флаг корневого элемента списка цепей влияет на все цепи.

При выделении одной или нескольких цепей, они выделяются в графическом окне (даже если флаг(и) отключен(ы)).

Рабочее графическое поле

Рабочее графическое поле (рисунок 38-5) – основное средство работы с размещением элементов. Оно имеет следующие возможности:

- Поддерживает механизм *Drag'n'Drop* для перемещения и назначения элементов в ячейки;
- *Масштабирование* посредством вращения колесика мыши;
- *Перемещение* обзора посредством перемещения мыши с зажатой клавишей СКМ (средней кнопки мыши);
- *Выделение* ячеек:
 - Выделение ячейки с помощью ЛКМ;
 - Выделение группы ячеек с помощью зажатой клавиши Ctrl и одиночного щелчка ЛКМ по необходимым ячейкам;
 - Выделение группы ячеек посредством движения мыши с зажатой ЛКМ;
 - Выделение всех ячеек с помощью горячей клавиши «Ctrl+A» (аналогично «Редактировать – Выделить все»);
- Двойной щелчок ЛКМ по ячейке открывает *информационное окно* для ячейки и размещенного элемента (если есть);
- *Всплывающие подсказки* с именами ячеек и размещенными в них элементами (если есть) при наведении мыши на ячейку.

Также для рабочего графического поля можно выставить различные настройки отображения (см. ниже «Настройки отображения»).

Окно «Tcl консоль»

Окно «Tcl консоль» (рисунок 38-6, 41) – окно в котором отображается информация о процессе размещения элементов на схеме. Окно также является консолью с поддержкой языка Tcl, где можно исполнять команды данного языка.

Окно обладает подсветкой синтаксиса, выделяя разными цветами информационные сообщения, предупреждения и ошибки.

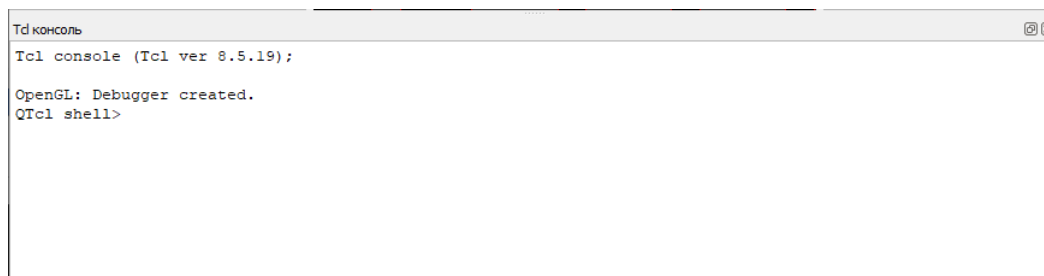


Рисунок 41 – Окно «Tcl консоль»

Статусная строка

Статусная строка (рисунок 38-7) необходима для отображения краткой информации о работе программы.

Пример работы программы

Для того чтобы показать, как работает программа, возьмем результаты работы ХА по схеме «fa» из примера работы программы X-CAD.

При запуске программы X-Place появляется главное окно (рисунок 42), в котором пока нет данных ни о кристалле, ни о схеме.

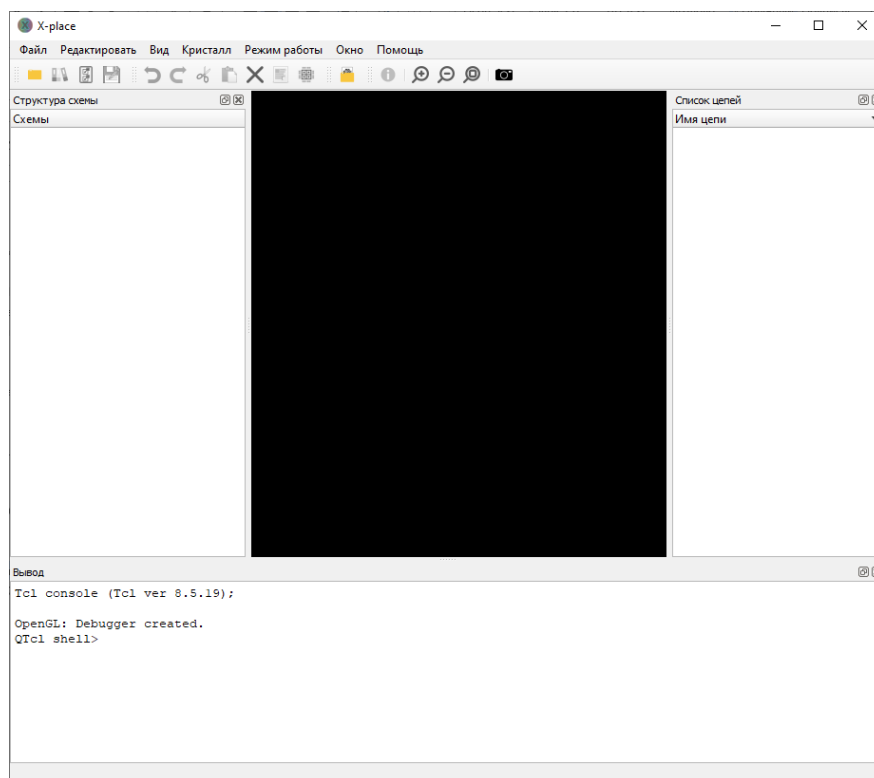


Рисунок 42 – Вид главного окна при запуске

Загрузка кристалла

Следующим шагом необходимо загрузить кристалл ПЛИС. Сделать это можно двумя способами:

1. открыть меню «Кристалл» из панели меню и выбрать необходимый кристалл, наведя указатель мыши на необходимое семейство кристаллов;
2. выбрать «Кристалл – Загрузить кристалл из index файла». Откроется окно файлового обозревателя, через который необходимо выбрать файл с расширением «*.index.tcl» (для того чтобы загрузить схему, рядом с этим файлом должен лежать файл с расширением «*.place.tcl») и нажать «Открыть».

После этого начнется загрузка кристалла, который в результате будет отображен в графической области главного окна (рисунок 43).

