

## ГЛАВА 3

# ИНСТРУМЕНТАЛЬНАЯ СРЕДА ACTIVE-HDL

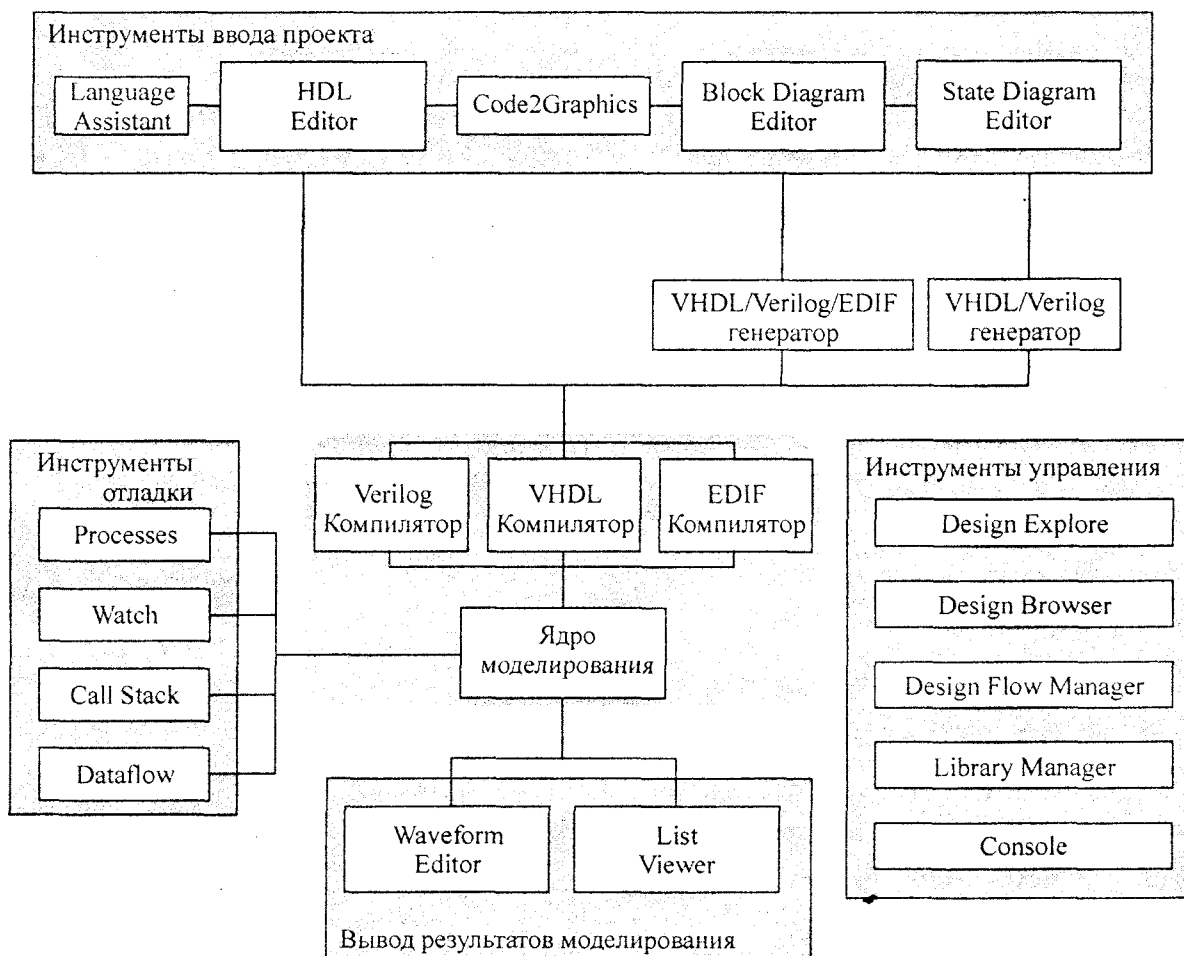
Введение в инструментальную среду проектирования Active-HDL. Представлены ее основные функции. Рассмотрена возможность управления инструментами среды Active-HDL с помощью макрокоманд.

### 3.1. Введение в Active-HDL

Active-HDL – это интегрированная среда проектирования. Она включает три различные программы для ввода проектов, VHDL'93 и Verilog компиляторы, единое ядро моделирования, программные модули отладки, графический и текстовый вывод результатов моделирования, вспомогательные инструменты для удобного управления файлами ресурсов, проектами и библиотеками (рисунок 3.1).

В Active-HDL включены также несколько мощных мастеров для облегчения создания новых исходных файлов, проектов и TestBench. Большинство операций, которые вызываются через графический пользовательский интерфейс, можно выполнять с помощью команд макроязыка Active-HDL. Командные файлы позволяют автоматизировать процесс проектирования.

Рисунок 3.1. Структурная схема элементов Active-HDL



## Компоненты Active-HDL

Все компоненты Active-HDL являются встроенными в интерактивную графическую среду. Каждый инструмент Active-HDL, за исключением ядра моделирования и компилятора, реализован в отдельном окне.

*Console (Alt-4)* – интерактивное окно для ввода макрокоманд и определенных пользователем скриптов, а также для вывода сообщений, генерируемых инструментами Active-HDL.

*Design Explorer* – облегчение управления проектами в Active-HDL.

*Design Browser* – отображение текущего содержания проекта: файлы ресурсов, присоединенных к проекту; состав файлов рабочих библиотек; структура выбранного для моделирования модуля проекта; объявленные в выбранной области текущего проекта VHDL, Verilog или EDIF объекты.

*Design Flow Manager* – автоматизация процесса проектирования. В графической форме представлен типичный процесс проектирования, а встроенные кнопки позволяют вызывать соответствующие данному этапу приложения.

*HDL Editor* – текстовый редактор, разработанный для создания исходных файлов VHDL. Он интегрирован с моделирующим устройством, что облегчает отладку исходного текста.

*Language Assistant* – вспомогательный инструмент, предоставляющий ряд типичных для VHDL или Verilog шаблонов, модели логических примитивов и функциональных блоков. Интегрирован с HDL Editor, что позволяет автоматически размещать выбранный шаблон в редактируемый файл. Допускает определение собственных шаблонов.

*State Diagram Editor (редактор автоматов)* – графический инструмент, разработанный для редактирования графов конечных автоматов. Осуществляет автоматический перевод графически задаваемых примитивов в коды языка описания аппаратуры VHDL.

*Waveform Editor* – отображает результаты моделирования в графической форме. Позволяет редактировать форму сигнала для создания требуемых тестовых векторов.

*Block Diagram Editor* – графический инструмент проектирования, позволяющий создавать структурные модели цифровых устройств. Выполняет автоматическое преобразование графического представления проекта в код VHDL или Verilog языков описания аппаратуры.

*List* – представление результатов моделирования в виде текстовой таблицы с точностью до дельта-цикла.

*Watch (Alt-5)* – вывод во время моделирования текущих значений выбранных сигналов и переменных.

*Processes (Alt-6)* – отображение во время моделирования текущего состояния параллельных процессов проекта.

*Call Stack (Alt-7)* – вывод списка подпрограмм (процедур и функций), вызванных в текущем процессе.

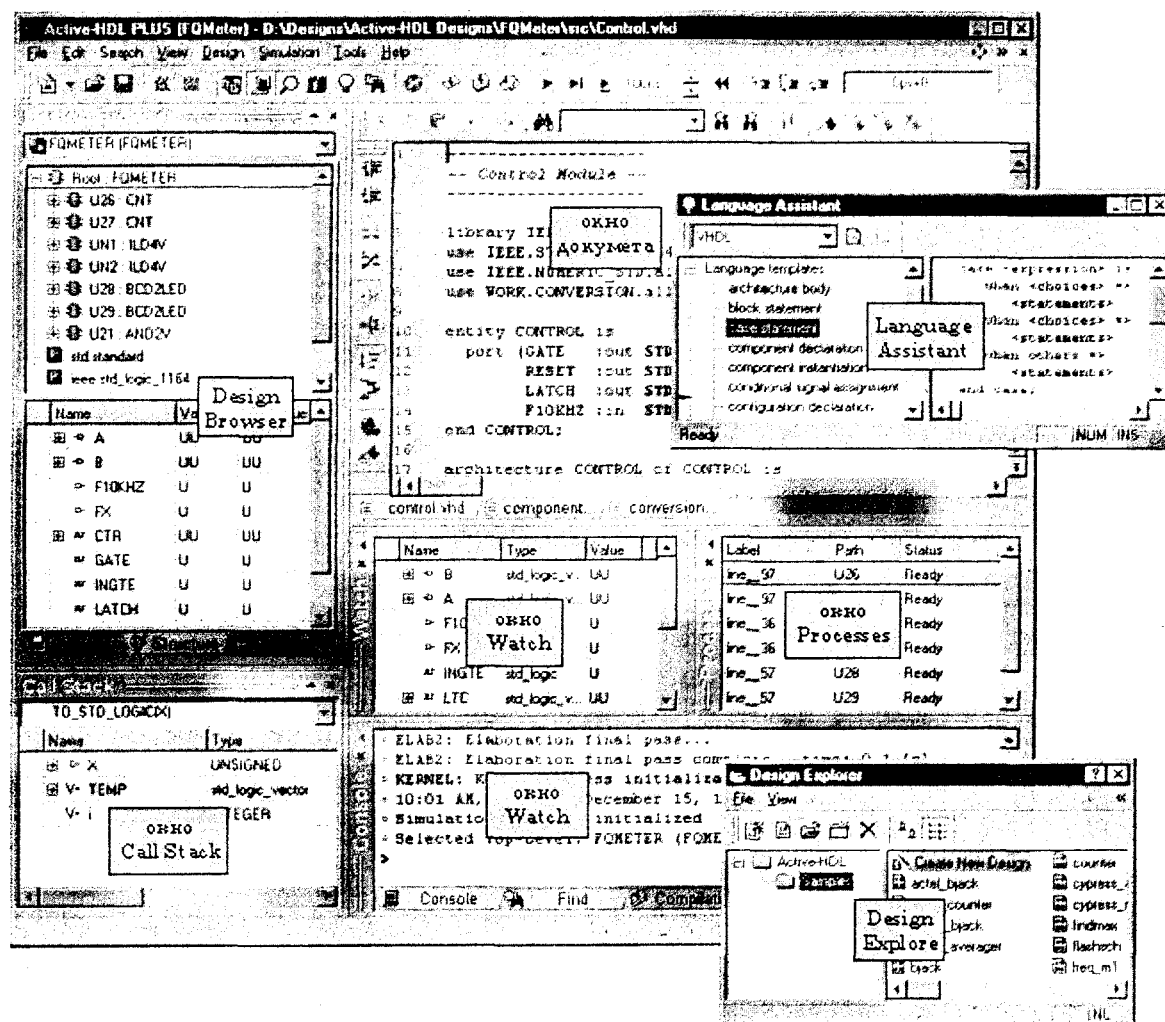
*Dataflow (Alt-9)* – графическое представление сигналов, входящих и выходящих из процессов во время моделирования.

*Library Manager* – управление VHDL-библиотеками и их содержанием.

*Code2Graphics™ Converter* предназначен для автоматического преобразования VHDL, Verilog или EDIF исходных файлов в Active-HDL структурную схему (block diagram) или граф (state diagram).

На рисунке 3.2 представлено задаваемое по умолчанию расположение на рабочем поле Active-HDL компонентов.

Рисунок 3.2. Рабочее окно программы Active-HDL





### 3.2. Design Browser

Design Browser (браузер проекта) предназначен для управления ресурсами текущего проекта. Состоит из трех подокон (вкладок): Files, Structure и Resources. Позволяет добавлять, удалять, просматривать, модифицировать и выполнять другие операции над файлами проекта. Выводит содержимое рабочей, post-synthesis и timing библиотек текущего проекта.

#### Подокно Files

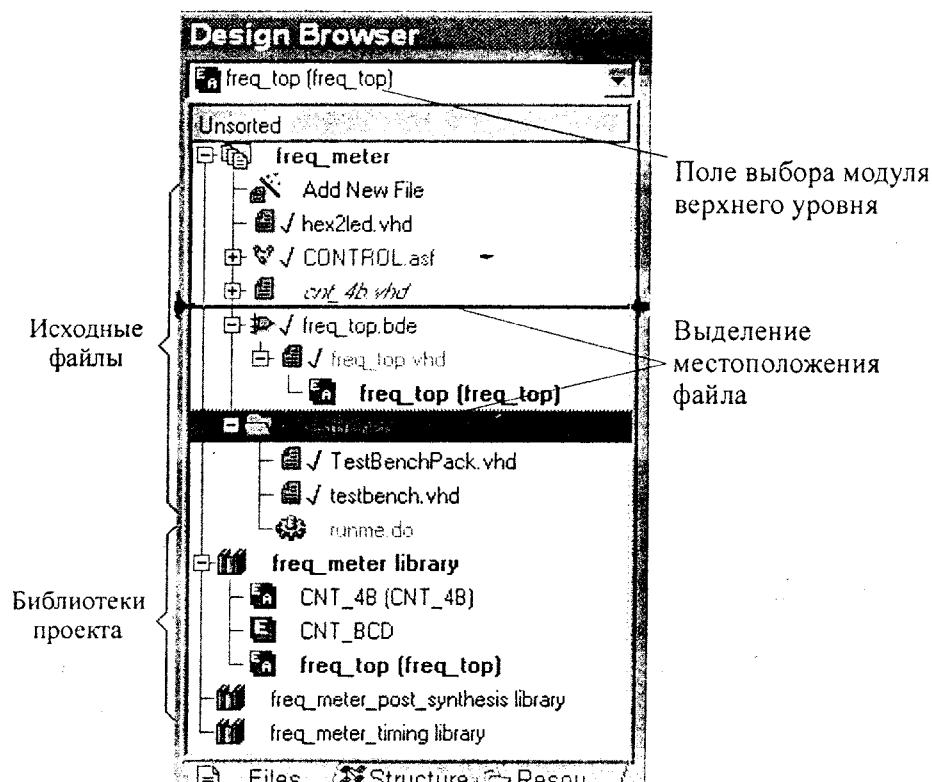
Отображает файлы ресурсов проекта и содержание рабочей, post-synthesis и timing библиотек.

