Продолжение. Начало в № 3 `2009

Изучаем Active-HDL 7.1. Урок 3. Альтернативные способы задания внешних воздействий

Александр ШАЛАГИНОВ
shalag@vt.cs.nstu.ru

Создание испытательных стендов (Test Bench)

Мы уже отмечали, что язык VHDL позволяет описывать не только проектируемые цифровые объекты, но и подаваемые на них внешние воздействия, так называемые тест-векторы (**test vectors**). Понятно, что входные воздействия можно задать вручную, однако в пакете **Active-HDL 7.1** в этом нет никакой необходимости. Достаточно вызвать соответствующий «мастер», и вся предварительная работа будет выполнена автоматически.

Более того, «мастер» создаст все условия, чтобы полученные тесты можно было применить к уже имеющемуся цифровому объекту. В среде **Active-HDL 7.1** такой объект называется **UUT — Unit Under Test** — тестируемое или испытуемое устройство.

Другими словами, «мастер» автоматически сгенерирует программу, называемую испытательным стендом (**Test Bench**), в виде объекта проекта, содержащего внутри себя испытуемое устройство и подаваемые на него тест-векторы (рис. 1).

Чтобы убедиться воочию, как работает «мастер», вновь сделаем активным проект **Lesson_1**, созданный на первом уроке, и из меню **Tools** командой **Generate Testbench** запустим «мастер» генерации испытательного стенда (**Test Bench Generator Wizard**).

«Мастер» предложит выбрать модуль проекта, для которого требуется сгенерировать испытательный стенд. В нашем простом проекте имеется всего один такой модуль — **mux2_schema**, поэтому выбора в действительности нет (рис. 2).

По умолчанию переключатель **Test Bench Type** установлен в положение **Single Process** (одиночный процесс). В этом режиме генерируются только входные сигналы, что нас вполне устраивает.

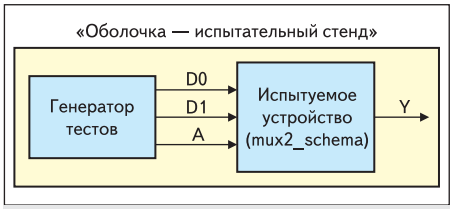

Рис. 1. Испытательный стенд (Test Bench) мультиплексора на два входа mux2_schema

На второй панели (рис. 3) необходимо установить флажок **Test vectors from file**, потому что для генерации тестовых наборов мы собираемся использовать ранее созданные временные диаграммы, сохраненные в волновом файле **mux2_schema.awf** (урок 1).

Нажав кнопку Browse, отыщем нужный файл с результатами моделирования, полученными на первом уроке, и откроем его. В левом нижнем окне диалоговой панели (**Signals found in file**) появится список сигналов (**A**, **D0**, **D1**), для которых будут созданы VHDL-тесты. Обратите внимание, эти сигналы соответствуют именам портов тестируемого устройства **mux2_schema** (правое окно).

На третьей панели (рис. 4) задается спецификация испытательного стенда: имя объекта проекта (**mux2_schema_tb**) и архитектуры (**TB_ARCHITECTURE**), а также имя файла, куда будет помещен исходный VHDL-код испытательного стенда (**mux2_schema_TB.vhd**) и название папки (**TestBench**), в которой будут находиться все генерируемые данные.

Перечисленные имена «мастер» предлагает по умолчанию, так что на этой панели лучше вообще ничего не трогать.

На последней панели, которую мы не станем показывать, «мастер» уведомляет нас, что он готов сгенерировать файлы испытательного стенда, и называет их имена еще раз. В среднем разделе этой панели вы можете дополнительно задать генерацию файла для временного моделирования (**for timing simulation**), но мы не станем использовать такую возможность и оставим флажок **Generate** сброшенным.

В нижнем разделе «мастер» информирует о том, что для запуска проекта на моделирование будет создан макрофайл **mux2_schema_TB_runtest.do**, благодаря которому весь эксперимент можно выполнить автоматически — одним щелчком мыши.