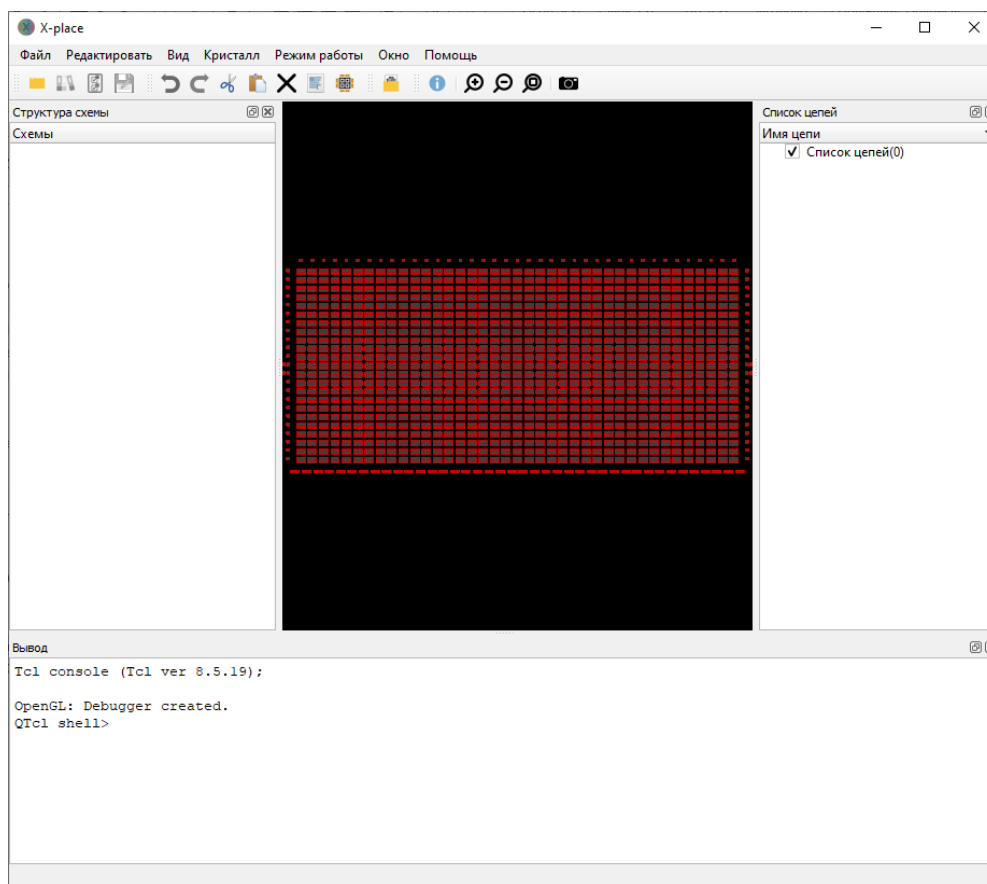


Рисунок 43 – Главное окно с загруженным графическим представлением кристалла ПЛИС

Загрузка схемы

Теперь следует загрузить схему, которая будет размещена на кристалле. Сделать это можно через меню «Файл – Открыть схему». Откроется окно файлового обозревателя, где следует выбрать файл схемы с расширением «*.tcl». Для примера, на рисунке 44 изображено окно после загрузки файла «fa.xmap.tcl» с описанием схемы «fa».

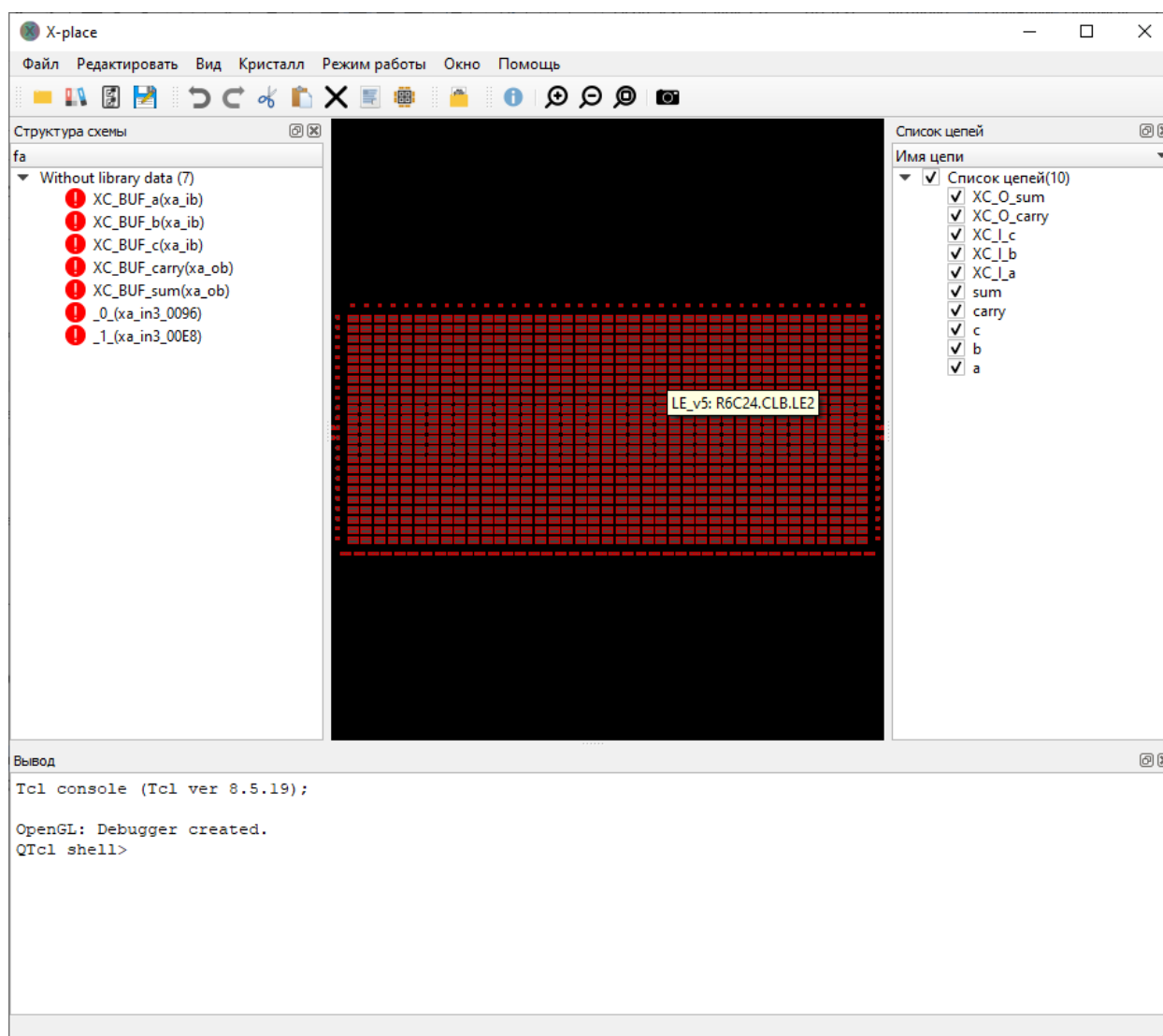


Рисунок 44 – Окно программы после загрузки схемы

Загрузка библиотеки элементов

Если рядом с файлом лежит библиотека элементов (в терминах ХА и расширением «*.xmap.lib.tcl»), то данные из неё будут автоматически подгружены программой после того, как пользователь подтвердит действие в информационном окне (например, для данной схемы – «fa.xmap.lib.tcl»), элементы будут объединены в группы по типу целевого блока и слева от имени элемента будет отображаться серый значок, означающий, что элемент пока не размещен на кристалле (рисунок 45б). Если же библиотека не была найдена или пользователь отказал в загрузке, то данные подгружены не будут, программа

предупредит об этом, все элементы будут объединены в группу «Without library data» и рядом с элементами будет отображаться красный значок (рисунок 45а).

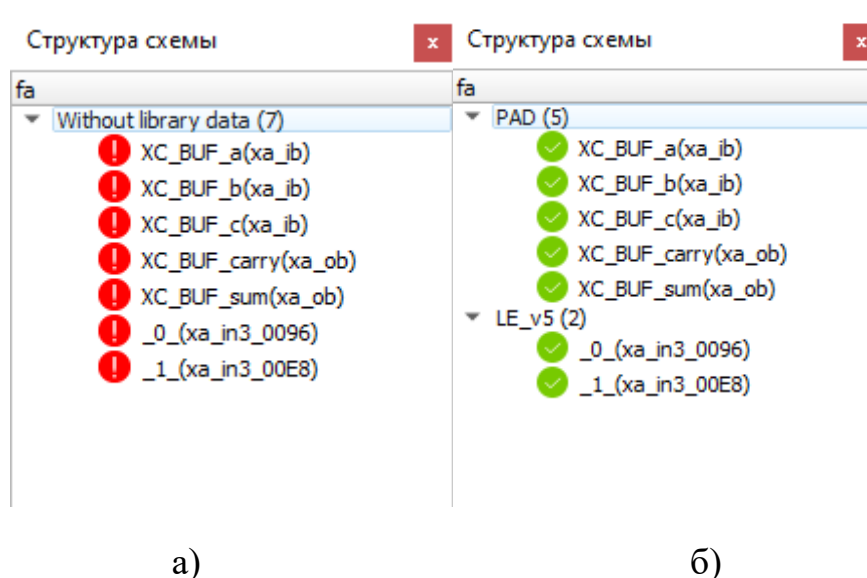


Рисунок 45 – обозреватель структуры: а) – до загрузки библиотечных данных; б) – после загрузки данных

Если после загрузки библиотеки все еще остались элементы без данных, то программа предложит подгрузить данные из стандартных библиотек.