Каждый проект обычно (но не обязательно) сохраняется в папке с тем же именем, что и имя проекта. В среде Active-HDL для обозначения папки текущего проекта может быть использована переменная (an environment variable) \$DSN. Разрешается сохранение исходных файлов проекта в любом месте. Тем не менее выделенной для них является папка \$DSN\Src. Ее содержание выводится в подокне Files. При необходимости файлы можно размещать в папках нижнего уровня. Если файл расположен вне \$DSN\Src, в окне Files его иконка имеет маркер . Например, .

На рисунке 3.3 содержимое проекта представлено в виде структурного дерева, верхняя часть которого содержит исходные файлы, а нижняя – библиотеки проекта. Для каждого файла имеется своя иконка, зависящая от его типа. Ветви дерева,

соответствующие исходным откомпилированным файлам, могут быть раскрыты для просмотра входящих в них модулей (за исключением пакетов). Аналогичным образом можно развернуть содержимое библиотек.

Рисунок 3.3. Подокно Files окна Design Browser



Исходные файлы могут быть отсортированы по имени или типу в восходящем или нисходящем порядке. Для этого используется кнопка Unsorted под полем модуля верхнего уровня.



Значок, следующий за иконкой исходного файла, описывает его состояние после компиляции:

- ✓ Успешная компиляция.
- ✗ Во время последней компиляции обнаружена ошибка.
- ! Во время последней компиляции выведено предупреждение.
- ? Файл не откомпилирован или был модифицирован после компиляции.

Если исходный файл откомпилирован и помещен в библиотеку, отличную от рабочей, ее идентификатор выводится в скобках после имени файла. Можно запретить компиляцию выбранного файла, который обозначается италиккой, например, *cnt\_4b.vhd* с рисунка 3.3.





























Нижняя часть окна Files представляет содержание библиотек проекта, обозначаемых иконкой . Каждый проект имеет рабочую библиотеку. К нему также могут быть добавлены одна или несколько дополнительных библиотек – post-synthesis и/или timing. В подокне Files выводятся только описания, которые могут быть назначены модулями верхнего уровня для моделирования. Таковыми являются:

- Интерфейс проекта. (Формально в VHDL он не является модулем проекта, а существует только в паре с архитектурой).
- Интерфейс без архитектуры. (Не может быть промоделирован).
- Архитектура.
- Декларация конфигурации.

-  Verilog модуль.
-  Модуль, полученный после компиляции EDIF файла.

### Типы исходных файлов

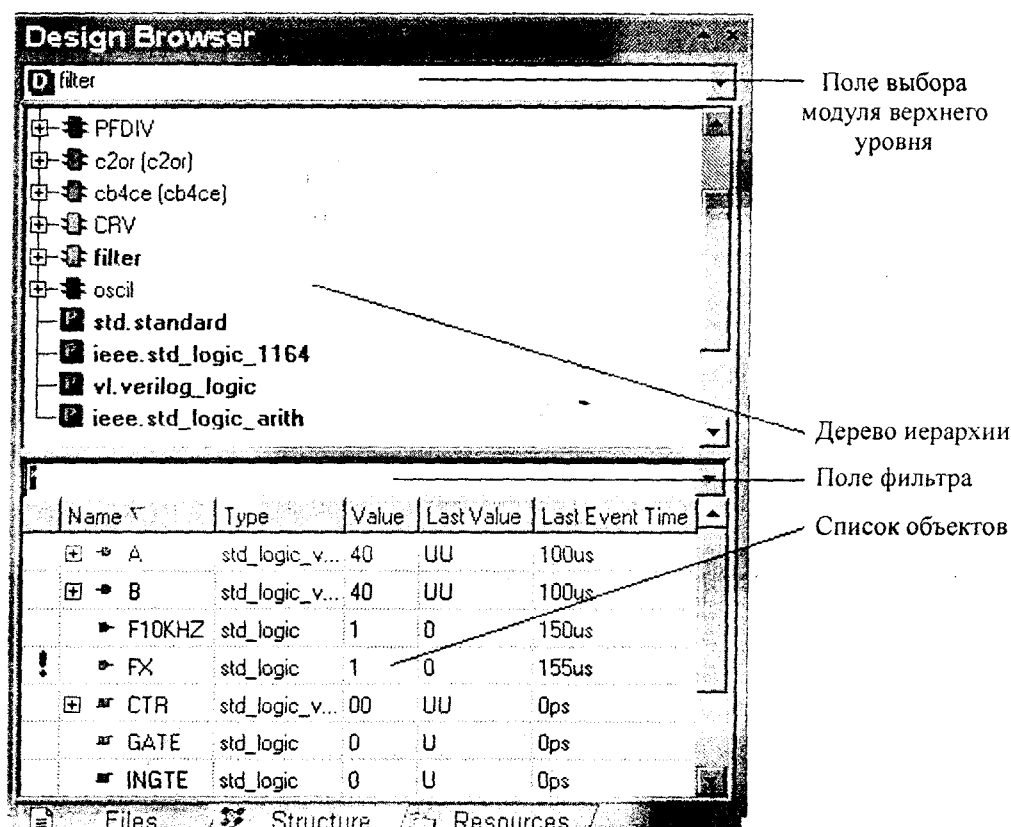
Список, представленный ниже, содержит поддерживаемые типы файлов, их расширения и иконки.

	VHDL- код (VHD, VHDL, VHQ, TVHD, VHO, VHM, VHI)	
	VHDL TestBench (VHD, VHDL, VHQ, VHO)	
	Файл конфигурации	(VHD)
	Файл редактора Block Diagram Editor	(BDE)
	Файл редактора State Diagram Editor	(ASF)
	Файл проекта Active-HDL	(ADF)
	List файл табличного представления результатов моделирования	(LST)
	Файл временных диаграмм waveform	(AWF)
	Проект Active-CAD	(PDF)
	Тестовые векторы Active-CAD	(ASC)
	Basic скрипт	(BAS)
	C++ файл	(CPP, C, H)
	Рисунок	(AFC)
	EDIF файл	(EDF, EDN, EDO)
	Схема EDIF	(EDI, EDS)
	Папка	—
	Внешний файл	—
	HTML document	(HTM, HTML)
	Командный файл	(DO)
	Perl скрипт	(PL, PM)
	SDF файл	(SDF, SDO)
	Поле символа	(BDS)
	Tcl скрипт	(TCL, TK)
	Text file	(TXT)
	Verilog модель (V, VEI, VEO, VCD, VO, VM;VMD;VLB;VLG)	
	Verilog TestBench	(V, VEI, VEO, VCD, VO)
	Схема редактора Viewlogic	(1)
	Файл XNF	(XNF)

### Подокно Structure

На рисунке 3.4 представлено подокно Structure, разделенное на две части. Верхняя содержит иерархичную структуру проекта. Для сортировки информации следует щелкнуть правой кнопкой мыши и задать необходимые опции из выпадающего меню Sort. Нижняя часть содержит объекты, определенные в модуле, который выделен в верхней части окна Structure. Выводится: имя объекта, тип, текущее и предыдущее значения, время последнего события. Класс объекта (сигнал, переменная, константа, порт, generic-константа, файл) обозначается иконкой рядом с его именем. Поле фильтра позволяет фильтровать выводимые объекты, используя регулярные выражения.

Рисунок 3.4. Подокно Structure окна Design Browser



Иерархическая структура проекта состоит из блоков и процессов, которые могут быть изображены следующими иконками:

- блок,
- параллельный процесс,
- пакет, используемый проектом.

Метка рядом с иконкой соответствует ее имени в исходном коде. Если параллельный оператор не имеет меток, Active-HDL автоматически генерирует их.

Следующие иконки используются для VHDL-объектов, выводимых в нижней части окна:

- порт в режиме in,
- порт в режиме out,
- порт в режиме inout,
- сигнал,
- переменная,
- константа,
- generic-константа,
- файл.

Список объектов содержит колонки:

1. **Event** (событие). Колонка не имеет названия и расположена скраю слева. Если событие имеет место в текущем цикле моделирования, сигнал отмечается красным восклицательным знаком.
2. **Name** (имя). Идентификатор объекта.
3. **Value** (значение). Текущее значение объекта. Для файлов в этой колонке выводится режим, в котором он открыт: `read_mode`, `write_mode`, `append_mode`.

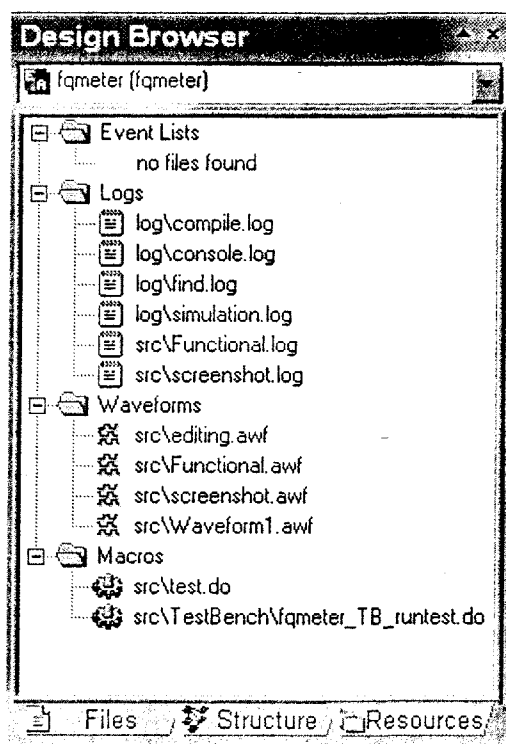
4. **Type** (тип). VHDL-тип объекта.
5. **Last Value** (последнее значение). Предыдущее значение объекта, которое он имел до последнего события. Эта колонка используется только для сигналов.
6. **Last Event Time**. Время последнего события. Используется также только для сигналов.

Можно выбирать нужные колонки для вывода. Изучать состояние объектов допускается только в момент приостановки процесса моделирования. Значения, отличные от выведенных в момент предыдущей приостановки, выделяются красным цветом.

### Подокно Resources

Представляет файлы ресурсов, отсортированные в соответствии с их типами, которые изображены на рисунке 3.5. Для каждой папки можно определить: её имя, множество расширений файлов, которые будут включены в нее. Разрешается добавлять или удалять папки ресурсов. Они не являются реальными, не влияют на папки, в которых содержатся файлы, и существуют только в подокне **Resource** окна **Design Browser**.

Рисунок 3.5. Подокно Resources окна Design Browser

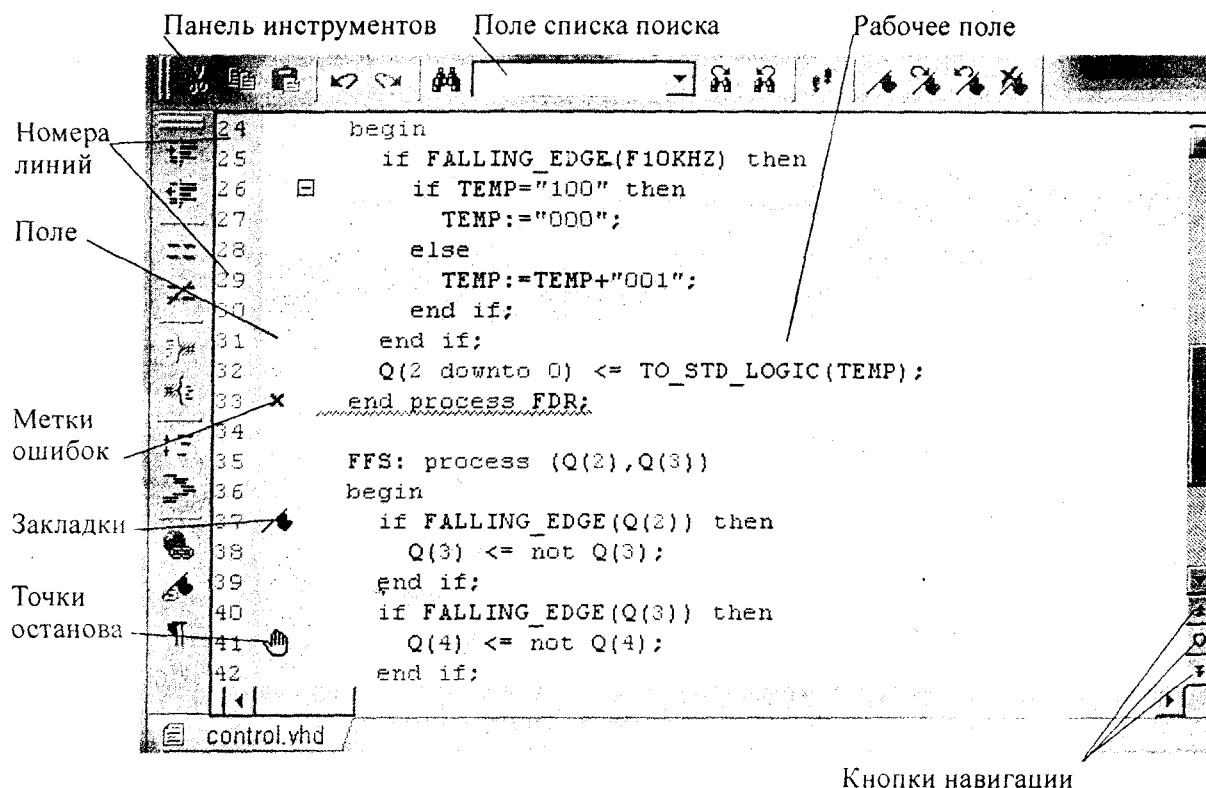


### 3.3. Редактор HDL Editor


Редактор HDL Editor – это инструмент для ввода моделей, представленных в виде VHDL, Verilog и C/C++ кодов, а также других текстовых файлов. Active-HDL предлагает несколько автоматических методов генерации VHDL-кода, таких как New Source File Wizard.

Рисунок 3.6 содержит окно редактора HDL Editor.