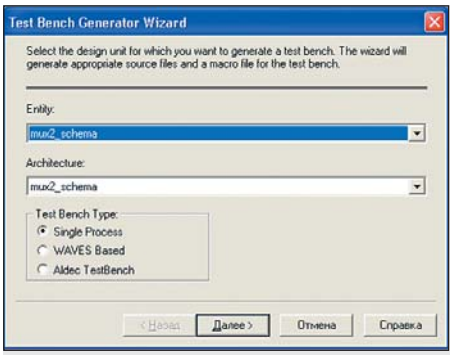

Рис. 2. Диалоговая панель «мастера» генерации испытательного стенда

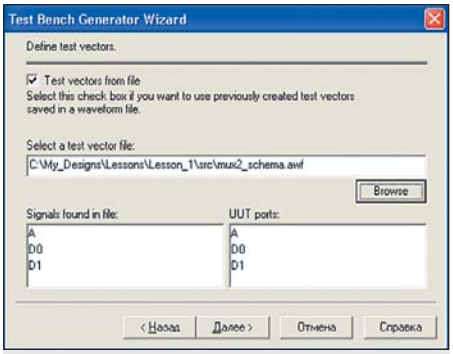

Рис. 3. Подключение волнового файла mux2_schema.awf, из которого будут генерироваться тестовые сигналы A, D0 и D1

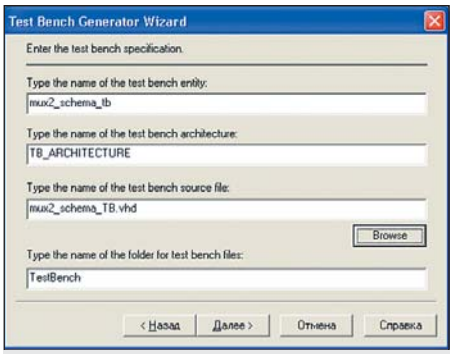

Рис. 4. Информация о расположении и генерируемых файлах верификационной среды

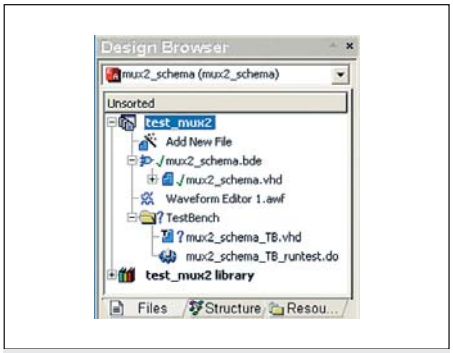

Рис. 5. «Мастер» генерации испытательного стенда создал папку TestBench с двумя файлами

По окончании работы «мастера» вы обнаружите, что в окне просмотра проекта **Design Browser** появилась новая папка **TestBench** (рис. 5), внутри которой находятся два обещанных «мастером» файла.

Откройте первый из них, чтобы посмотреть, как выглядит испытательный стенд в формате языка VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
-- интерфейс испытательного стенда
entity mux2_schema_tb is
end mux2_schema_tb;
-- архитектурное тело испытательного стенда
architecture TB_ARCHITECTURE of mux2_schema_tb is
   component mux2_schema  -- декларация тестируемого устройства
   port ( A, D0, D1 : in std_logic;
        Y : out std_logic );
   end component;
   signal A, D0, D1 : std_logic;  -- объявление сигналов
   signal Y : std_logic;
   signal END_SIM: BOOLEAN:=FALSE;  -- «флажок» окончания
                                    -- моделирования
begin
-- карта порта испытуемого устройства (Unit Under Test port map)
   UUT : mux2_schema port map (A => A, D0 => D0, D1 => D1, Y => Y);