Элементы, которые не имеют библиотечных данных *невозможно* будет разместить на кристалле.

Библиотечные данные можно подгрузить и из сторонней библиотеки. Для этого необходимо выбрать «Файл – Загрузить библиотеку», через окно обозревателя файлов выбрать необходимый файл библиотеки и нажать «Открыть».

Размещение элементов на кристалле

После успешной загрузки схемы и в случае отсутствия элементов без библиотечных данных, программа предпримет попытку автоматического поиска файлов размещения (с расширением «*.place.tcl»). Если файл размещения был

найден, то программа запросит разрешение на его загрузку и, в случае положительного ответа пользователя, данные размещения будут загружены.

После загрузки схемы можно приступить к ручному размещению элементов или загрузить уже готовый файл размещения для его редактирования. Для загрузки размещения необходимо выбрать «Файл – Загрузить размещение» и через обозреватель выбрать файл размещения (часто с расширением «*.place.tcl»). Для примера был открыт файл «fa.place.tcl» (рисунок 46).

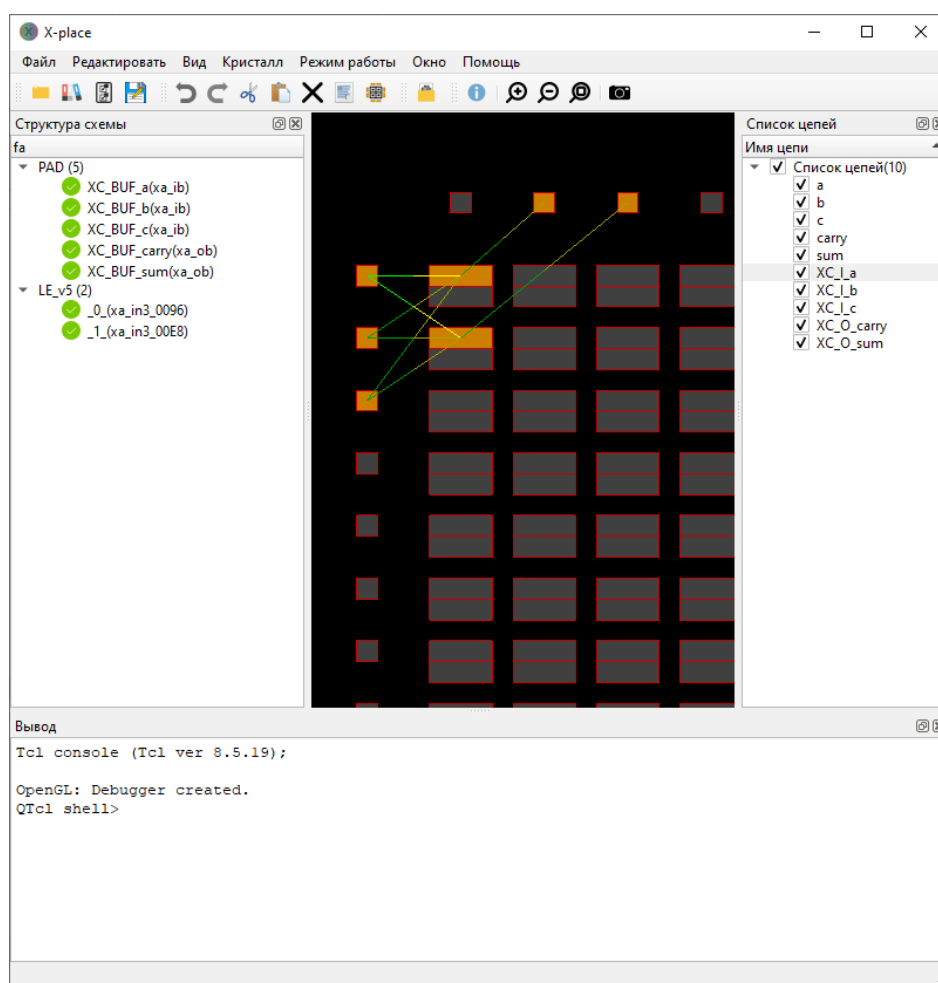


Рисунок 46 – Окно с загруженными данными размещения схемы «fa»

После того, как элементы будут расположены на кристалле, занятые ячейки будут окрашены в оранжевый цвет, а значки в списке элементов поменяют цвет с серого на зеленый. Также будет сгенерирована система межсоединений элементов в зелено-жёлтой окраске: жёлтая часть соединения – цепь для этого элемента является выходной, зеленая – входной.

Если при загрузке размещения некоторые элементы уже оказываются размещены, появится информационное окно с вопросом о том, что хочет сделать пользователь. Тут имеются три варианта:

- «Продолжить» – элементы со схожими именами будут перемещены, а все остальные останутся на местах;
- «Очистить и продолжить» – текущее размещение полностью очищается, чтобы произвести «чистое» размещение;
- «Отмена» – загрузка осуществлена не будет.

Сохранение полученного размещения

В случае необходимости (ручное изменение размещения логических элементов, ячеек ввода/вывода, макроблоков) размещение элементов можно сохранить в отдельный файл для дальнейшего использования при проектировании в программе X-CAD.

Данная функция доступна в меню «Файл» (рисунок 47) и поддерживает сохранение следующих типов файлов:

- «*.place.tcl» – размещение всей схемы (рисунок 47-1);
- «*.inout.tcl» – размещение ячеек ввода/вывода(рисунок 47-2);
- «*.macro.tcl» – размещение макроблоков(рисунок 47-3).
- «*.ip.tcl» – размещение IP-блоков(рисунок 47-4).

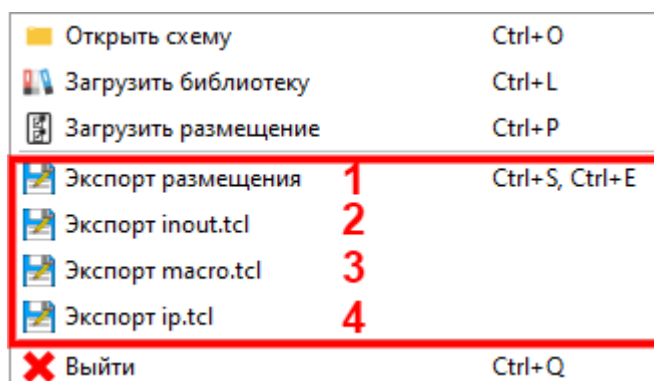


Рисунок 47 – Функция экспорта размещения всей схемы, ячеек ввода/вывода или макроблоков

Настройки отображения

Общие настройки отображения («Вид – Отображаемые объекты») позволяют настроить (включить/отключить) отображение:

- *Заполненные ячейки*;
- *Кристалл* (не затрагивает занятые ячейки);
- *Цепи* (не затрагивает отображение цепей выделенных ячеек);
- *Подсвеченные цепи выделенных объектов*.