Рисунок 3.6. Окно редактора HDL Editor



Окно редактора HDL Editor содержит:

- 1) Панель инструментов (**Toolbar**), на которой расположены кнопки для наиболее часто используемых команд.
- 2) Рабочее поле (**Editing Workspace**), предназначенное для редактирования HDL документов. Для улучшения читаемости исходного кода отдельные синтаксические категории выделяются различным цветом.
- 3) Поле (**Margin**), на котором размещаются закладки, метки ошибок и номера линий.
- 4) Номера линий (**Line Numbers**), облегчающие редактирование больших документов. Относительно их выдаются сообщения об ошибках.
- 5) Закладки (**Bookmarks**), улучшающие навигацию больших документов. Можно определить несколько закладок и затем быстро переходить от одной к другой.
- 6) Точки останова (**Breakpoints**) в VHDL используются для приостановки процесса моделирования.
- 7) Метки ошибок (**Error Marks**), помогающие их обнаруживать. Для того чтобы найти метку ошибки из окна **Console**, следует дважды щелкнуть по полю сообщения об ошибке. После этого будет открыт исходный файл и выделена строка с оператором, требующим исправления.
- 8) Поле списка поиска (**Find List Box**), используемое для быстрого поиска описанной строки. Для этого необходимо ввести строку и нажать кнопки .
- 9) Кнопки навигации (**Browse Buttons**) предназначены для быстрой навигации редактируемого текста.



### 3.4. Создание нового проекта

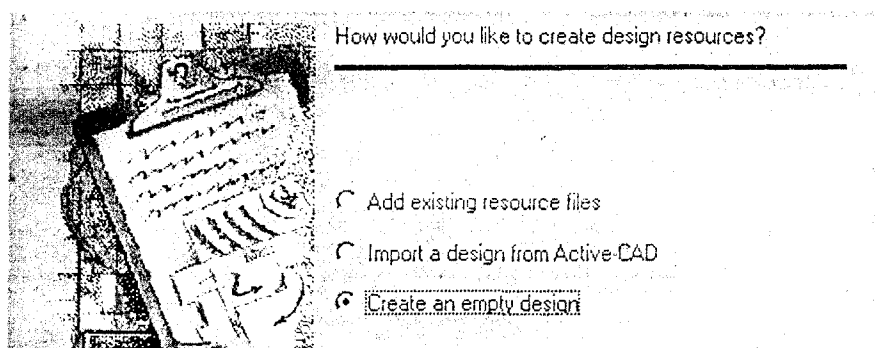
Для того чтобы создать проект непосредственно из окна Active-HDL, следует выбрать команду New Design из меню File. После этого будет запущен на выполнение мастер New Design Wizard.

В первом окне (рисунок 3.7) следует выбрать содержание проекта из вариантов:

- "Add existing resource files" – добавить существующие файлы.
- "Import a design from Active-CAD" – импортировать проект из Active-CAD.
- "Create an empty design" – создать пустой проект.

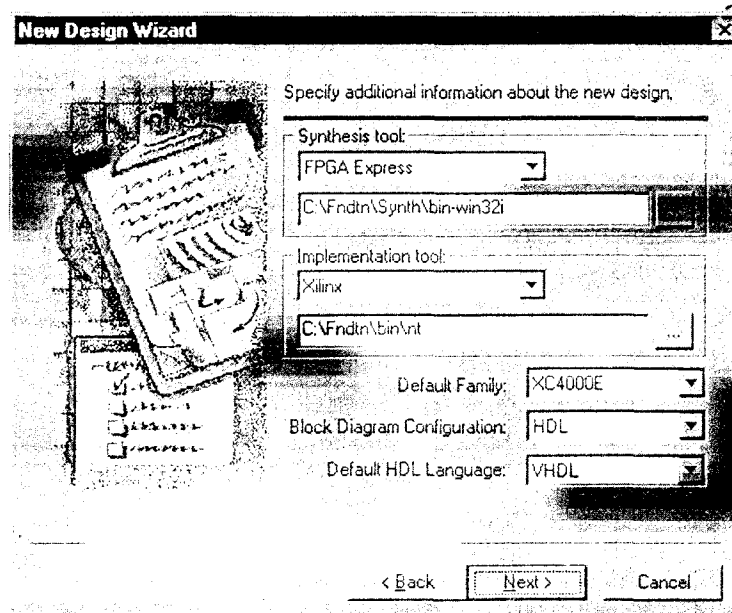
На примере модели полного сумматора проиллюстрируем создание нового проекта. В первом окне отмечается вариант "Create an empty design".

Рисунок 3.7. Фрагмент первого окна мастера New Design Wizard



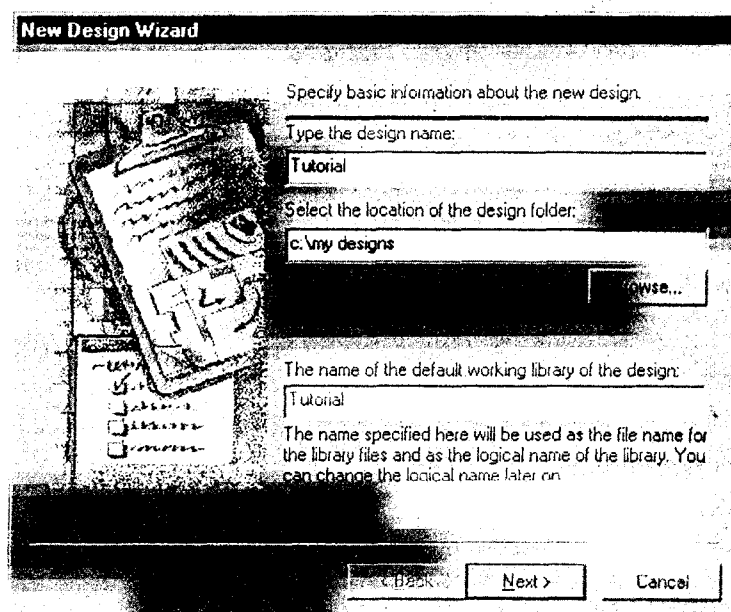
Второе окно мастера (рисунок 3.8) предназначено для выбора инструментов синтеза и имплементации. Необходимо также указать тип микросхемы, конфигурацию редактора Block Diagram и HDL-язык. Поскольку до сих пор не рассматривался синтез и имплементация, данные параметры имеют значения, устанавливаемые по умолчанию. В поле "Default HDL Language" выбирается параметр "VHDL".

Рисунок 3.8. Окно мастера для описания средств синтеза и имплементации




В следующем окне (рисунок 3.9) задаются имена проекта и рабочей библиотеки, описывается расположение проекта на диске.

Рисунок 3.9. Окно мастера для ввода имени проекта



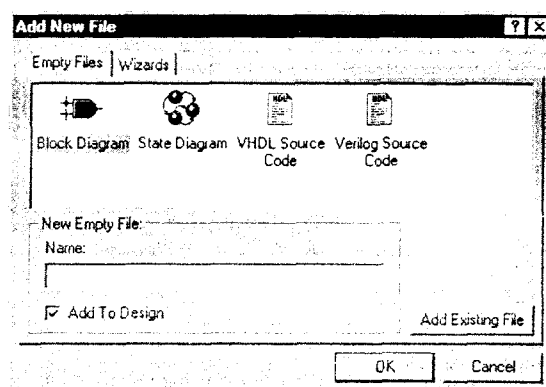
Последнее окно выводит свойства создаваемого проекта для подтверждения их принятия – кнопка Finish, изменения – кнопка Back и отмены создания нового проекта – кнопка Cancel.

### Создание нового документа с помощью New Source File Wizard

Мастер New Source File Wizard вызывается командой File/New/VHDL Source или двойным щелчком по иконке  Add New File в окне Design Browser. После этого появляется окно Add New File (рисунок 3.10), которое позволяет выбрать создание нового файла мастером (вкладка Wizards) или задание пустого файла (вкладка Empty) с указанием типа создаваемого документа.

Для ввода проекта сумматора следует выбрать тип HDL Source Code из подокна Wizards.

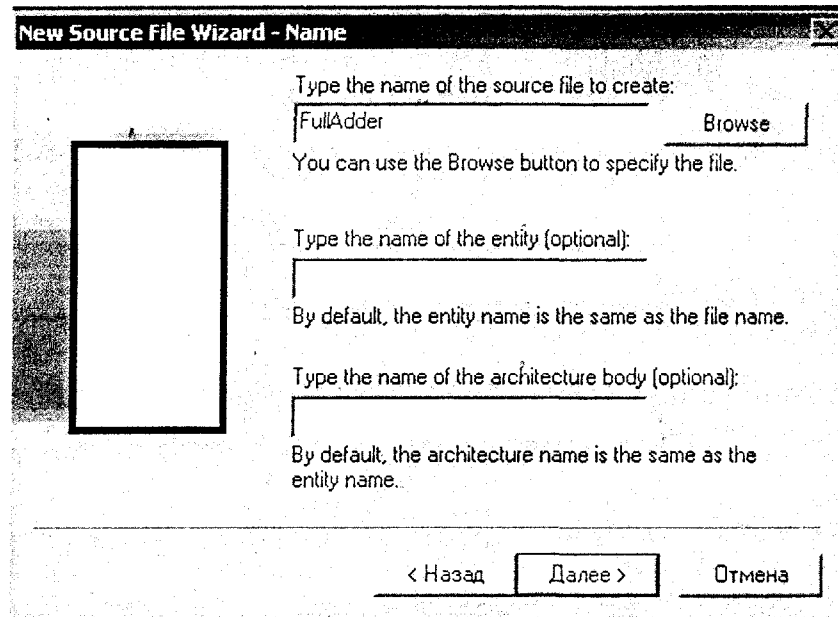
Рисунок 3.10. Окно Add New File



Мастер облегчает ввод VHDL-модели, который выполняется за несколько шагов.

- 1) Описывает, будет ли создаваемый код добавлен к проекту. Для этого должен быть установлен флажок Add the generated file to design.
- 2) Вводятся имена файла, интерфейса и архитектуры проекта. Имя файла является обязательным параметром, остальные – нет. Для сумматора это имя FullAdder (рисунок 3.11).

Рисунок 3.11. Окно Name мастера Source File Wizard

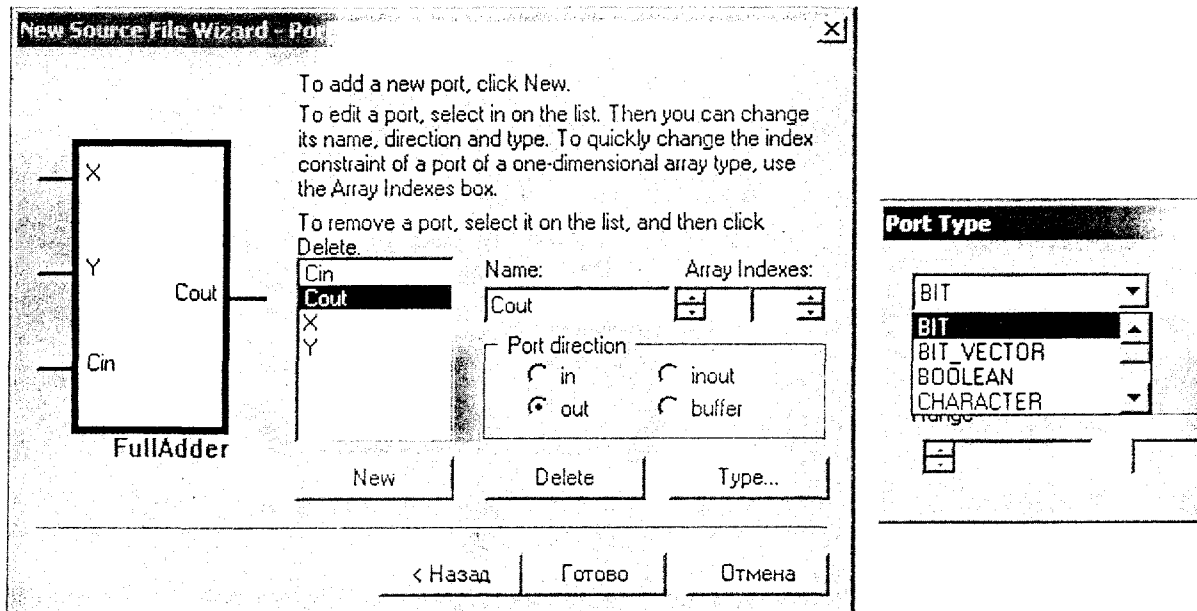


3) Вводится информация о портах (рисунок 3.12). Для создания нового порта необходимо нажать кнопку New. В поле Name ввести его имя. Для выбора направления используется группа параметров – Port direction. Чтобы выбрать тип создаваемого сигнала, следует щелкнуть по кнопке Type и открыть окно Port Type. Параметр Array Indexes используется для ввода диапазонов массивов.

Полный сумматор имеет следующие порты:

X, Y, Cin: in bit;  
Cout, Sum: out bit.

Рисунок 3.12. Окно Ports мастера Design Wizard Window



После того как будет нажата кнопка Готово (Finish), создается шаблон VHDL-кода модели сумматора. В архитектуре вместо надписи "--enter your statement here" необходимо добавить код, описывающий поведение сумматора:

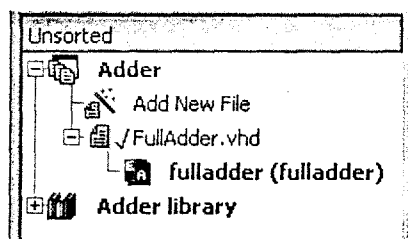
```
Sum <= X xor Y xor Cin after 10 ns;
Cout <= (X and Y) or (X and Cin) or (Y and Cin) after 10 ns;
```

Полученная VHDL-модель приведена на рисунке 3.13. Структура созданного проекта представлена на рисунке 3.14.