STIMULUS: process       -- процесс, контролирующий окончание
                        -- моделирования
begin
 wait for 1200 ns;      --1200 ns — время моделирования
   END_SIM <= TRUE;  -- установка «флажка» окончания
                        -- моделирования
   wait;
end process;
-- генераторы тестов
CLOCK_D0 : process      -- процесс, генерирующий периодический
                        -- сигнал с частотой 10 МГц
begin
   if END_SIM = FALSE then D0 <= '0';     -- текущее время 0 fs
       wait for 50 ns; else       wait;
   end if;
   if END_SIM = FALSE then D0 <= '1';     -- текущее время 50 ns
       wait for 50 ns; else       wait;
   end if;
end process;
CLOCK_D1 : process    -- процесс, генерирующий периодический
                      -- сигнал с частотой 5 МГц
begin
-- аналогично первому процессу, только время ожидания равно 100ns
end process;
CLOCK_A : process     -- процесс, генерирующий периодический
                      -- сигнал с частотой 1 МГц
begin
-- аналогично первым двум процессам, только время ожидания
-- равно 500ns
end process;
end TB_ARCHITECTURE;
-- конфигурация испытательного стенда
configuration TESTBENCH_FOR_mux2_schema of mux2_schema_tb is
   for TB_ARCHITECTURE
      for UUT : mux2_schema
         use entity work.mux2_schema(mux2_schema);
      end for;
   end for;
end TESTBENCH_FOR_mux2_schema;
```

В целях экономии места он слегка подредактирован по отношению к оригиналу и снабжен комментарием. Разбираясь в содержимом этого файла, полезно освежить в памяти структуру испытательного стенда, показанную на рис. 1.

Как видно, автоматически сгенерированный испытательный стенд имеет такую же структуру, что и обычный объект проекта. Он содержит интерфейс с именем **mux2_schema_tb** и архитектурное тело **TB_ARCHITECTURE**.

Единственная его особенность — в нем нет портов, а значит, отсутствуют и связи с внешним миром. Кажется, это совершенно бессмысленная «вещь в себе». Но как говорится, первое впечатление обманчиво. Перед нами «оболочка», в которую помещается тестируемый объект и подаваемые на него внешние воздействия. Эта оболочка, называемая испытательным стендом или верификационной средой, и предназначена для имитации процесса тестирования цифровой аппаратуры.

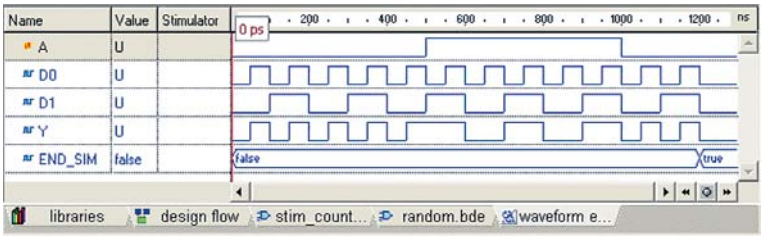

Рис. 6. Результаты моделирования мультиплексора mux2_schema, полученные с помощью испытательного стенда

Генератор тестов может быть оформлен как самостоятельный объект проекта, но проще, как в нашем примере, его «встроить» в оболочку в виде отдельных процессов: **CLOCK_D0**, **CLOCK_D1** и **CLOCK_A**.

«Мастер» реализует именно эту, вторую возможность. Обратите внимание на один очень важный момент: внешние воздействия являются «внешними» только для тестируемого объекта, для испытательного стенда они остаются внутри «оболочки». Понятно, что теперь вы можете изменять диаграммы входных сигналов, редактируя VHDL-код испытательного стенда (раздел «генераторы тестов») и не пользоваться более услугами стимуляторов.

В окне просмотра проекта **Design Browser** щелкнем правой кнопкой мыши на строке **mux2_schema_TB_runtest.do** и исполним команду **Execute** («Выполнить макрофайл»). Система **Active-HDL 7.1** в автоматическом режиме выполнит все команды, записанные в указанном файле. Мы увидим результаты моделирования (рис. 6), близкие к тем, что были получены на первом уроке.

Единственное отличие — это пустой столбец **Stimulator**. Объяснение «лежит на поверхности»: теперь для описания внешних воздействий используются не стимуляторы, а тестовые VHDL-векторы, включенные в архитектурное тело испытательного стенда. Записаны они в текстовом формате, и пользователь их может редактировать.

Использование силовых команд (force command)

Откроем макрофайл **mux2_schema_TB_runtest.do**, ранее созданный «мастером» генерации испытательного стенда:

```
SetActiveLib -work
comp -include "$DSN\src\mux2_schema.bde"
comp -include "$DSN\src\TestBench\mux2_schema_TB.vhd"
asim TESTBENCH_FOR_mux2_schema    -- имя конфигурации
wave
wave -noreg A
wave -noreg D0
wave -noreg D1
wave -noreg Y
run 1200.00 ns
```

Как видно, это текстовый файл, в котором записано несколько команд макроязыка **Active-HDL**. Даже не зная названного языка, можно догадаться о назначении некоторых из них. Например, команда **run 1200.00 ns** запускает процесс моделирования на 1200 нс. С помощью команды **comp** (от слова **compilation**) выполняется компиляция исходных файлов. Команда **asim** (от слова **simulation**) вызывает имитатор. Команда **wave** открывает редактор временных диаграмм **Waveform Editor**, и в него загружаются сигналы **A**, **D0**, **D1** и **Y**. Некоторые из перечисленных команд вам наверняка попадались в окне **Console**.

Другими словами, макрофайл среды **Active-HDL** напоминает хорошо знакомый командный BAT-файл операционной системы MS DOS и служит для тех же целей.

Макроязык пакета **Active-HDL** разработан как альтернатива графическому пользовательскому интерфейсу (GUI). Вместо того чтобы щелкать мышкой на кнопках и пиктограммах, вы можете прямо с клавиатуры вводить аналогичные команды непосредственно в командную строку окна **Console**.

Подготовим уже открытый проект для нового эксперимента. Для этого нам придется удалить все следы предыдущего опыта, связанного с испытательным стендом. Щелкнем правой кнопкой мыши на рабочей библиотеке **Lesson_1 library** и выполним команду **Delete simulation data**. Удалим из рабочего проекта и папку **TestBench**. Наконец, закроем окно редактора **Waveform Editor** и откомпилируем проект заново.

В командной строке **Console** введем макрокоманду **wave D0 D1 A Y** и нажмем **Enter**. Откроется окно редактора **Waveform Editor**, и в него загрузятся перечисленные в команде сигналы **D0**, **D1**, **A** и **Y**. Далее запрограммируем временные диаграммы всех входных сигналов. Эта работа выполняется макрокомандой **force** (силовая команда). Одну за другой введем с клавиатуры следующие команды:

```
force D0  0 0ns, 1 50ns    -repeat   100ns
force D1  0 0ns, 1 100ns   -repeat   200ns
force A   0 0ns, 1 500ns,  0         1000ns
```

Обратите внимание, система ведет себя так, как если бы сигналы программировались стимуляторами типа **formula**. Откройте диалоговую панель **Stimulators** и вы увидите, что

все данные, введенные в командной строке, продублированы на диалоговой панели.

Однако при повторном моделировании вас ждет разочарование: придется заново вводить описания входных сигналов. Более того, даже при повторном старте все данные о параметрах входных сигналов будут потеряны.

Чтобы избежать такой неприятности, лучше создать макрофайл, например, **Macro1.do** (команда **File/New/Macro**), и записать в него все необходимые макрокоманды:

```
wave D0 D1 A Y
asim mux2_schema
force D0  0 0ns, 1 50ns    -repeat  100ns
force D1  0 0ns, 1 100ns   -repeat  200ns
force A   0 0ns, 1 500ns,  0        1000ns
run 1400ns
```

Можно облегчить себе работу, если воспользоваться услугами языкового помощника **Language Assistant** (о нем речь впереди) и активизировать категорию **Macro**.

Для выполнения самого макрофайла достаточно ввести в окне **Console** команду **do macro1.do** и нажать **Enter**.

Рисование временных диаграмм сигналов в графической форме

Редактор **Waveform Editor** позволяет не только наблюдать результаты моделирования, но и редактировать формы временных диаграмм как входных, так и выходных сигналов. Такая возможность появляется только после окончания процесса моделирования. Для этого нужно выполнить команду **End Simulation** из меню **Simulation** или ввести в окне **Console** макрокоманду **endsim**.

Более того, в редакторе **Waveform Editor** вы можете сначала нарисовать графики временных диаграмм, а затем выполнить модельный эксперимент. С такой возможностью вы могли встретиться, знакомясь, например, с пакетом **DesignLab 8.0** фирмы **MicroSim**. В названной САПР имеется графический редактор внешних воздействий, именуемый **Stimulus Editor**. Нечто похожее предусмотрено и в среде **Active-HDL 7.1**.

Познакомимся с процессом рисования и редактирования графиков временных диаграмм в программе **Waveform Editor**. Для начала выполним самую простую работу: займемся рисованием форм сигналов в режиме 0–1.

Откроем новое окно редактора и добавим в него сигналы **D0**, **D1**, **A** и **Y**. На инструментальной панели найдем кнопку **Edit Mode** и переведем редактор в режим редактирования. Обратите внимание, указатель мыши сменил свою форму и теперь он похож на знак + (плюс).

Чтобы появилась возможность рисовать графики, понадобится выполнить инициализацию моделирования (команда **Initialize Simulation** из меню **Simulation**). А затем можно приступить к работе. Щелкнем мышью в начале графика сигнала **D0**. Курсор приобретет форму мигающей вертикальной черточки. Положение курсора можно изменять с помощью управляющих клавиш. По умолчанию шаг перемещения курсора равен 50 нс, но его легко изменить командой **Preferences** из меню **Tools**. Для этого на диалоговой панели с одноименным названием в окне **Category** нужно выделить строку **Waveform Editor** и в поле **Grid length** ввести желаемое значение.

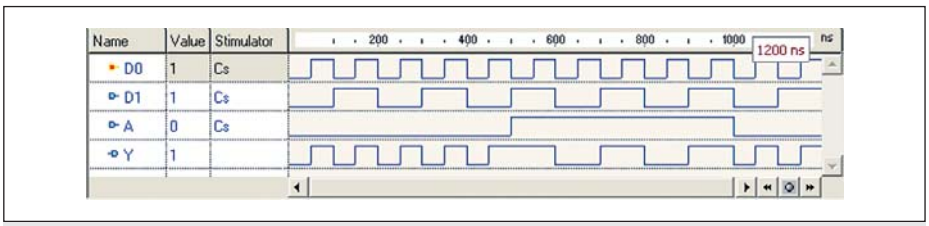

Рис. 7. Рисуем графики внешних воздействий D0, D1, A и задаем им тип стимулятора Custom

Последовательно нажимая на клавиши 0 и 1, нарисуем график сигнала **D0**. Аналогичным образом создадим графики для сигналов **D1** и **A** (рис. 7).

Чтобы не потерять в дальнейшем нарисованные временные диаграммы и привязать их к соответствующим сигналам, зададим каждому из них тип стимулятора **Custom** (заказной). Другими словами, мы жестко фиксируем формы временных диаграмм, не позволяя изменять их другими возможными инструментами.

Использование стимулятора типа **Custom** не ограничивается приведенным примером. Если в процессе моделирования вы использовали стимуляторы типа **Hotkey** и **Value** и в следующем эксперименте хотите сохранить созданные с их помощью графики, то без стимулятора **Custom** вам не обойтись.

При моделировании проекта убеждаемся, что система «понимает» нарисованные графики временных диаграмм и корректно выполняет предлагаемую работу.

Получив представление о режиме рисования графиков временных диаграмм, более детально познакомимся с другими его инструментами.