Настройки режима отображения цепей (пример на рисунок 48) позволяют выбрать один из двух режимов отображения:

- *Прямые соединения* – из каждого размещенного элемента выходят соединяющие линии напрямую к элементам, связанным с данным;
- *Центрированные соединения* – межсоединение представляет собой группу линий с общим центром масс, откуда эти линии выходят по направлению к подсоединенным элементам.

Настройки режима отображения подсвеченных цепей позволяет выбрать режим подсвечивания соединений выделенных объектов:

- *Только подсоединенные элементы* – подсвечиваются только те линии, которые связывают данный элемент с другими;
- *Полностью подсветить цепь* – межсоединение подсвечивается полностью, то есть не только соединения данного элемента с другими, но и все другие направления, куда распространяется это межсоединение.

Динамическое выделение рамкой подразумевает отображение результата выделения сразу после изменения положения выделяющей рамки, а не после того, как пользователь отпустит кнопку мыши (ВНИМАНИЕ! На больших схемах сильно снижает производительность).

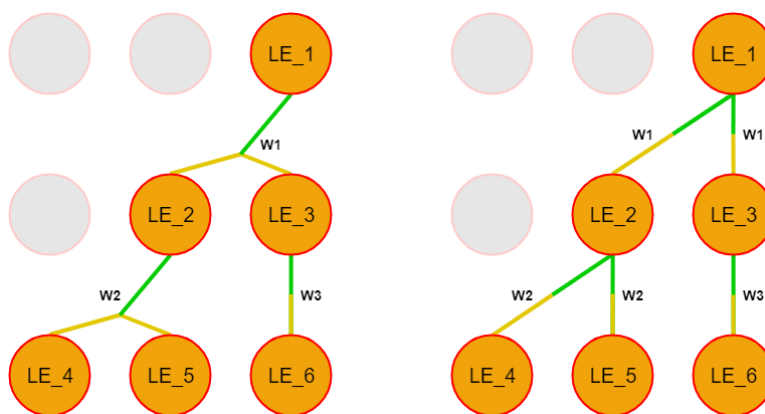


Рисунок 48 – Способы представления цепей (пример на регулярной структуре ПЛИС): центрированный (слева) и прямой (справа)

ИНТЕРФЕЙС ПЛАНИРОВКИ КРИСТАЛЛА FLOORPLANNER

FloorPlanner – режим программы X-Place, предназначенный для ручного размещения и настройки периферийных элементов, макроблоков и IP-блоков на кристалле ПЛИС до этапа физического синтеза схемы.

Запуск программы

Запуск программы может быть осуществлен посредством запуска исполняемого файла через обозреватель файлов, из командной строки или через соответствующее окно программы X-CAD.

Запуск через обозреватель файлов

Для запуска через обозреватель файлов необходимо открыть папку с исполняемым файлом «x-place.exe», который расположен в директории «bin» пакета программ, и активировать двойным щелчком левой кнопки мыши или вызвать с помощью правой кнопки мыши контекстное меню и выбрать соответствующий пункт для запуска приложения. Затем необходимо в меню программы переключить режим работы на режим планировки кристалла («Режим работы» - «FloorPlanner»).

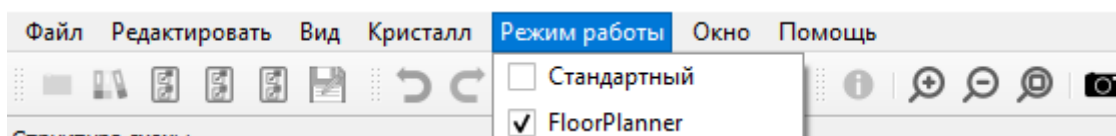


Рисунок 49 – Смена режима работы через меню программы

Запуск через командную строку

Чтобы запустить программу посредством командной строки, необходимо перейти в папку «bin» пакета программ и вызвать на исполнение файл «x-place.exe», используя следующие команды:

- Windows: “start x-place.exe -pp <аргументы_и_значения>”
- Unix: “./x-place.exe -pp <аргументы_и_значения>”

Программа поддерживает следующие аргументы командной строки:

- -i «файл» – описание схемы (на языке Verilog);

- *-l «файл»* – библиотека элементов схемы (на языке Tcl);
- *-p «файл»* – файл размещения (на языке Tcl, *.inout.tcl, *.macro.tcl);
- *-c «имя_кристалла»* – данные кристалла по имени, которое соответствует имени кристалла в файле «ха_config.ini».
- *-pp* – запуск программы в режиме планировки кристалла;
- *-top* – имя топ-модуля.

Запуск через интерфейс X-CAD

Для запуска программы необходимо выбрать соответствующий пункт меню программы X-CAD («Инструменты» - «FloorPlanner») или кнопку на панели инструментов, затем, в появившемся окне (рисунок 50) нажать на кнопку «Чистый запуск» или «Запустить». Нажатие на кнопку «Запустить» приведет к запуску программы с передачей в качестве аргументов всех Verilog-файлов, лежащих в разделе «HDL-описания», а также топ-модуля и используемого кристалла. Также можно передать inout-файл, содержащий размещение I/O элементов.

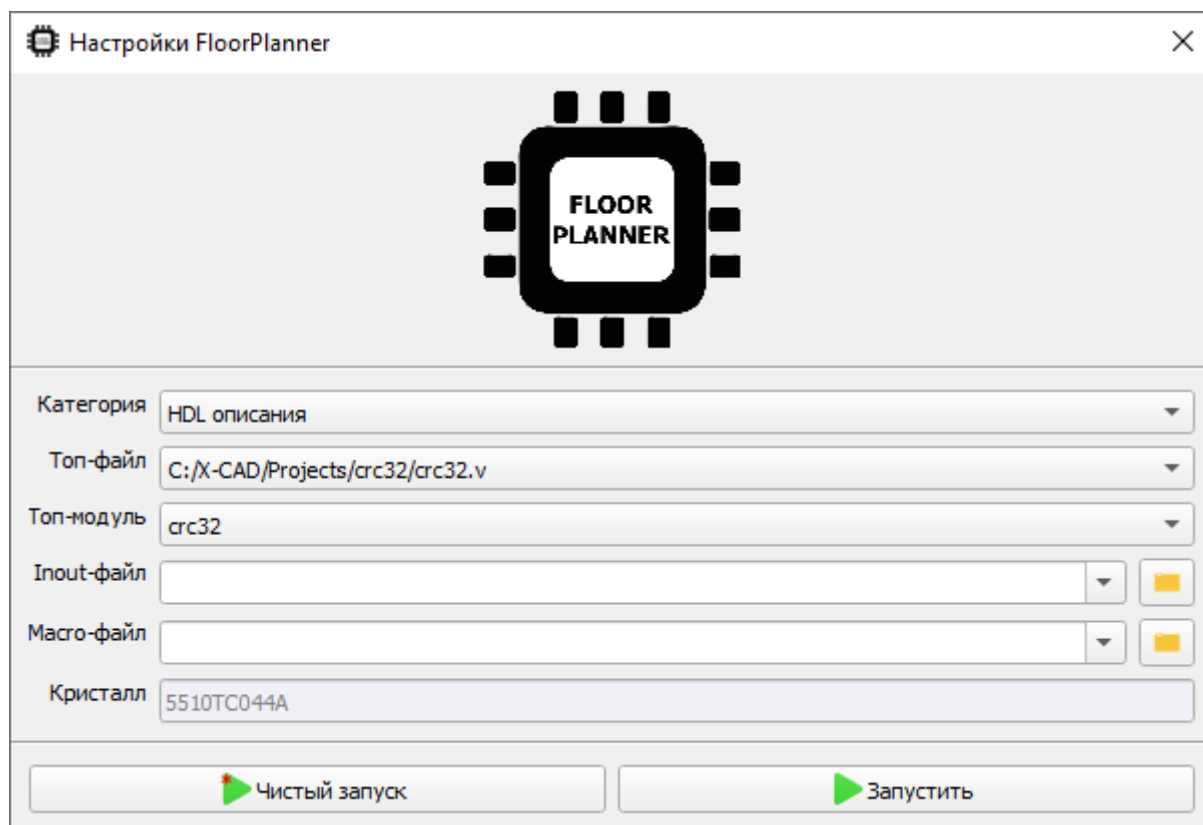


Рисунок 50 – Окно запуска X-Place в режиме FloorPlanner

Интерфейс программы

Окно программы представлено на рисунок 51 и содержит следующие элементы интерфейса:

- *Главное меню* (рисунок 51-1);
- *Панели инструментов* (рисунок 51-2);
- *Обозреватель элементов* (рисунок 51-3);
- *Рабочее графическое поле* (рисунок 51-4);
- *Окно «Tcl консоль»* (рисунок 51-5);
- *Статусная строка* (рисунок 51-6).