Рисунок 3.13. VHDL-код модели полного сумматора

```
18
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.all;
21
22 entity FullAdder is
23     port(
24         X : in BIT;
25         Y : in BIT;
26         Cin : in BIT;
27         Cout : out BIT;
28         Sum : out BIT
29     );
30 end FullAdder;
31
32 --)) End of automatically maintained section
33
34 architecture FullAdder of FullAdder is
35 begin
36
37     Sum<= X xor Y xor Cin after 10 ns;
38     Cout<= (X and Y) or (X and Cin) or (Y and Cin) after 10 ns;
39
40 end FullAdder;
```

Рисунок 3.14. Структура проекта, представленная в окне Design Browser



### Создание нового пустого документа в HDL Editor

Для создания нового документа следует щелкнуть по стрелке кнопки New , а затем из выпадающего меню выбрать тип документа:

- VHDL Source (VHDL-код),
- Verilog Source (Verilog-код),
- Text Document (Текстовый документ),
- Perl Script (Perl скрипт),
- Tcl Script (Tcl скрипт),
- Macro (Макрос).

Содержание кнопки New всегда имеет иконку типа последнего созданного документа. По этой причине она может выглядеть по-разному.

Таким образом, можно ввести и добавить к проекту модель 4-разрядного сумматора (см. рисунок 2.5). Рисунок 3.15 представляет содержимое окна Design Browser после выполнения этой операции. Аналогично создается и командный файл Macro, содержащий макрокоманды для моделирования сумматора adder4:

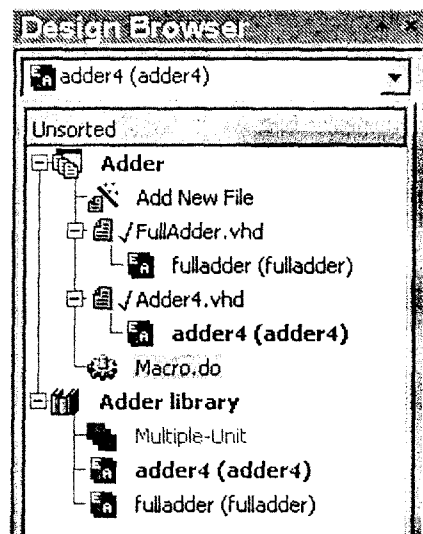
```
asim Adder4
list A B Co C Ci S
```

```

wave A B Co C Ci S
force A 1111
force B 0001
force Ci 1
run 50 ns

```

Рисунок 3.15. Структура проекта



### 3.5. Language Assistant

VHDL-файл, генерируемый мастером, представляет собой шаблон, в который добавляются определенные пользователем имена. Более опытные разработчики скорее предпочитают создавать пустые файлы. Тем же, кто только начинает разрабатывать VHDL-проекты, лучше пользоваться мастером, который имеет непосредственную связь с Language Assistant. Этот инструмент позволяет выбирать шаблоны кода и копировать их из окна Language Assistant в исходные файлы. Таким образом, увеличивается скорость разработки моделей и уменьшается количество ошибок. Кроме использования готовых шаблонов есть возможность создавать и свои собственные.

Предлагается несколько групп шаблонов:

- 1) **Language templates** языковые шаблоны с основными конструкциями языка.
- 2) **Synthesis templates** синтезируемые шаблоны – модели основных блоков, таких как мультиплексоры, триггеры, счетчики.
- 3) **Simulation templates** моделируемые шаблоны с примерами полезных конструкций.
- 4) **Code Auto Complete** автоматическое дополнение кода с шаблонами, требующими Auto-Complete и Interactive templates свойств редактора HDL Editor. Эти свойства позволяют вводить ключевые слова и шаблоны по первым набранным буквам. Создание новых шаблонов для этой группы расширяет возможности автоматического ввода.
- 5) **Macro Commands** макрокоманды – шаблоны с примерами основных команд, полезных для компиляции, моделирования, выполнения скриптов, управления проектом и наблюдения сигналов.


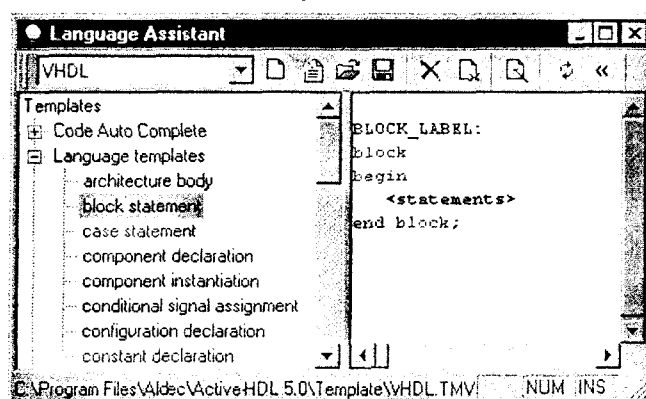
На рисунке 3.16 представлено окно Language Assistant. Его левая часть задает иерархическое дерево шаблонов. Правая позволяет выполнить предварительный просмотр выбранного шаблона. Добавить шаблон к исходному коду можно, используя технологию drag-and-drop или выполнив команду Use из меню правой кнопки мыши. Также можно использовать кнопку .

Рисунок 3.16. Окно Language Assistant

Поле выбора языка



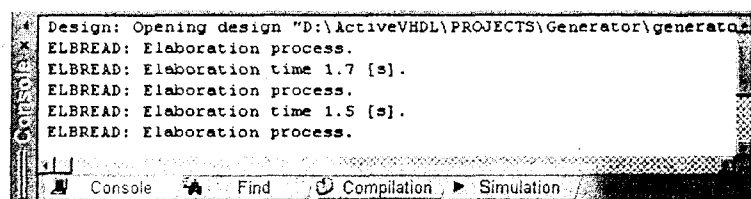
Список шаблонов

Панель предварительного просмотра шаблона

### 3.6. Окно Console

Окно Console (рисунок 3.17) позволяет вводить макрокоманды и скрипты. Все инструменты Active-HDL используют Console для вывода сообщений.

Рисунок 3.17. Окно Console




Окно Console содержит несколько вкладок для управления сообщениями от различных Active-HDL инструментов. Вкладка Console выводит сообщения, генерируемые всеми инструментами. Она используется также для ввода макрокоманд. Подокно Compilation выводит только сообщения, генерируемые компилятором. Подокно Find используется для просмотра результатов поиска, выполненного командой Find in Files из меню Search. Подокно Simulation выводит сообщения, генерируемые во время моделирования.


Сообщения, генерируемые средой Active-HDL и выводимые в окне Console, сохраняются в log файлах и доступны для просмотра. Сообщения из каждого подокна сохраняются в отдельном файле. Log файлы текущего проекта перечислены в подокне Resources окна Design Browser.


### 3.7. Компиляция

Компиляция – это процесс анализа исходных файлов, которые затем размещаются в рабочей библиотеке в формате, понятном для системы моделирования. По умолчанию все файлы проекта компилируются в рабочую библиотеку. Однако её можно задавать индивидуально для каждого исходного файла проекта. После того как модуль был откомпилирован, можно выполнять его моделирование.

Active-HDL предоставляет несколько способов для выполнения компиляции файлов проекта:

1. Исходные файлы могут быть откомпилированы отдельно. Для этого используется команда **Design /Compile** или кнопка на панели инструментов .
2. Все исходные файлы компилируются за один проход. В этом случае файлы

обрабатываются в том порядке, в котором они представлены в подокне Files окна Design Browser (сверху вниз). Команда **Design /Compile All**. Кнопка .

3. Все файлы компилируются за один проход с предварительным изменением порядка обработки. Их сортировка выполняется автоматически, гарантируя таким образом правильный порядок компиляции. Команда **Design /Compile All with File Reorder**. Кнопка .
4. Можно компилировать за один проход все исходные файлы, собранные в одну папку. Она должна быть определена в подокне Files окна Design Browser. Команда **Compile All in Folder** доступна из меню, вызываемого щелчком правой кнопки мыши по папке.
5. Можно также выполнить макрокоманду **acom**.

Design Browser позволяет исключать выбранные файлы и папки из процесса компиляции. Команды **Compile All**, **Compile All with File Reorder** и **Compile All in Folder** игнорируют такие файлы. Они также игнорируются при выполнении компиляции макрокомандой **acom**, если в ней не были описаны исходные файлы.

Все сообщения, генерируемые во время компиляции, выводятся в окне **Console**.

### 3.8. Окно List

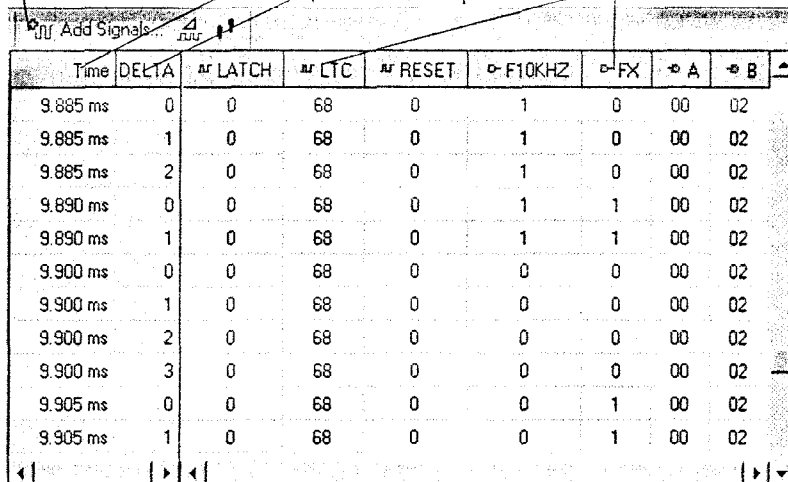
Окно **List** (рисунок 3.18) отображает значения выбранных сигналов, полученные во время моделирования, в текстовой табличной форме. Содержание окна может быть сохранено в текстовом формате. Каждому сигналу соответствует колонка значений для последовательных моментов моделирования. Значения сигналов могут быть представлены двумя способами:

- для всех циклов моделирования, включая дельта-циклы, выполняемые для каждого шага моделирования;
- только последние циклы для каждого шага моделирования.


Рисунок 3.18. Окно List




Панель инструментов      Время моделирования      Сигналы

Циклы моделирования



Time	DELTA	LATCH	LTC	RESET	F10KHZ	FX	A	B
9.885 ms	0	0	68	0	1	0	00	02
9.885 ms	1	0	68	0	1	0	00	02
9.885 ms	2	0	68	0	1	0	00	02
9.890 ms	0	0	68	0	1	1	00	02
9.890 ms	1	0	68	0	1	1	00	02
9.900 ms	0	0	68	0	0	0	00	02
9.900 ms	1	0	68	0	0	0	00	02
9.900 ms	2	0	68	0	0	0	00	02
9.900 ms	3	0	68	0	0	0	00	02
9.905 ms	0	0	68	0	0	1	00	02
9.905 ms	1	0	68	0	0	1	00	02

Открыть окно **List** можно командой **New> List** из меню **File** или использовать для этого кнопку  панели инструментов. В одно и то же время можно работать с несколькими окнами **List**.

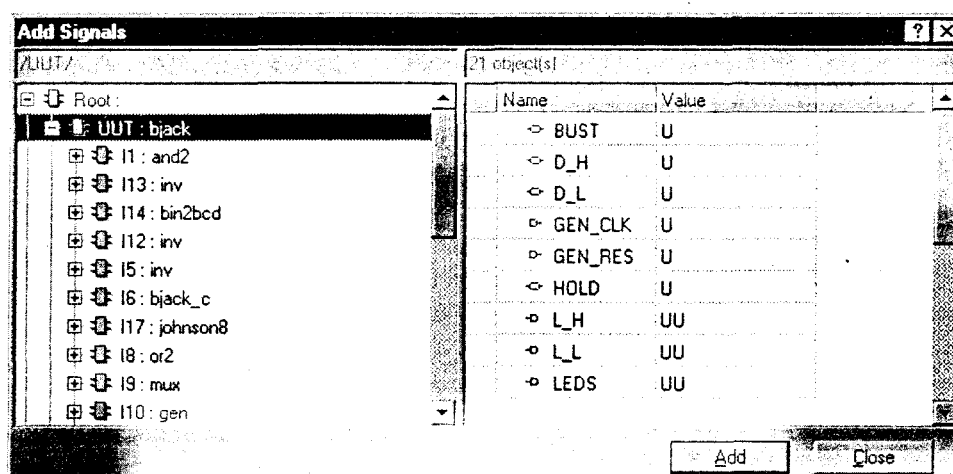
Панель инструментов окна List содержит кнопки:  – добавить сигналы,  – разрешение вывода дельта-циклов,  – прокрутка содержимого окна до заданной временной точки.

Добавить сигналы в окно List можно, используя технологию "Drag-and-drop". Для этого следует выбрать сигнал(ы) из списка, расположенного в нижней части вкладки Structure окна Design Browser, а затем переместить его на рабочее пространство окна List. Если нужно выделить несколько сигналов, следует использовать клавишу Ctrl.

Другим способом добавить сигналы в окно List можно с помощью команды Copy и Paste. Выбранный сигнал или ветвь иерархии копируется с помощью команды Copy. Чтобы добавить их к содержимому окна List, используется команда Paste. Обе команды доступны из меню правой кнопки мыши, в окне List которой следует щелкать по свободному пространству, а не по колонке сигнала.

Также можно добавлять сигналы, используя диалоговое окно Add Signals. Для этого следует щелкнуть правой кнопкой мыши в окне List и выбрать команду Add Signals из контекстного меню, после чего будет открыто окно Add Signals (рисунок 3.19). В левой его части нужно указать область проекта, сигналы из которой будут помещены в окно List. В правой – выбирается сам сигнал. Затем нажимается кнопка Add. Для того чтобы выбрать более одного сигнала, используется кнопка Ctrl.

Рисунок 3.19. Диалоговое окно Add Signals dialog



Чтобы удалить сигнал из окна List, его нужно выделить и нажать клавишу Delete.

Для выводимых значений сигналов можно указывать различные системы счисления. При этом следует выделить один или несколько сигналов и выбрать команду Properties из меню правой кнопки мыши. Будет открыто окно Signal Properties, в котором задается требуемый параметр системы счисления: Binary (двоичная), Octal (восьмеричная), Decimal (десятичная) или Hexadecimal (шестнадцатеричная).