Итак, чтобы перевести программу **Waveform Editor** в режим редактирования, достаточно щелкнуть по иконке **Edit Mode**, расположенной на инструментальной панели. Кроме того, эта команда дублируется в выпадающем меню **Waveform** и в контекстном меню, которое «всплывает» при щелчке ПКМ в зоне временных диаграмм.

Визуально обнаружить переключение программы в режим редактирования очень легко: к списку сигналов снизу добавляется строка **Click here to add new signal** («Щелкните здесь, чтобы добавить новый сигнал»). Еще одна деталь, которая помогает понять, что программа находится в режиме редактирования: при перемещении курсора в правую часть окна волнового редактора он приобретает форму значка «+», но это мы уже знаем.

Отметим, что в режим редактирования можно перейти независимо от того, активен процесс моделирования или нет. Но попытка изменить форму какого-либо графика, нарисованного на желтом фоне (признак активности имитатора), приведет к неудаче. Редактор не позволит это сделать, выдав сообщение: **Simulator result cannot be edited while simulator is running** («Результаты имитатора не могут быть отредактированы во время его работы»).

Для получения полной свободы действий вам нужно закончить моделирование или хотя бы выполнить команду **Restart Simulation**.

Основные инструменты редактирования временных диаграмм

По-видимому, чаще всего возникает необходимость перемещения влево или вправо событий (переключений) одного сигнала относительно другого. Этим приходится заниматься при нарушении временных соотношений в схеме, например времени предустановки или удержания. Возможно, моделирование обнаружит слишком короткие импульсы, которые придется расширять.

Такая работа выполняется чрезвычайно просто. Надо «наехать» мышью на редактируемое событие, и когда курсор примет форму двунаправленной стрелки ↔, нажать **ЛКМ** и отбуксировать переключение в нужное место (рис. 8).

Во время данной операции курсор сопровождает небольшое окно (рис. 8), в котором указывается текущее положение события на оси времени. Эта информация позволяет более точно выполнить операцию.

Установив привязку к сетке (**Snap to grid**) и изменяя масштаб изображения графиков, можно добиться перемещения события с заданным шагом сетки (по умолчанию 50ns). Изменяя шаг сетки (**Grid length**), легко получить подходящую разрешающую способность операции перемещения.

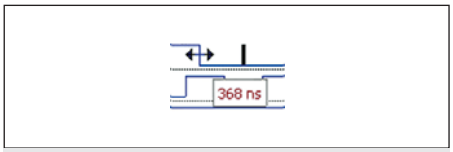

Рис. 8. Перемещение переключения в необходимое место

Если перемещаемое событие «наезжает» на соседнее событие, последнее поглощается. Когда переключения происходят между '0' и '1', такие действия приводят к исчезновению обоих событий. Так можно удалять ненужные на графиках импульсы.

Когда вы задаете диаграммы внешних воздействий с помощью стимулов, возникают определенные трудности их взаимного согласования во времени. Программируя очередной сигнал, вы не видите ранее введенных временных диаграмм. Они появятся только после пробного моделирования, которое и выявит все дефекты «вслепую» выполненной работы.

Для их устранения приходится перемещать один график относительно другого, начиная с выбранного события. Нажмите клавишу **Insert**, чтобы перевести клавиатуру в режим вставки. А дальше действуйте так же, как при перемещении события. Вы увидите, что вслед за курсором тянется вся правая часть временной диаграммы.