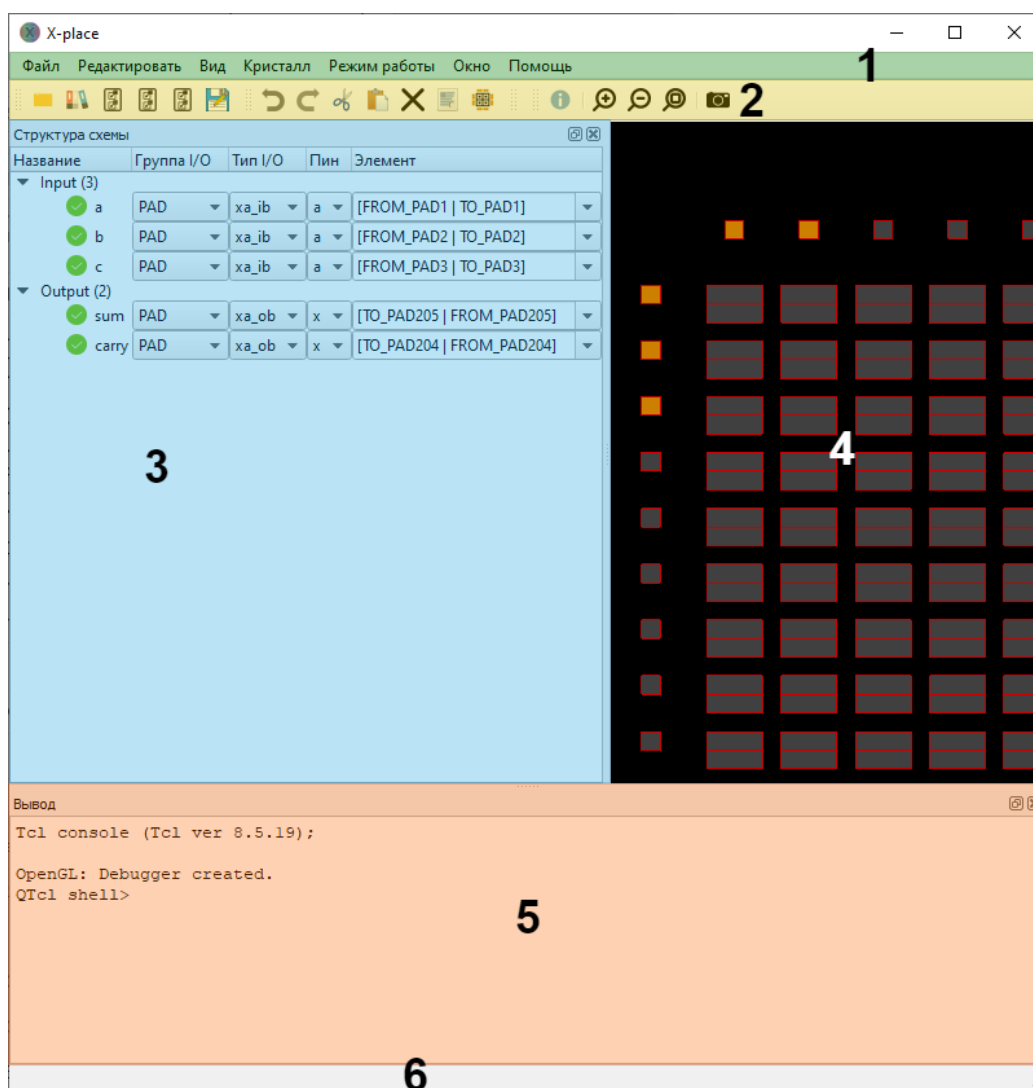


Рисунок 51 – Структура главного окна программы в режиме FloorPlanner:

1 – главное меню; 2 – панели инструментов; 3 – обозреватель элементов;

4 – рабочее окно; 5 – окно «Tcl консоль»; 6 – статусная строка

Главное меню

Главное меню – панель с дочерними меню, содержащими основные действия для работы с программой. Меню содержит следующие дочерние меню:

- «Файл» – открытие описания схемы на языке Verilog, сохранение текущего размещения элементов, загрузка размещения, завершение работы программы;
- «Редактировать» – удаление выбранных размещенных элементов, выделение всех элементов;
- «Вид» – позволяет:
 - управлять графическим отображением кристалла;
 - Устанавливать режим отображения межсоединений;
 - Устанавливать режим отображения межсоединений выделенных объектов;
 - Включать/отключать динамическое выделение объектов на кристалле;
- «Кристалл» – выбор целевого кристалла, загрузка кристалла из index.tcl файла;
- Режим работы – переключение режимов работы: стандартный или FloorPlanner;
- «Окно» – настройка шрифта приложения, выбор языка;
- «Помощь» – позволяет получить вспомогательную информацию (находится в разработке).

Панели инструментов

Панели инструментов повторяют наиболее востребованные в процессе работы действия, содержащиеся в меню, для быстрого доступа. Сюда включены следующие панели:

- Главная панель инструментов (повторяет меню «Файл»);
- Инструменты для работы с размещением (повторяет меню «Редактировать»);
- Панель дополнительных инструментов.

Панели могут быть перемещены пользователем в любую часть окна.

Обозреватель элементов

Обозреватель элементов (рисунок 52) – окно, в котором отображаются входные/выходные сигналы, а также макроблоки и IP-блоки, используемые в Verilog-описаниях.

| Структура схемы | | | | |
|-----------------|------------|---------|-----|---------------------------|
| Название | Группа I/O | Тип I/O | Пин | Элемент |
| ▼ INPUT (3) | | | | |
| ✓ a | PAD | xa_ib | a | [FROM_PAD2 TO_PAD2] |
| ✓ b | PAD | xa_ib | a | [FROM_PAD3 TO_PAD3] |
| ✓ c | PAD | xa_ib | a | [FROM_PAD1 TO_PAD1] |
| ▼ OUTPUT (2) | | | | |
| ✓ sum | PAD | xa_ob | x | [TO_PAD204 FROM_PAD204] |
| ✓ carry | PAD | xa_ob | x | [TO_PAD203 FROM_PAD203] |

Рисунок 52 – Обозреватель структуры схемы

В данном окне для сигналов может быть выбран тип элемента, который должен быть использован для данного сигнала, а также ячейка на кристалле, в которую следует разместить этот элемент, и пин, к которому необходимо подключить сигнал.

Над элементами данного окна доступны те же операции, что и над элементами обозревателя структуры схемы «Стандартного» режима, описанного ранее.

Рабочее графическое поле

Рабочее графическое поле – основной инструмент планировки кристалла.