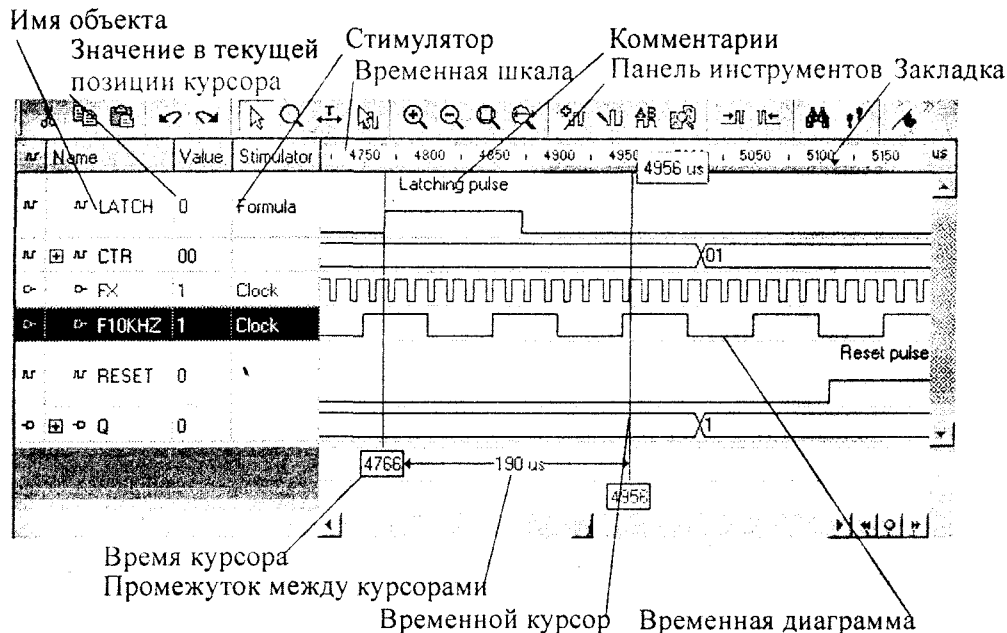
### 3.9. Редактор временных диаграмм

Waveform Editor (редактор временных диаграмм) – инструмент, предназначенный для представления результатов моделирования в форме временных диаграмм. Во время моделирования временные диаграммы предварительно выбранных сигналов и переменных выводятся в окно Waveform Editor. Новые объекты можно добавлять к окну в любой момент моделирования. Однако их временные диаграммы будут созданы, начиная с момента, в который они были добавлены в окно.



Waveform Editor является также средством редактирования и позволяет изменять временные диаграммы сигналов. Они могут быть сохранены в файле и использоваться как тестовые последовательности во время последующего моделирования. Существует возможность их экспортирования в различные файловые форматы: VHDL Process (\*.VHS), VHDL Waves Vectors (\*.VEC), List file (\*.LST). В один момент времени можно работать с несколькими окнами **Waveform Editor**.

Рисунок 3.20. Окно **Waveform Editor**



























Левая часть окна, изображенного на рисунке 3.20, содержит колонки, описывающие сигнал. Они представляют его основные характеристики, временные диаграммы которых выводятся в окне. Могут быть выведены следующие типы колонок:


1. Режим. Эта колонка содержит небольшие иконки, обозначающие, является ли сигнал обыкновенным (M) или портом. Для последних описывается их режим: in, out, inout, buffer, linkage.
2. Name (имя) содержит идентификатор сигнала.
3. Hierarchy (иерархия) – представляет иерархическую позицию сигнала (путь к нему) в проекте.
4. Type (тип) описывает VHDL-тип сигнала.
5. Value (значение) содержит значение сигнала для текущей позиции временного курсора.
6. Stimulator (стимулятор). В колонке выводится тип стимулятора, назначенного сигналу.


Временная шкала представляет собой масштабную линейку с временными метками и закладками, определенными пользователем. Панель инструментов содержит кнопки для часто используемых команд. Временной курсор (Timing Cursor) позволяет получать значения сигналов в любом месте временной шкалы.

### Кнопки панели инструментов окна **Waveform Editor**

-  Вырезать выделенную временную диаграмму и поместить ее в Clipboard.
-  Скопировать выделенную временную диаграмму в Clipboard.
-  Вклеить временную диаграмму из Clipboard.
-  Отменить последнюю операцию.

-  Восстановить последнюю операцию.
-  Перейти в режим выделения.
-  Перейти в режим масштабирования.
-  Перейти в режим измерения.
-  Перейти в режим редактирования.
-  Увеличить масштаб.
-  Уменьшить масштаб.
-  Настроить масштаб по всей временной диаграмме.
-  Добавить сигнал.
-  Добавить стимулятор к сигналу.
-  Ввести комментарий.
-  Сравнить текущие временные диаграммы с сохраненными в файле.
-  Переместить курсор к следующему событию выбранного сигнала или всех сигналов, если ни один из них не выделен.
-  Переместить курсор к предыдущему событию выбранного сигнала или всех сигналов, если ни один из них не выделен.
-  Поиск описанного значения или текстового комментария для выделенного сигнала.
-  Переместить курсор в заданную временную точку.
-  Установить / удалить закладку.
-  Переместить курсор к ближайшей закладке вперед.
-  Переместить курсор к ближайшей закладке назад.
-  Удалить все закладки.

Для того чтобы открыть новое окно Waveform Editor, можно использовать команду New>Waveform из меню File или кнопку  на основной панели инструментов.

Добавить сигналы к окну Waveform Editor можно таким же образом, как и для окна List. Кроме того, имя сигнала и путь к нему можно ввести вручную в окне Waveform Editor. Для этого нужно перейти в режим редактирования . Дважды щелкнуть по линии с меткой Click here и ввести имя объекта. Щелкнуть по колонке Hierarchy и ввести путь к сигналу, если это необходимо.

Для описания объектов, которые не объявляются на верхнем уровне, используется путь, показывающий их точное расположение в иерархии проекта. При этом применяется следующее синтаксическое правило:

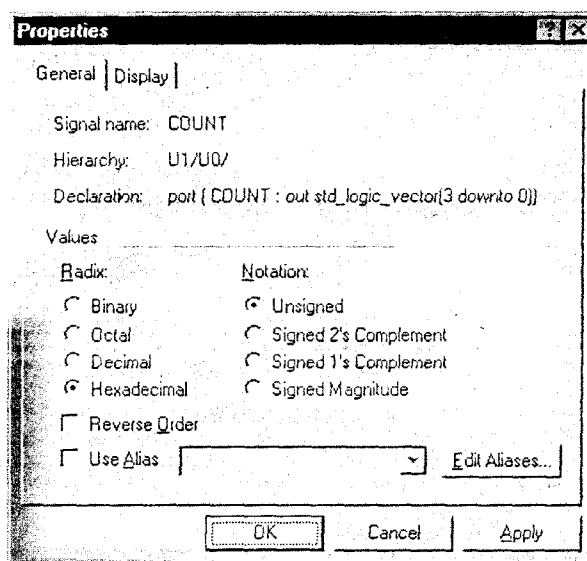
```
object_path ::= { instance_name / } object_name
```

Формальные параметры и локальные объекты в VHDL-подпрограммах (функциях и процедурах) выводятся с иерархическим путем следующего вида:

```
local_object_path ::= [ process_name / ] ( subprogram_name / )
object_name
```

Для редактирования свойств сигналов и их временных диаграмм существует команда Properties из контекстного меню правой кнопки мыши. Эта команда открывает окно Signal Properties (рисунок 3.21), которое содержит две вкладки: General и Display.

Рисунок 3.21. Окно Signal Properties–General



Вкладка General (см. рисунок 3.21) включает пункты:

- Signal name(имя сигнала).
- Hierarchy (иерархия). Выводится путь к выбранному сигналу.
- Declaration (декларация). Представлено описание сигнала из исходного кода.
- Группа Values описывает систему счисления, в которой представляются сигналы: Binary (двоичная), Octal (восьмеричная), Decimal (десятичная), Hexadecimal (шестнадцатеричная).
- Reverse Order(обратный порядок). В одномерном массиве меняется местоположение старшего и младшего битов на обратное.
- Use Alias (использование псевдонимов). Разрешение использовать выбранные из списка псевдонимы.
- Edit Aliases (редактирование псевдонимов). Открывается окно Alias Editor для создания и редактирования псевдонимов.
- Группа Notations (представление) содержит пункты: Unsigned (беззнаковое), Signed 2's Complement (знаковое двоичное дополнение), Signed 1's Complement (знаковый обратный код), Signed Magnitude (знаковый модуль). Если выбран пункт Unsigned, значение сигналов выводится в беззнаковом формате. В противном случае старший бит вектора рассматривается как знак.

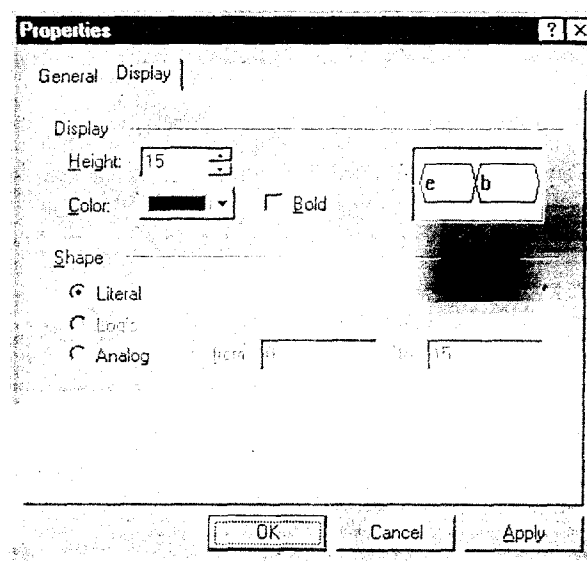
В таблице 3.1 даются значения двоичных чисел, представленных в различных кодах.

Таблица 3.1. Представления чисел в знаковых формах

Decimal	Singed-2' s complement	Singed-1' s complement	Singed- magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	-	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	-	-

Вкладка Display представлена на рисунке 3.22. Она содержит пункты, определяющие графическое изображение временных диаграмм: Height (высота), Color (цвет), Bold (полужирное), Preview (предварительный просмотр) и группу Shape (форма). Height позволяет задавать высоту временной диаграммы в пикселах. Повышение данного параметра увеличивает расстояние между сигналами, а не размер диаграммы. Параметр Preview разрешает вывод примера временной диаграммы с использованием текущих параметров. Группа Shape описывает форму, в которой будут представлены результаты моделирования. Это может быть Literal, когда значения временной диаграммы представлены в виде чисел в заданной системе счисления, или Logic, что соответствует логической форме (1, 0, X). Analog позволяет выводить временную диаграмму в аналоговой форме. После выбора соответствующего параметра аналоговые данные вычисляются в некотором диапазоне from .. to. Эта опция полезна для типов integer, physical и floating point.

Рисунок 3.22. Окно Signal Properties–Display

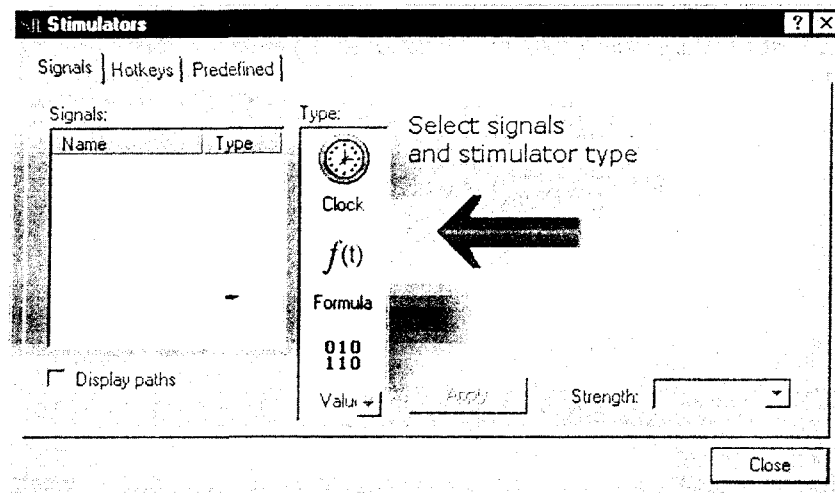


### 3.10. Стимуляторы

Стимуляторы – это определенные пользователем виртуальные источники входных значений, подключенные к сигналам. Значение, присвоенное стимулятором, может быть константой или изменяться по правилам, заданным типом стимулятора либо параметрами. Это самый быстрый и простой метод задания сигналам требуемых значений. Может применяться к любому сигналу или порту в иерархии проекта.

Active-HDL предлагает графический интерфейс для определения и применения стимуляторов. Для этого используется окно **Stimulators** (рисунок 3.23), вызываемое командой **Stimulators** из меню **Waveform** или краткого меню правой кнопки мыши.

Рисунок 3.23. Окно Stimulators, вкладка Signals



На вкладке **Signals** в поле **Signals** выводится список сигналов с соответствующими им стимуляторами. Колонка **Name** содержит имена сигналов, колонка **Type** – типы связанных с ними стимуляторов. Флаговая кнопка слева от имени сигнала позволяет включить или отключить стимулятор. Отключенные стимуляторы игнорируются во время моделирования. Чтобы добавить сигнал, необходимо щелкнуть по его имени в открытом окне **Waveform Editor**. Для того чтобы удалить стимулятор, его следует выделить, затем нажать **Delete**.

Поле **Strength** определяет, как стимулятор будет влиять на значение выбранной линии. Возможные режимы.

1 – **Deposit** – значения, присвоенные стимулятором, перекрывают текущее состояние сигнала, генерируемое в тестируемом устройстве. Эффект длится пока есть последовательность управляемых транзакций или программа моделирования не будет остановлена.

2 – **Drive** – значение(я), генерируемое стимулятором, воздействует на текущий сигнал так, как если бы еще один драйвер был подсоединен к этой линии. Эффект длится пока есть последовательность управляемых транзакций или программа моделирования не будет остановлена. Может использоваться только для разрешенных сигналов.

3 – **Override** – значение стимулятора заменяет текущее значение сигнала. Эффект длится пока программа моделирования не будет остановлена.