Поле имеет следующие возможности:

- Поддерживает механизм *Drag'n'Drop* для перемещения и назначения элементов в ячейки;
- *Масштабирование* посредством вращения колесика мыши;
- *Перемещение* обзора посредством перемещения мыши с зажатой клавишей СКМ (средней кнопки мыши);
- *Выделение* ячеек:
 - Выделение ячейки с помощью ЛКМ;
 - Выделение группы ячеек с помощью зажатой клавиши Ctrl и одиночного щелчка ЛКМ по необходимым ячейкам;
 - Выделение группы ячеек посредством движения мыши с зажатой ЛКМ;
 - Выделение всех ячеек с помощью горячей клавиши «Ctrl+A» (аналогично «Редактировать – Выделить все»);
- Двойной щелчок ЛКМ по ячейке открывает *информационное окно* для ячейки, которое также позволяет назначить элемент в данную ячейку;
- *Всплывающие подсказки* с именами ячеек и размещенными в них элементами (если есть) при наведении мыши на ячейку.

Окно «Tcl консоль»

Окно «Tcl консоль» – окно в котором отображается информация о процессе размещения элементов на схеме. Окно также является консолью с поддержкой языка Tcl, где можно исполнять команды данного языка.

Статусная строка

Статусная строка необходима для отображения краткой информации о статусе работы программы.

Пример работы программы в режиме FloorPlanner

Работа программы в режиме FloorPlanner будет продемонстрирована на схеме fa.

При запуске программы X-Place пользователя встречает главное окно (рисунок 53). Если программа была запущена без дополнительных аргументов, то рабочее поле и окно «Структура схемы» будут пустыми.

Если программа была запущена без аргумента «-pp», то необходимо переключиться в режим «FloorPlanner» («Режим работы – FloorPlanner»).

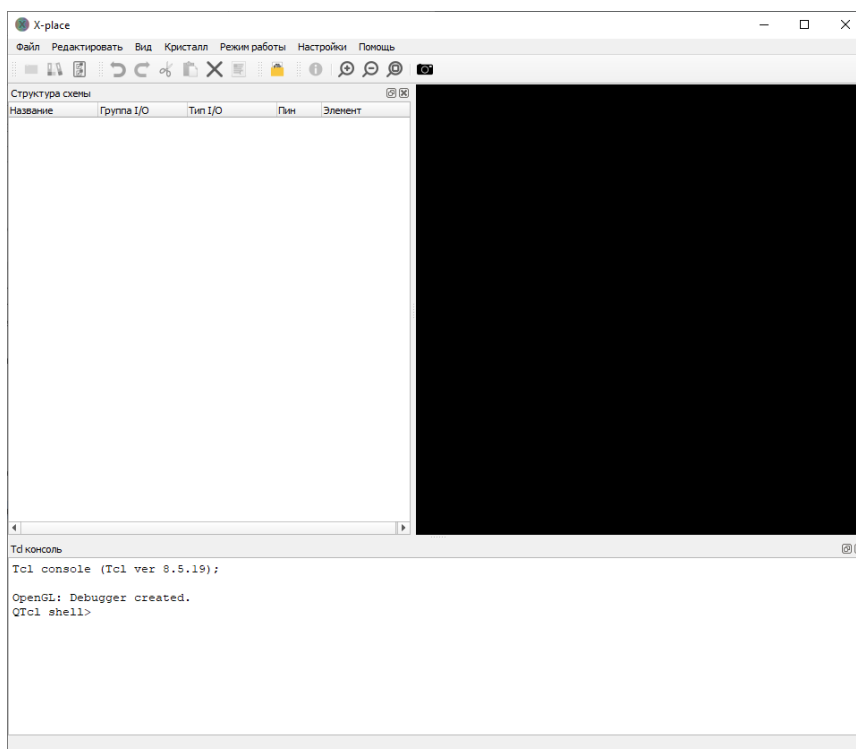


Рисунок 53 – Вид главного окна при запуске

Загрузка кристалла

Следующим шагом необходимо загрузить кристалл ПЛИС. Сделать это можно следующим образом: открыть меню «Кристалл» из панели меню и выбрать необходимый кристалл, где кристаллы разбиты по семействам. После этого начнется загрузка ячеек схемы, которые в результате будут отображены в графической области главного окна (рисунок 54). После завершения данного процесса автоматически будут загружены стандартные библиотеки целевого кристалла ПЛИС.

В том случае, если для загрузки доступен ровно один кристалл ПЛИС, то при старте программы, данный кристалл будет загружен автоматически.

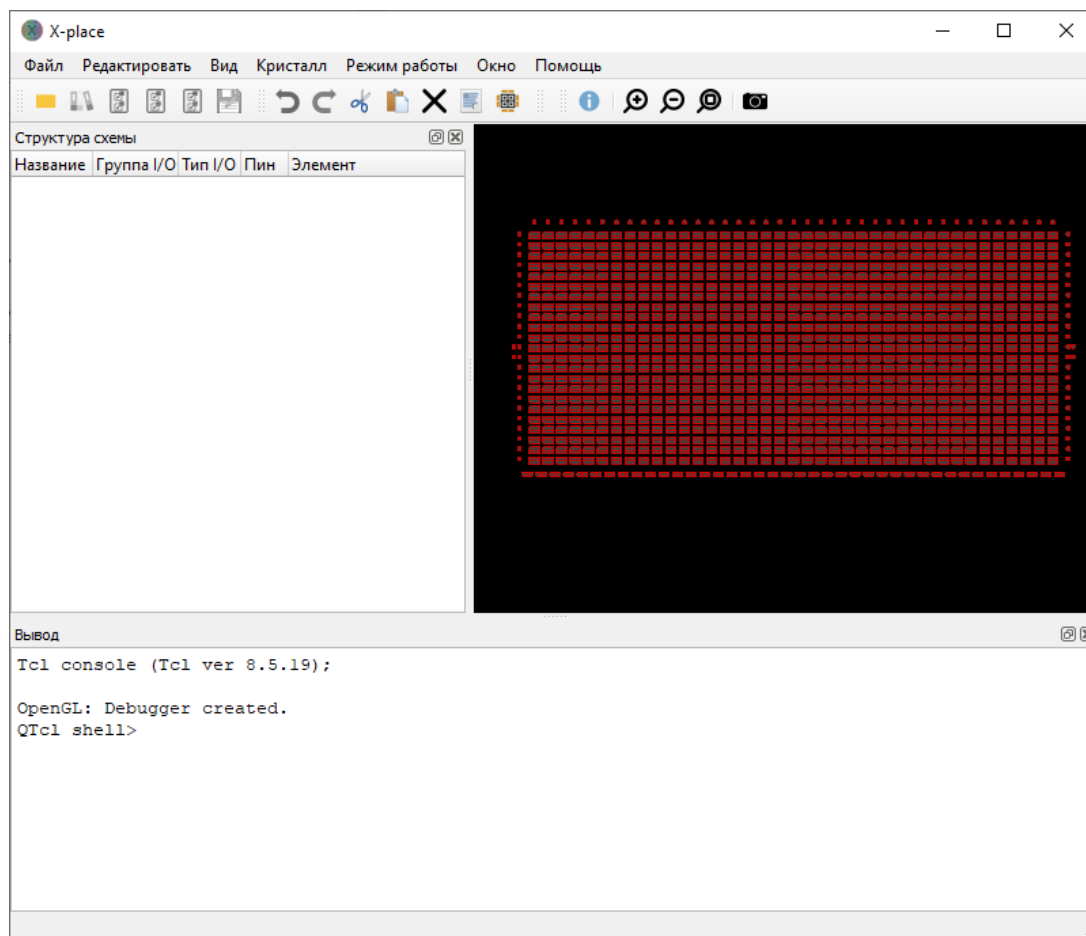


Рисунок 54 – Главное окно с загруженным графическим представлением кристалла ПЛИС

Загрузка схемы

Теперь следует загрузить Verilog-описание, элементы которого необходимо разместить. Сделать это можно через меню «Файл – Открыть схему». Откроется окно файлового обозревателя, где следует выбрать файлы с расширением «*.v». Следом за этим откроется окно, в котором необходимо выбрать какой из выбранных файлов и какой модуль в выбранном файле будут главными (рисунок 55).

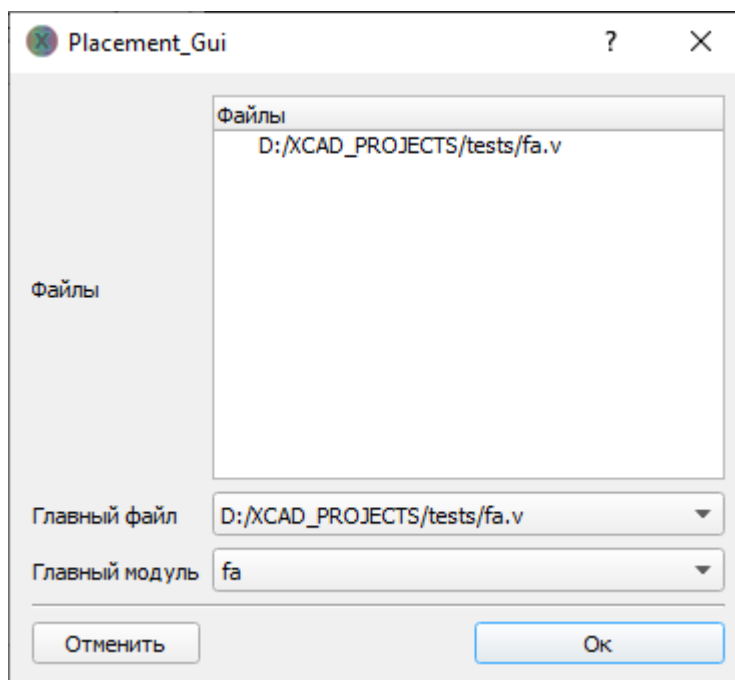


Рисунок 55 – Окно выбора главного файла и главного модуля

Для примера, на рисунок 56 изображено окно после загрузки файла «fa.v» с описанием схемы «fa» в базисе выбранной ПЛИС.

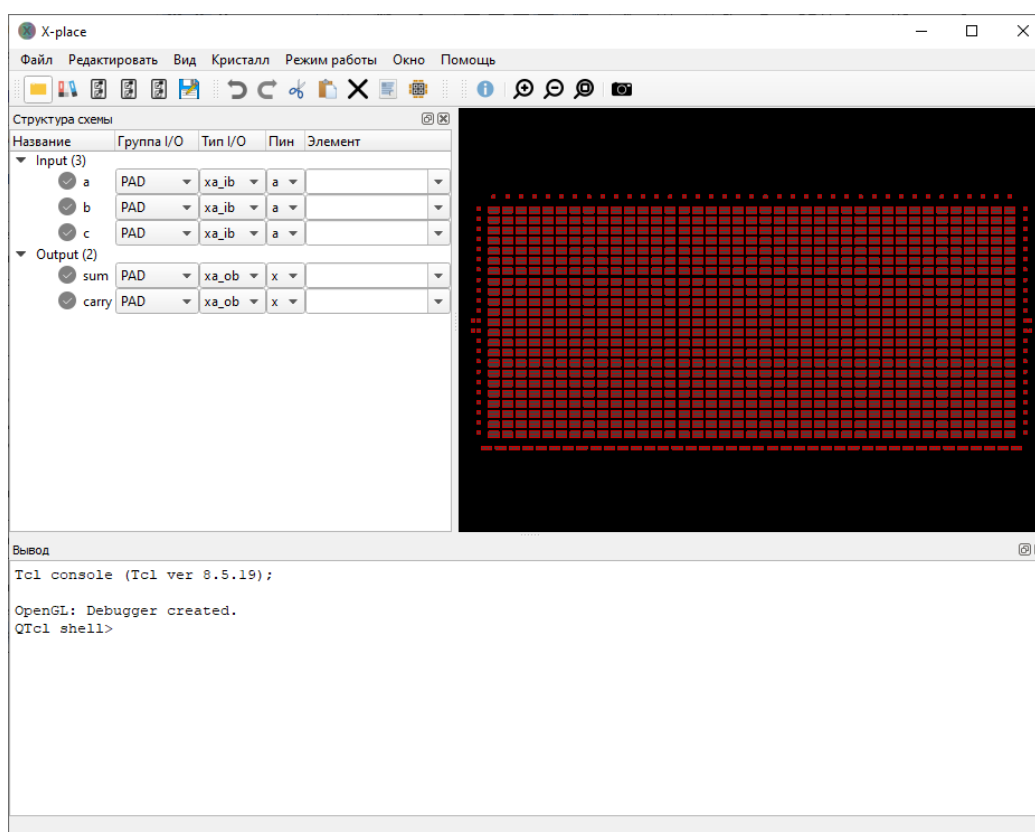


Рисунок 56 – Окно программы после загрузки схемы

Загрузка библиотеки элементов

Если Verilog-описание использует элементы, неописанные в стандартных библиотеках кристалла, то данные элементы будут недоступны для использования. Для того, чтобы появилась возможность использования данных элементов, необходимо подгрузить данные элементы из сторонней библиотеки. Для этого необходимо выбрать «Файл – Загрузить библиотеку», через окно обозревателя файлов выбрать необходимый файл библиотеки и нажать «Открыть».

Размещение элементов на кристалле

После загрузки Verilog-описания можно приступать к размещению элементов или загрузить файл, содержащий размещение элементов, в случае его наличия для текущей схемы. Элементы, имена которых совпадают с названиями имен I/O ячеек будут размещены автоматически. Для загрузки размещения необходимо выбрать «Файл – Загрузить inout-файл» и через обозреватель выбрать файл с расширением «*.inout.tcl». Аналогичные действия могут быть проделаны для загрузки размещения macro-блоков и IP-блоков.

Для ручного размещения элементов существует несколько способов:

- Выбрать необходимую ячейку в обозревателе структуры схемы;
- Перетащить элемент на необходимую ячейку с помощью механизма Drag'n'Drop;
- Назначить элемент на ячейку из информационного окна ячейки.

Также можно воспользоваться опцией «Автоматическое размещение I/O» нажав на соответствующую кнопку в панели инструментов или в меню «Редактировать». После нажатия на данную кнопку каждому периферийному элементу будет назначена подходящая по типу ячейка кристалла.

После того, как элементы будут расположены на кристалле, занятые ячейки будут окрашены в оранжевый цвет, а значки в списке элементов поменяют цвет с серого на зеленый (рисунок 57).