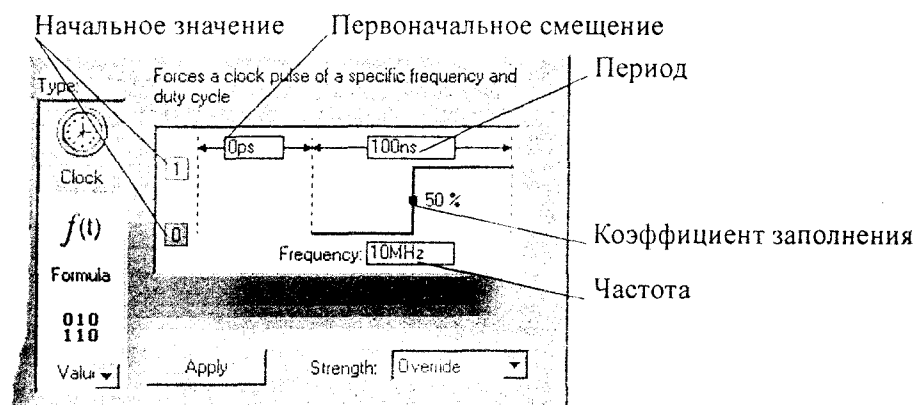
Флаг **Display Path** разрешает вывод пути иерархии сигнала в поле **Signals**. Кнопка **Apply** используется для назначения стимуляторов.

На вкладке **Signals** поле **Type** предназначено для выбора стимулятора выделенному сигналу. Существуют следующие их типы: **Clock Stimulators** (стимуляторы синхроимпульсов), **Counter Stimulators** (счетчики), **Custom Stimulators** (пользовательские стимуляторы), **Formula Stimulators** (формульные стимуляторы), **Value Stimulators**

(стимуляторы значений), Hotkey Stimulators (горячие клавиши), Predefined Stimulators (предопределенные стимуляторы).

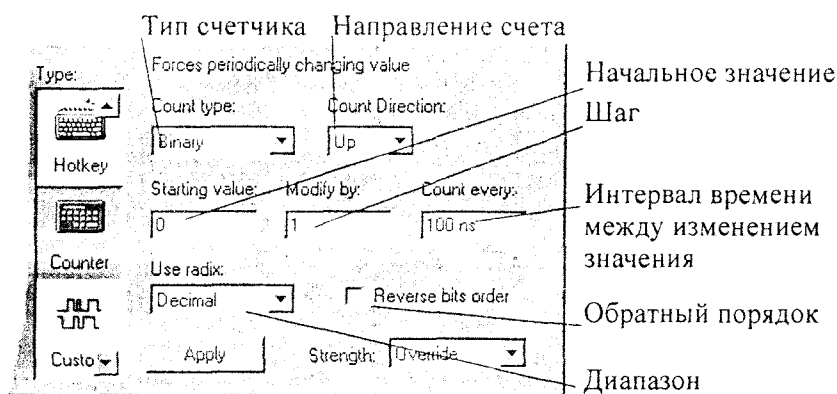
**Clock Stimulators** (стимуляторы синхроимпульсов) (рисунок 3.24) генерируют синхросигналы, имеющие следующие параметры: частоту/период, первоначальное смещение, коэффициент заполнения и начальное значение. Обычно такие стимуляторы используются для управления синхровходами.

Рисунок 3.24. Стимулятор синхроимпульсов



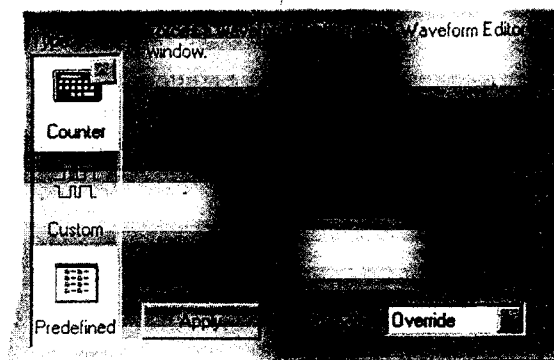
**Counter** (счетчик) может быть применен к сигналам типа одномерный массив или integer. Стимулятор создает последовательность значений, которые представляют собой очередь состояний счетчика. На рисунке 3.25 показаны параметры, которые можно задавать – это шаг и направление счета, интервал времени между изменением значения, начальное состояние и тип счетчика. Возможные типы счетчиков – binary (двоичный), Gray (Грея), Johnson (Джонсона), Circular One (циклическая единица) и Circular zero (циклический ноль). Параметр Increment by (шаг) применим только при использовании двоичного счетчика и счетчика Грея.

Рисунок 3.25. Стимулятор типа счетчик



**Custom Stimulators** (пользовательские стимуляторы) назначают сигналу его собственную временную диаграмму, которая уже существует в окне Waveform Editor. Ее можно создать вручную с помощью средств редактора Waveform Editor. Однако чаще используется временная диаграмма, полученная на предыдущем шаге моделирования или загруженная из файла. Например, для генерации формы сигнала сначала использовалась горячая клавиша. Затем для повторного применения полученной временной диаграммы тип стимулятора этого сигнала изменяется на Custom. Как видно из рисунка 3.26, стимулятор Custom не имеет параметров в подокне Signals.

Рисунок 3.26. Пользовательский стимулятор



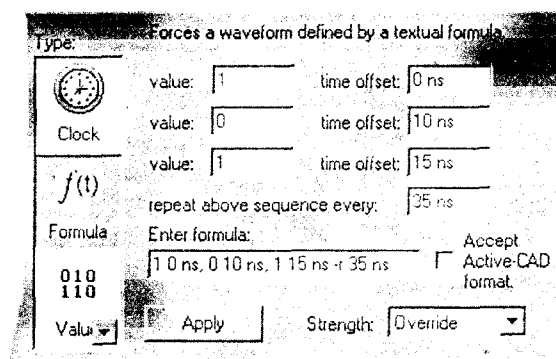
**Formula Stimulators** (формульные стимуляторы) (рисунок 3.27) генерируют временные диаграммы, заданные с помощью простой текстовой формулы. Она содержит последовательность пар параметров, состоящих из значения сигнала и момента времени, в которое он его получает. Также формула может содержать параметр повтора, означающий, что данная последовательность должна повторяться с заданным периодом.

Синтаксис формулы:

`<value> <time> [ , <value> <time> ... ] [ -r <period> ]`

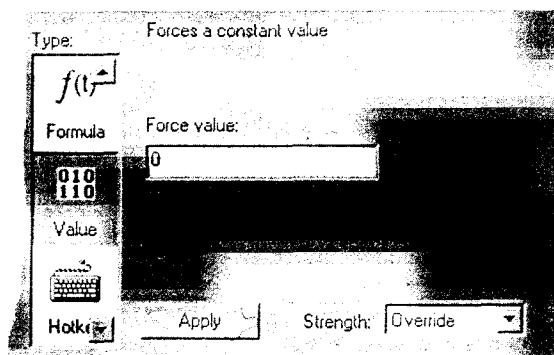
Параметры для ввода: "Value" – значение сигнала, "Time offset" – время присвоения значения, "Repeat above sequence every" – ввод периода для повторяющихся последовательностей, "Enter Formula" – позволяет ввести значение непосредственно в виде формулы. По умолчанию, если не задано явно, значение времени измеряется в ps.

Рисунок 3.27. Формульный стимулятор



**Value Stimulators** (стимуляторы значений) назначают сигналу константное состояние (рисунок 3.28).

Рисунок 3.28. Стимулятор значения



**Hotkey Stimulators** (горячие клавиши) подобны стимуляторам значений, но предлагают более удобный механизм для изменения значения. Для этого надо просто нажать описанную (горячую) клавишу, после чего произойдет переключение значения, например, из '0' в '1'. Для определения горячей клавиши или комбинации клавиш используется поле "Press new hotkey" (рисунок 3.29).

Можно определить список значений, которые будут циклически изменяться при нажатии горячей клавиши. Для этого применяется подокно Hotkeys окна Stimulators (рисунок 3.30). Колонка Hotkey содержит список горячих клавиш стимуляторов, колонка Sequence – список значений для каждого стимулятора горячей клавиши. Стимулятор присваивает их сигналу, начиная с самого левого значения и заканчивая самым правым. Для изменения значений или порядка их следования нужно щелкнуть по колонке Sequence, а затем выполнить необходимое редактирование. Значения должны быть разделены запятыми и принадлежать типу сигнала, которому они назначаются.

Рисунок 3.29. Стимуляторы, использующие горячие клавиши

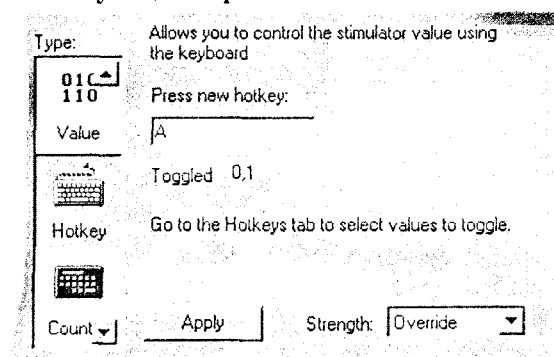
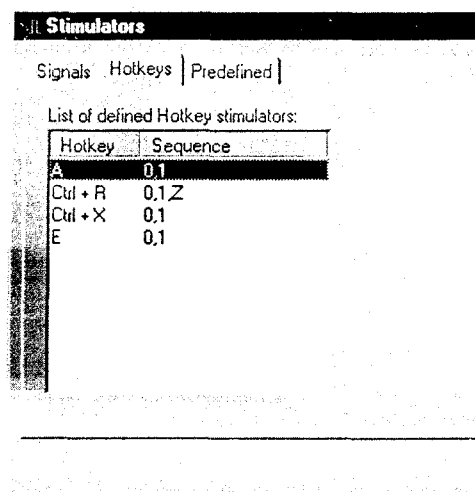


Рисунок 3.30. Подокно Hotkey окна Stimulators



**Predefined Stimulators** (предопределенные стимуляторы) – это стимуляторы синхросигналов или формульные, имеющие собственное уникальное имя (рисунок 3.31). Поскольку они имеют собственный идентификатор, их легко назначить нескольким сигналам, не повторяя каждый раз ввод параметров. Поле Predefined вкладки Signals окна Stimulators содержит список доступных предопределенных стимуляторов.

Для создания новых предопределенных стимуляторов используется вкладка Predefined (рисунок 3.32). Поле "List of predefined stimulators" содержит список имеющихся предопределенных стимуляторов. В колонке Name описывается имя, в колонке Type – тип. Стимулятор может иметь тип Clock (синхросигнал) или Formula (формула). Кнопка Add используется для создания стимуляторов, а Remove – для их удаления.

*Примечание.* Описание новых предопределенных стимуляторов должно быть сохранено в системном реестре. Они не могут быть сохранены в файле временных



диаграмм (\*.awf). Поэтому их нельзя эффективно использовать при моделировании на других компьютерах.

Рисунок 3.31. Предопределенные стимуляторы

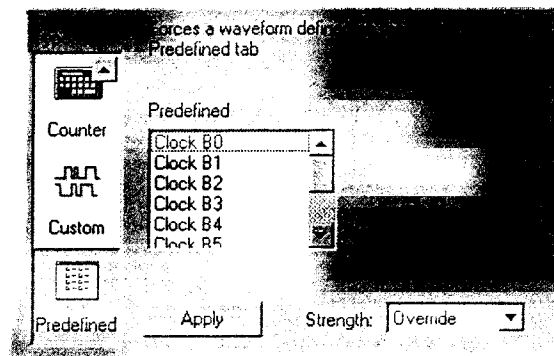
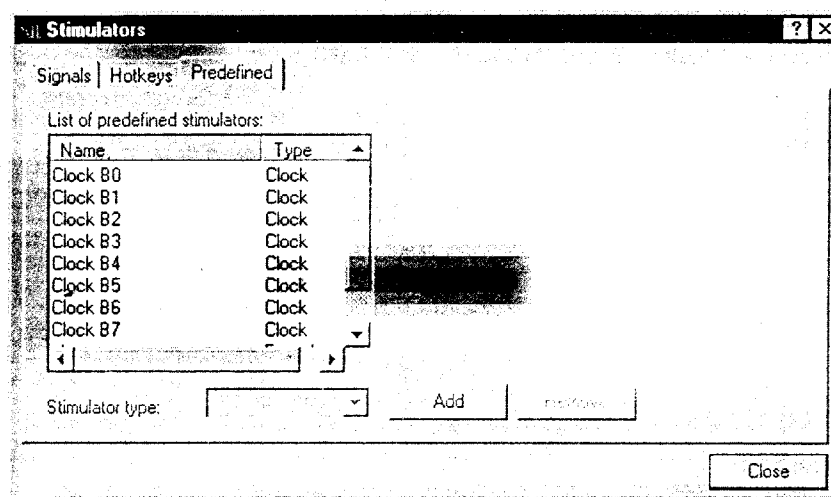


Рисунок 3.32. Вкладка Predefined окна Stimulators

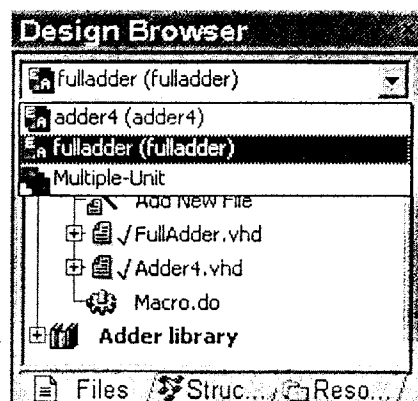


### 3.11. Моделирование

После того как VHDL-модель устройства успешно откомпилирована, для ее верификации следует использовать моделирование. Перед этой процедурой, если в проекте существует несколько HDL-файлов, необходимо установить модуль верхнего уровня описания.

Для того чтобы промоделировать модель полного сумматора с рисунка 3.13 и 3.14, следует в окне Design Browser на вкладке Files или Structure в поле выбора модуля верхнего уровня выбрать FullAdder, как это показано на рисунке 3.33. Затем выполняется инициализация моделирования – команда Initialize Simulation из меню Simulation.