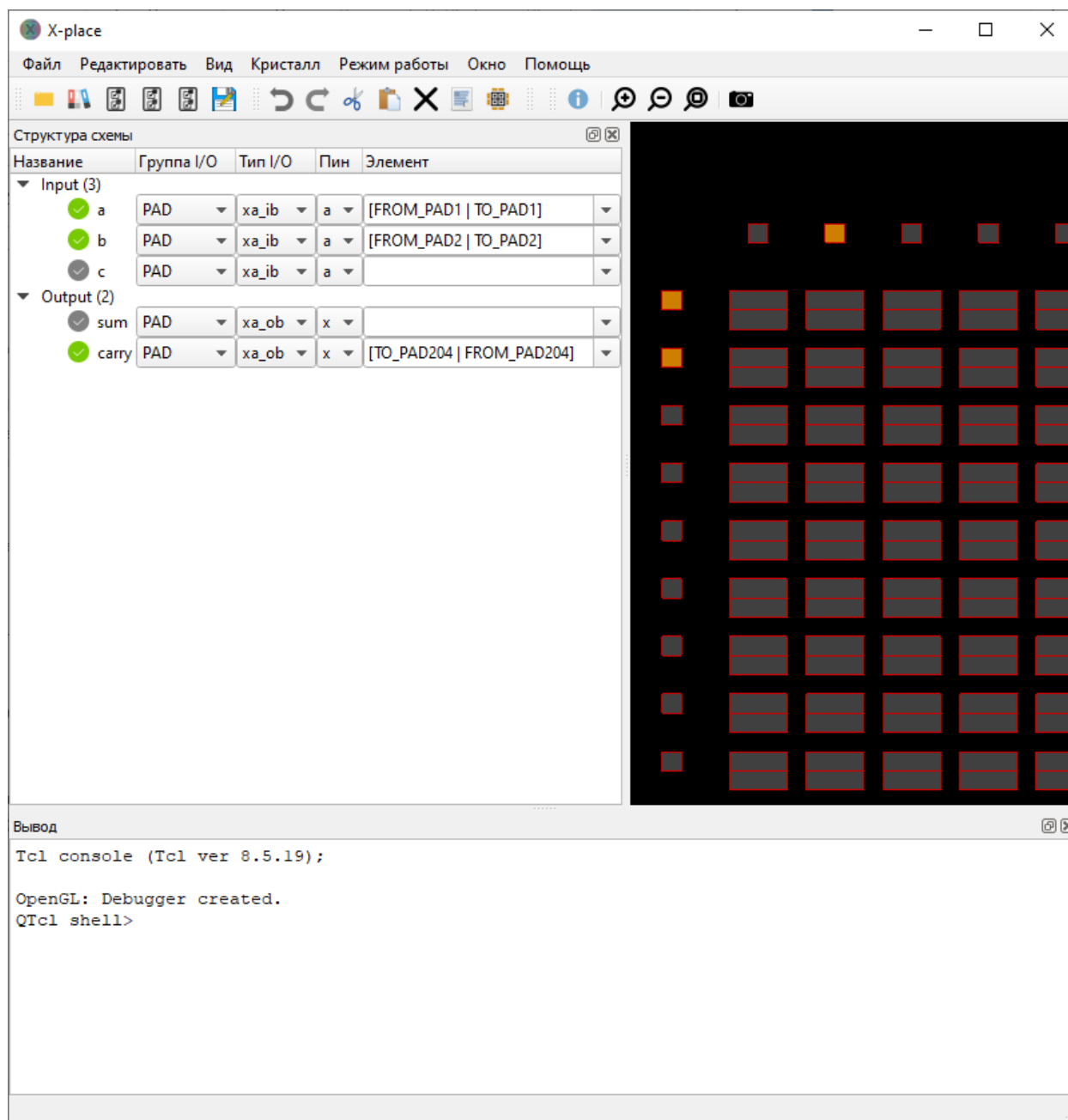


Рисунок 57 – Окно с некоторым количеством размещенных периферийных элементов

Сохранение полученного размещения

Для сохранения полученного размещения необходимо выбрать соответствующую опцию в меню «Файл» (рисунок 58). Режим FloorPlanner поддерживает следующие типы файлов:

«*.inout.tcl» – размещение ячеек ввода/вывода (рисунок 58-1);

«*.macro.tcl» – размещение макроблоков (рисунок 58-2).

«*.ip.tcl» – размещение IP-блоков (рисунок 58-3).

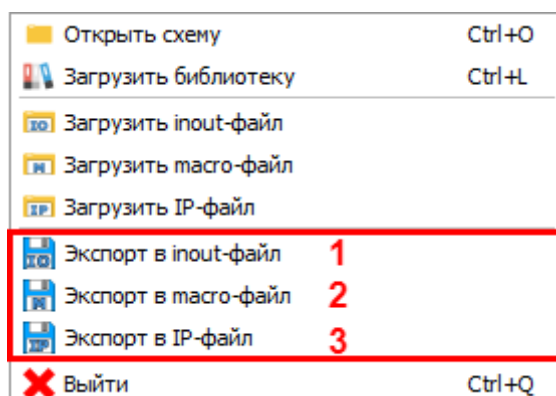


Рисунок 58 – Функция экспорта размещения всей схемы, ячеек ввода/вывода или макроблоков

Использование полученного размещения

Для использования полученных файлов размещения необходимо в окне «Настройки запуска» выбрать меню «Размещение элементов», активировать требуемую опцию, изменив значение «Нет» на «Да», и добавить сгенерированный файл в соответствующее поле ввода (Рисунок 59).

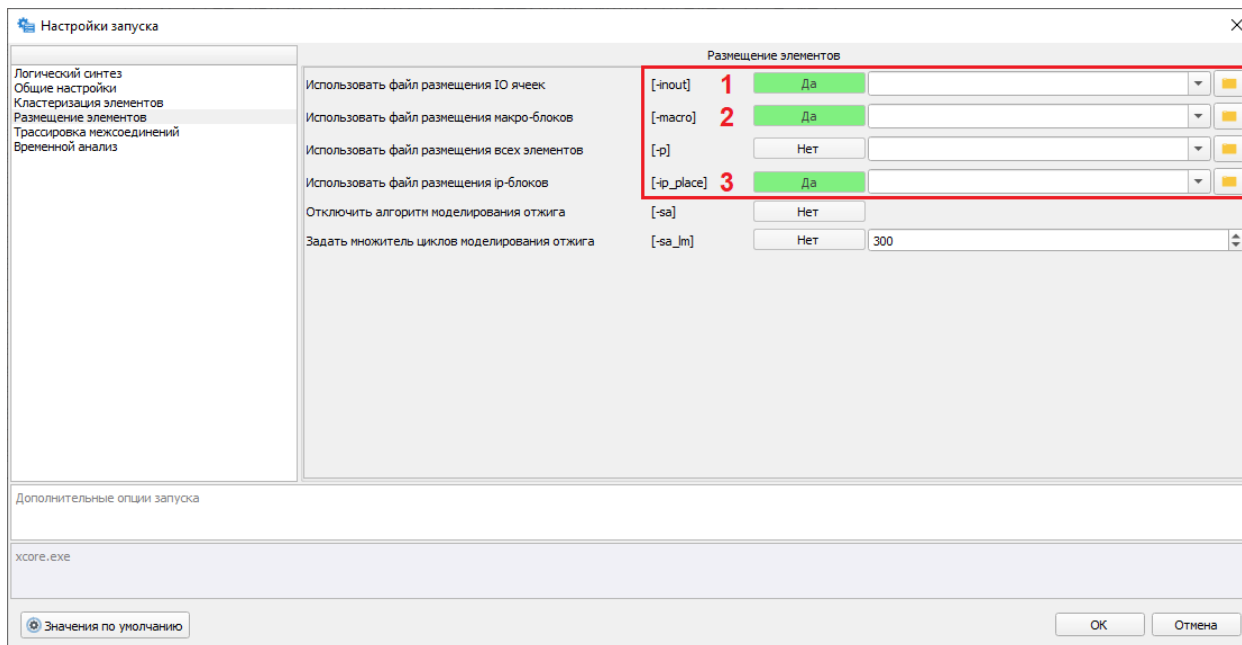


Рисунок 59 – Настройки размещения элементов

Лист регистрации изменений

[illegible]