Рисунок 3.33. Выбор модуля верхнего уровня



## Назначение стимуляторов


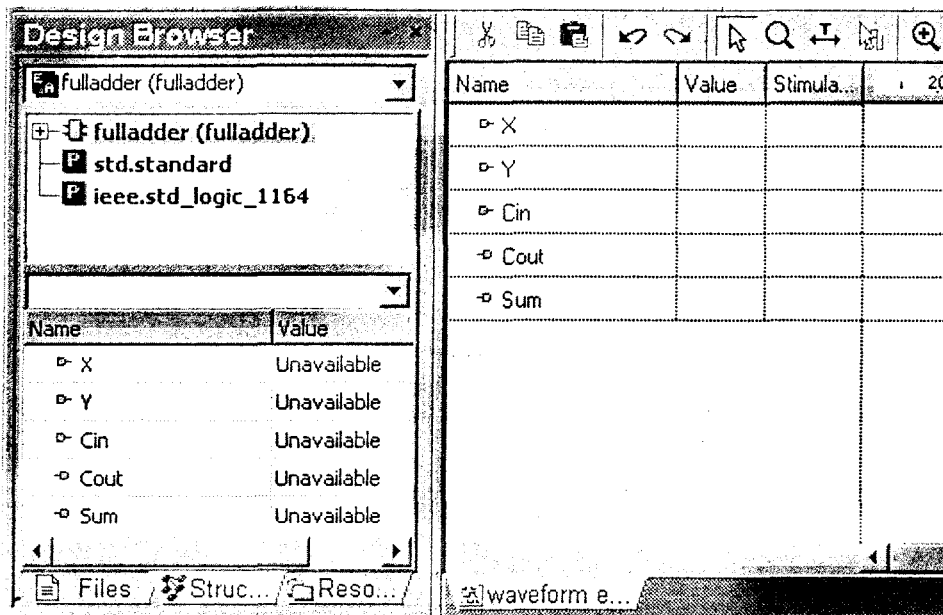
Командой File/New/Waveform или щелчком по кнопке  открывается новое окно Waveform Editor. В него с вкладки Structure окна Design Browser переносятся сигналы X, Y, Cin, Cout, Sum (рисунок 3.34).

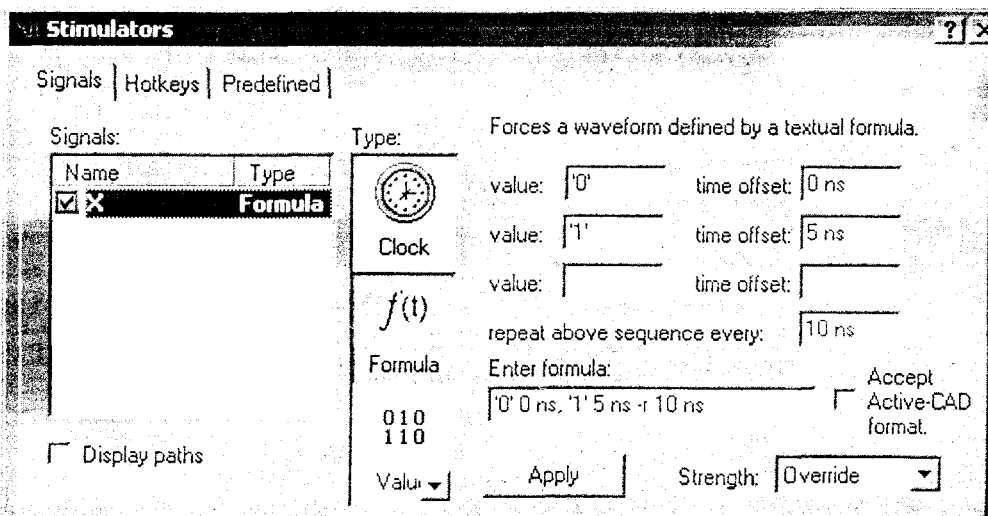
Рисунок 3.34. Окно Waveform Editor с добавленными к нему сигналами



Добавить сигналы можно также с помощью команд Copy и Paste или из окна Add Signals, которое вызывается командой Add Signals из контекстного меню правой кнопки мыши.

Чтобы открыть окно Stimulators, необходимо выделить порт X, щелкнуть правой кнопкой мыши и из контекстного меню выбрать команду Stimulators. В поле Type вкладки Signals выбрать формульный стимулятор, ввести параметры, как это показано на рисунке 3.35, нажать кнопку Apply.

Рисунок 3.35. Ввод параметров формульного стимулятора



Не закрывая окна Stimulators, выбрать сигнал Y в окне Waveform Editor. Для него также назначить формульный стимулятор и ввести формулу: 0 0 ns, 1 10 ns, 0 20 ns, 1 30 ns. Нажать кнопку Apply. Выбрать сигнал Cin, не закрывая окна Stimulators, и задать для него горячую клавишу (рисунок 3.36). Нажать клавишу Apply и закрыть окно

Stimulators. После этого окно Waveform Editor должно содержать информацию, такую же как и на рисунке 3.37. Обратите внимание: в начале моделирования сигнал Cin должен иметь значение '0'.

Рисунок 3.36. Назначение горячей клавиши сигналу Cin

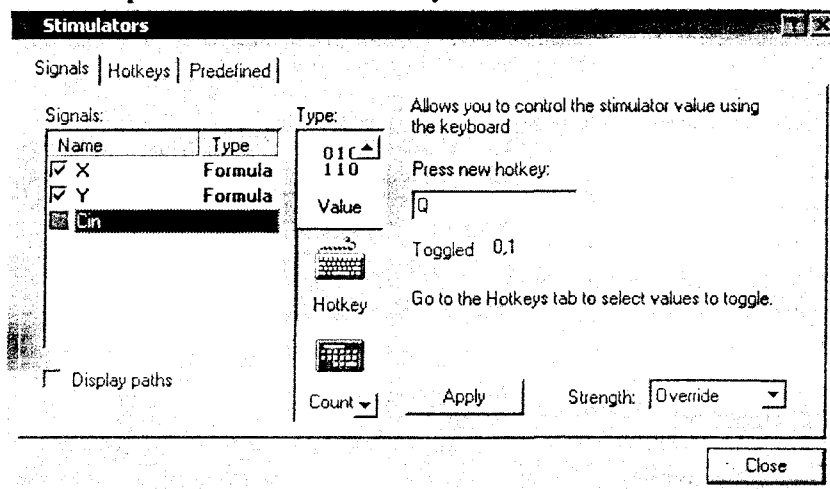


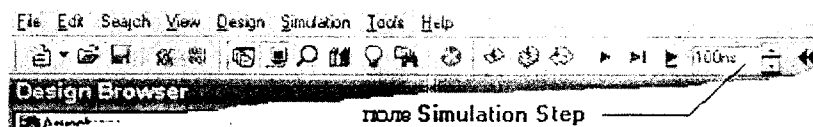
Рисунок 3.37. Окно Waveform Editor с назначенными стимуляторами

Name	Value	Stimula...	
X	0	Formula	
Y	0	Formula	
Cin	0	Q	
Cout	0		
Sum	0		


### Выполнение моделирования


Для проведения моделирования в Active-HDL предлагаются три команды:



1. Команда Simulation /Run запускает моделирование на неопределенный промежуток времени. Моделирование будет закончено при выполнении одного из условий: когда текущее время моделирования станет равным TIME'HIGH; нет ни одного события или другой причины для возобновления выполнения процесса.
2. Команда Simulation /Run For выполняет моделирование в течение описанного временного промежутка (command advances simulation by a specified time step). Для задания этого промежутка используется поле Simulation Step на основной инструментальной панели:



3. Команда Simulation /Run Until запускает выполнение моделирования до описанной временной точки.

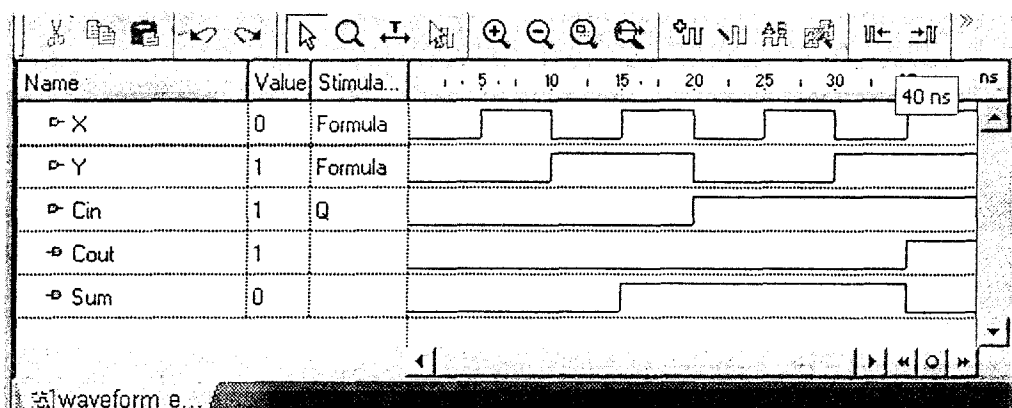
В каждом из представленных выше способов моделирование можно приостановить командой Stop или кнопкой .

Для моделирования полного сумматора в поле Simulation Step следует ввести время 20 ns и нажать кнопку . Затем с помощью горячей клавиши изменить


значение сигнала Cin с '0' в '1'. И снова нажать кнопку , чтобы запустить моделирование еще на 20 ns. Таким образом, будет достигнута точка 40 ns. Иначе можно было бы использовать команду Simulation /Run Until, кнопка . При этом задать точки моделирования сначала 20 ns, затем 40 ns.

Результаты моделирования приведены на рисунке 3.38.

Рисунок 3.38. Результаты моделирования полного сумматора



Для завершения процесса моделирования используется команда Simulation /End Simulation. Программа освобождает память, занимаемую рабочей моделью, и деинициализирует программу моделирования.

Чтобы повторить моделирование, нет необходимости перезагружать модель в программу. Вместо этого следует реинициализировать процесс моделирования командой Simulation /Restart Simulation или с помощью кнопки .

### 3.12. Другие инструменты для наблюдения за результатами и процессом моделирования

#### Окно Watch

Окно Watch – это инструмент отладки, предназначенный для вывода значений объектов (сигналов, переменных), выбранных в тестируемой модели (рисунок 3.39). Отображаются только текущие значения, без какой-либо информации об истории их изменения. Объекты представляются в таком же формате, как и на вкладке Structure окна Design Browser (см. рисунок 3.4).


Окно Watch также позволяет выводить значения формальных параметров и локальных объектов VHDL-подпрограмм. Кроме того, для таких объектов существует и специальный инструмент наблюдения – окно Call Stack.

Рисунок 3.39. Окно Watch

Name	Type	Value	Last ...	Last Event Time
SEL	std_logic	1	0	1500ns
SHIFT	std_logic	0	1	1500ns
CLK	std_logic	0	1	1500ns
RESET	std_logic	0	1	175ns
START	std_logic	0	1	625ns
D	std_logic_va...	4F	00	470ns
RDY	std_logic	1	0	1500ns
TxD	std_logic	1	0	1500ns

Для того чтобы получать значения объектов, их надо добавить в окно Watch. Это можно сделать до инициализации моделирования или в любой момент после. Объект, добавленный в окно Watch, остается в нем, пока не будет удален или до изменения проекта. Окно позволяет наблюдать значения объектов только в момент приостановки моделирования. Формальные параметры и локальные объекты подпрограмм могут быть добавлены в окно только после инициализации моделирования.

Объекты в окне Watch, которые не соответствуют ни одному объекту из тестируемой модели, отмечаются словом Unavailable. Объекты, для которых не указан путь и которые могут одновременно принадлежать модулю верхнего уровня или быть локальными объектами выполняемой подпрограммы, считаются локальными для неё, чтобы избежать неоднозначности.

Для того чтобы открыть или спрятать окно Watch, используется команда Watch из меню View или кнопка .

Для того чтобы разрешить или запретить вывод отдельных колонок окна Watch, следует щелкнуть правой кнопкой мыши по заголовкам колонок и из краткого меню выбрать необходимые пункты. Возможны варианты: Name, Type, Value, Last Value, Event, Last Event Time.

Добавить сигналы в окно Watch можно несколькими способами:

- Переместить их с вкладки Structure окна Design Browser.
- Переместить их из открытого в окне HDL Editor VHDL файла.
- Использовать команды Copy и Paste.
- Набрать имя объекта и путь непосредственно в окне Watch.

В первых двух случаях путь к объекту копируется вместе с именем, если моделирование было инициализировано.

Окно Watch позволяет редактировать значения переменных и констант. Для этого нужно щелкнуть по соответствующей строке в колонке Value и ввести новое значение. Изменение можно вносить только в момент приостановки процесса моделирования.

Для изменения системы счисления, в которой представляются значения, выводимые в окне Watch, необходимо щелкнуть правой кнопкой мыши и выбрать команду Display Options из контекстного меню. Будет открыто окно Preferences. Вкладка Debug позволяет выбрать требуемую систему счисления в полях Numbers и Vectors. Numbers даёт возможность задавать систему счисления для численных значений, а Vectors — для одномерных векторов. Возможны варианты: Binary (двоичная), Octal (восьмеричная), Decimal (десятичная) и Hexadecimal (шестнадцатеричная) системы счисления. Система счисления, задаваемая в диалоговом окне Preferences, является глобальным параметром и оказывает влияние на представление значений в Watch, Call Stack и Design Browser окнах.

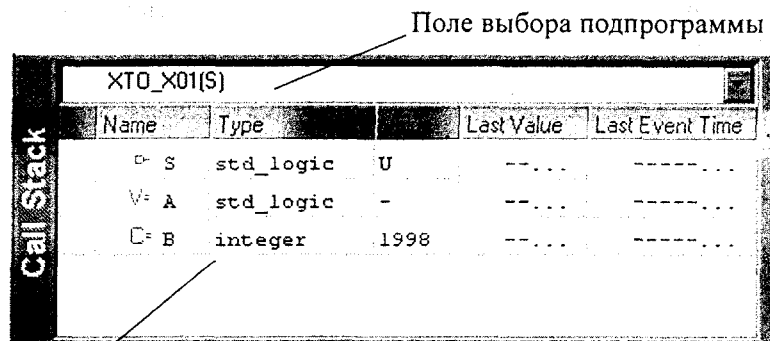
### Окно Call Stack

Окно Call Stack (рисунок 3.40) является инструментом отладки, позволяющим просматривать список подпрограмм (процедур и функций), вызванных в выполняемом в данный момент процессе. Под словом процесс в данном случае подразумевается любой параллельный оператор, например: оператор process, параллельное назначение сигнала, параллельный оператор assert, параллельный вызов процедуры.

Для каждой подпрограммы в окне выводится информация: формальные параметры с реальными значениями, переменные, константы и файлы, являющиеся локальными для подпрограммы (они продекларированы в ней), вместе со своими значениями. Если в моделируемом проекте имеется более одного процесса, можно использовать окно Processes для выбора процесса, подпрограммы которого будут выведены в активном окне Call Stack.

Последнее доступно только во время выполнения моделирования.

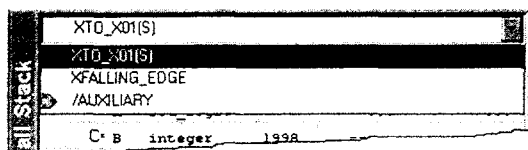
Рисунок 3.40. Окно Call Stack



Список формальных параметров и локальных объектов

Форма представления списка формальных параметров и локальных объектов такая же, как и в окне Watch.

Щелчком по полю выбора подпрограмм открывается список вложенных подпрограмм текущего процесса. Имя процесса расположено внизу списка, а имя самой вложенной подпрограммы – вверху. После ее выбора будет открыто окно **HDL Editor**, содержащее исходный файл этой подпрограммы:



### Окно Processes

Окно Processes является отладочным инструментом, отображающим процессы, которые входят в тестируемую модель. Окно доступно только после инициализации моделирования. Оно содержит три колонки, описывающие метку (label), путь (hierarchical path) и состояние или статус (status) каждого процесса (рисунок 3.41). Для процессов, не имеющих метки, компилятор генерирует псевдометку, представляющую собой номер строки, с которой начинается процесс в исходном коде, например, line\_12.

Рисунок 3.41. Окно Processes

Label	Hierarchy path	Status
UB1	U_BLOCK	Ready
UB2	U_BLOCK	Ready
line_11	U_BLOCK/UB3	Ready
line_11	U_COMPONENT	Ready
/OUTPUT_DRIVER	/	Wait
/U_ASSIGNMENT	/	Wait
/U_PROCESS	/	Wait
/U_PROCEDURE	/	Wait

Процесс может находиться в одном из следующих состояний, которое выводится в колонке Status:

- *Ready(готов)* – означает, что процесс занесен в список (is scheduled) на выполнение во время текущего цикла моделирования. После этого процесс получает статус *<Wait>*.
- *Wait (ожидает)* – процесс ожидает изменения (for an HDL item to change) или окончания описанного периода времени.

Процессы выполняются в порядке, в котором они приведены в окне: сверху вниз. Желтая подсветка означает их выполнение в текущий момент. Порядок следования процессов можно изменить вручную.

Для выбранной области тестируемого проекта окно Processes может выводить или все процессы, независимо от их статуса в текущем цикле моделирования, или только активные в данном цикле. Выделить область для вывода процессов можно в подокне Structure tab окна Design Browser.

### 3.13. Active-HDL Macro Language

Active-HDL-макроязык предоставляет возможность работать в Active-HDL среде, не используя ее графический интерфейс. Макрокоманды могут быть введены непосредственно в окне **Console**. После того как будет нажата кнопка **Enter**, осуществляется выполнение команды. Для инициализации единичных команд такой способ работы не очень удобный. Но макрокоманды незаменимы, когда следует выполнить большую последовательность операций. В таком случае создается текстовый файл, содержащий команды – макрофайл или командный файл, который запускается на выполнение.

Для этого следует набрать команду в окне **Console**:

```
do <filename> [ <parameter_value> ...]
```

Параметр <filename> соответствует имени выполняемого макрофайла. Можно указать полный путь расположения файла. Если никакой путь не указан, по умолчанию подразумевается текущая рабочая директория Active-HDL.

Можно также ввести необязательные аргументы <parameter\_value>, которые будут переданы в макрофайл через параметры \$1...\$99. Если в командной строке описано меньше параметров, чем в макрофайле, неописанные параметры будут заменены пустой строкой.

Пример вызова макрофайла может иметь следующий вид:


```
do $dsn\macrofilename.do
```

Следует обратить внимание, что оператор **do** является макрокомандой и может вызывать командный файл из аналогичного файла. В среде Active-HDL макрофайлы являются файлами ресурса, поэтому могут быть включены в проект как и любые другие ресурсы.

Для того чтобы запустить на выполнение макрофайл из окна Design Browser, необходимо выделить его на вкладке Files, щелкнуть правой кнопкой мыши и выбрать команду Exclude из контекстного меню.

#### Разработка макрофайла

Макрофайл можно создать, используя любой текстовый редактор. Тем не менее, лучше использовать Active-HDL Editor, поскольку он поддерживает цветное выделение ключевых слов макрокоманд.

Для создания нового командного файла следует выполнить команду File /New/ Macro или, щелкнув по стрелке кнопки , выбрать из выпадающего меню пункт Macro. Будет открыто пустое окно HDL Editor, в которое вводится набор макрокоманд. Чтобы добавить созданный файл к проекту, можно использовать команду Add Current Document из меню Design. Если документ до этого момента не был сохранен на диске, появляется диалоговое окно Save As. Далее следует указать имя файла, тип и выбрать папку. Нажать ОК.

#### Использование макрокоманд для моделирования

Рассмотрим макрокоманды, которые были использованы для моделирования 4-битного сумматора Adder (см. рисунок 2.4). Этот макрофайл (рисунок 3.42) инициализирует

процесс моделирования, открывает окно List, добавляет к нему наблюдаемые сигналы и выполняет процесс моделирования. Ниже описаны команды, использованные в командном файле.

**Рисунок 3.42. Макрофайл для моделирования 4-битного сумматора**

```
asim Adder4
list A B Co C Ci S
-- размещение сигналов в окне List
force A 1111
-- присвоение сигналу A значения "1111"
force B 0001
-- присвоение сигналу B значения "0001"
force Ci 1
-- установка Ci в '1'
run 50 ns
-- выполнение моделирования в течение 50 ns
force Ci 0
force A 0101
force B 1110
run 50 ns
```

### Команда **asim**

Инициализирует моделирование.

Синтаксис:

```
asim [ -help ] [ -file <filename> ] [ -i <iteration_limit> ]
<configuration> [ <entity> [ <architecture> ]
```

Аргументы:

-help – вывод короткого описания синтаксиса команды;  
 -file <filename> – описывает необязательный командный файл, содержащий аргументы для команды **asim**. Эти аргументы можно использовать вместо того, чтобы непосредственно вводить их в команде;

-i <iteration\_limit> – устанавливает максимальное число дельта-итераций, которые могут быть выполнены в одно и то же время моделирования. Это позволяет избежать бесконечных циклов;

<configuration> – задает имя конфигурации верхнего уровня для моделирования;

<entity> – имя интерфейса верхнего уровня для моделирования;

<architecture> – имя архитектуры для моделирования. Если имя не указано, будет использована последняя откомпилированная архитектура для данного интерфейса. Этот аргумент может использоваться только с интерфейсом.

### Команда **list**

Открывает окно List или добавляет сигналы к существующему окну.

Синтаксис:

```
list [-in] [-out] [-inout] [-internal] [-ports] [-signals]
[ -collapse ] [ -<radix> ] [ -width <n> <item_name> ... ]
```

Аргументы:

-in – команда влияет только на порты в режиме IN;

-out – команда влияет только на порты в режиме OUT;

-inout – команда влияет только на порты в режиме INOUT;

-internal – команда влияет только на внутренние объекты;

-ports – команда влияет только на порты;



- signals – команда влияет только на сигналы;
- collapse – переключатель между промежуточными и результирующими значениями, которые выводятся на экран. Результирующим считается значение последнего цикла моделирования в описанном времени моделирования. Все значения из промежуточных циклов считаются промежуточными;
- <radix> – задает систему счисления для сигналов, описанных в команде. Доступные опции:
  - binary (abbr. **bin**) – двоичная;
  - octal (abbr. **oct**) – восьмеричная;
  - decimal (abbr. **dec**) – десятичная;
  - hex – шестнадцатеричная;
- width <n> – параметр “n” задает ширину колонки;
- <item\_name> – имя сигнала для размещения в окне List.

### Команда force

Присваивает значение или последовательность значений сигналу.

Синтаксис:

```
force [ -rec | -recursive ] [ -repeat <period> ]
      <signal_name> <value> [ <time> ] [ , <value> <time> ... ]
```

Аргументы:

- rec | -recursive – поиск описанного сигнала по всему проекту. Если аргумент не указан, команда применяется только к текущей области проекта;
- repeat <period> – повторяет последовательность описанных значений с указанным периодом;
- <signal\_name> – имя сигнала. Можно указывать элементы массива, диапазон и весь массив целиком;
- <value> – значение, которое получит сигнал. Тип значения должен соответствовать типу сигнала.

Одноэлементные массивы типа BIT, STD\_ULOGIC и их подтипов могут быть описаны как последовательность символов (01XZ011Z) или как число с указанием системы счисления 2, 8, 10 или 16.

Например, сигналу типа BIT\_VECTOR (3 downto 0) можно назначить следующие значения:

1011	символьная последовательность
2#1011	двоичная система счисления
10#11	десятичная система счисления
16#B	шестнадцатеричная система счисления

<time> – время, в которое сигнал должен получить значение. Это время является относительным для текущего цикла моделирования. Символ @ перед временным параметром обозначает абсолютное время.

### Команда run

Выполняет моделирование.

Синтаксис:

```
run [ <time_step> | @<time> | -all | -next ]
```

Аргументы:

- <time\_step> – промежуток времени для моделирования;
- @<time> – абсолютное время, до которого будет осуществляться моделирование;

-all – выполняются все возможные шаги моделирования, пока не будет обработано последнее событие из очередей драйверов;

-next – выполнение до следующей записи в очередь драйвера.

### Команда wave

Вместо команды **list** или вместе с ней можно использовать команду **wave**, которая позволяет отображать результаты моделирования в виде диаграмм в окне Wave.

Команда добавляет указанные сигналы в окно Wave.

Синтаксис:

```
wave [-in] [-out] [-inout] [-internal] [-ports] [-signals] [-<radix>]
    [ -<format> ] [ -height <pixels> ] [ -color <red_value,
    green_value, blue_value> ] [ <item_name> ] ... ] ...
```

Аргументы:

-<format> – необязательный параметр, задающий один из следующих типов:

– literal (**abbr. li**) – символьный;

– logic (**abbr. lo**) – логический;

– analog – аналоговый.

Символьная waveform представляет собой прямоугольники, в которые вписаны значения. Логические сигналы могут быть либо 1, 0, X или Z.

-height <pixels> – высота waveform в пикселах;

-color <red\_value, green\_value, blue\_value> – цвет waveform, описываемый тремя параметрами с диапазоном значений 0-255, соответствующими красной, зеленой и синей составляющей цвета (RGB-модель цвета);

<item\_name> – имя сигнала.

Аргументы -in, -out, -inout, -internal, -ports, -signals, -<radix> описаны в команде list.

### Команда close

Закрывает указанное окно документа.

Синтаксис:

```
close -wave | -list | -hde | -fsm | -bde
```

Аргументы:

-wave – окно **Waveform**;

-list – окно **List**;

-hde – окно **HDL Editor**;

-fsm – окно **State Editor**;

-bde – окно **Block Diagram Editor**.

Среда проектирования Active-HDL фирмы Aldec – это мощный программный пакет, позволяющий создавать и управлять проектами цифровых устройств. Окно Design Browser предоставляет инструменты для управления файлами проекта, облегчающие процесс их создания, просмотра, копирования, удаления. Существуют мастера для создания новых проектов и генерации шаблонов исходных файлов. Окно Language Assistant содержит стандартные шаблоны VHDL-конструкций и VHDL-модели типовых элементов. Среда Active-HDL реализует возможность моделирования поведения разрабатываемых проектов цифровых устройств. Для просмотра результатов моделирования создано несколько инструментов, облегчающих наблюдение за работой тестируемой модели. Это окна List, Waveform, Watch, Call Stack, Processes.

Макрокоманды позволяют работать со средой Active-HDL без использования ее графического интерфейса.

### 3.14. Задачи

3.14.1. Разработать проект, состоящий из одного VHDL-файла, созданного в задаче 2.15.1. Откомпилировать. Промоделировать, используя для вывода результатов окно Waveform. Для входных сигналов назначить формульные стимуляторы, так чтобы выполнялся перебор всех 16 входных последовательностей. Будьте внимательны с определением задержек переключения входных сигналов.

3.14.2. Для устройства, разработанного в задаче 2.15.2, создать проект. Выполнить моделирование полного и 4-разрядного устройства вычитания. Для инициализации моделирования, присвоения входных значений и вывода результатов моделирования разработать и использовать макрофайл.

3.14.3. В среде Active-HDL создать проект и выполнить тестирование модели, разработанной в задаче 2.15.9.

3.14.4. В среде Active-HDL создать проект и выполнить тестирование модели, разработанной в задаче 2.15.10.

3.14.5. В среде Active-HDL создать проект и выполнить тестирование модели, разработанной в задаче 2.15.11.

3.14.6. В среде Active-HDL создать проект и выполнить тестирование модели, разработанной в задаче 2.15.12.

3.14.7. В среде Active-HDL создать проект и выполнить тестирование модели, разработанной в задаче 2.15.13.

3.14.8. Протестировать функцию, разработанную в задаче 2.15.17. Для этого создать VHDL-модель устройства, в котором бы использовалась эта функция. Для наблюдения за значениями формальных параметров и локальных объектов функции использовать окно Call Stack.

3.14.9. Протестировать функцию, разработанную в задаче 2.15.18. Для этого создать VHDL-модель устройства, в котором бы использовалась эта функция. Для наблюдения за значениями формальных параметров и локальных объектов функции использовать окно Call Stack.