Если вам потребовалось вставить короткий импульс в уже существующий график, то следует нажать клавишу **Shift**, указать курсором нужное место (он примет форму карандаша ◪) и щелкнуть левой кнопкой мыши (рис. 9). Ширина вставленного импульса определяется в пикселях и зависит от текущего масштаба изображения временных диаграмм. Обычно по окончании операции ширину импульса приходится редактировать вручную.

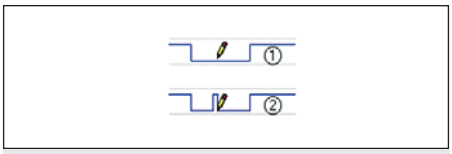

Рис. 9. Вставка короткого импульса в диаграмму

Существует другой, более надежный способ добавления в график прямоугольных импульсов. Он отличается от предыдущего тем, что гарантирует заранее установленную ширину импульса, равную текущему шагу сетки (**Grid length**). Для наглядности можно сделать сетку видимой, установив флажок **Grid visible**.

Подведите курсор к тому месту, где должен появиться импульс. Когда курсор приобретет вид двунаправленной вертикальной стрелки (рис. 10), нажмите **ЛКМ** и переместите его в направлении вершины будущего

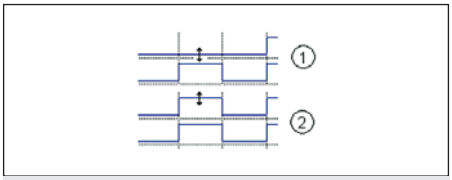

Рис. 10. Более надежный способ добавить в график прямоугольный импульс

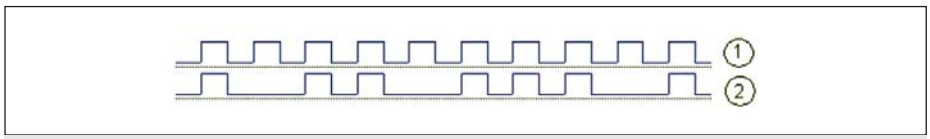

Рис. 11. Из периодического сигнала получаем «пачки» импульсов возрастающей длины

импульса (рис. 10). Увидев результат, отпустите кнопку мыши. Выполнив эти же операции в обратном порядке, вы легко избавитесь от созданного импульса.

В некоторых случаях вам может потребоваться импульс со значением '**X**', '**Z**', '**U**', '–', '**W**', '**H**' или '**L**'. Если вы будете действовать «прямолинейно», у вас ничего не получится. А вот «окольным путем» можно достичь желаемого результата.

Создайте предварительно импульс со значением '1' или '0', а затем отредактируйте его значение. Для этого двойным щелчком **ЛКМ** выделите импульс (занимаемый им временной интервал окрасится в серый цвет) и нажмите клавишу с нужным значением. Данную операцию желательно выполнять при включенном режиме **Caps Lock**.

До сих пор мы говорили о редактировании временных диаграмм, полученных в результате ранее выполненного моделирования. Но при желании вы можете сначала нарисовать графики внешних воздействий и только затем выполнить моделирование. Такая последовательность действий представляется более логичной. Кроме того, проектируя испытательный тест, вы визуально контролируете взаимное расположение временных диаграмм. Это повышает вероятность избежать ошибок.

Впрочем, мы уже немного знакомы с технологией проектирования временных диаграмм «с нуля», когда рисовали графики в режиме 0–1, поочередно нажимая на соответствующие клавиши. Это самый простой, но далеко не самый эффективный способ создания периодических последовательностей.

Теперь мы попробуем выполнить ту же работу более подходящими инструментами. Сначала выделим область для будущего графика. С этой целью подведем указатель к началу координат создаваемого графика и нажмем **ЛКМ**. Не отпуская кнопку, переместим указатель мыши вправо, определив тем самым желаемую длину графика. Выделенная область окрасится серым цветом.

Щелкнем правой кнопкой мыши в области выделения и исполним команду **Fill** («Заполнение»). Откроется соответствующая панель, и вы увидите хорошо знакомые иконки **Clock**, **Formula** и **Value**. Выберем способ заполнения, например, **Clock**, и зададим желаемые параметры. Выделенная область заполнится периодическими импульсами.

Возникает законный вопрос, где же выигрыш, если в обоих случаях используются одни и те же инструменты? Понятно, что в скорости создания графика никакого выигрыша нет. Но теперь у вас есть возможность, как говорится, «на лету» отредактировать полученный график. Например, вы можете «выбросить» лишние импульсы и получить временную диаграмму, показанную на рис. 11.

Кроме того, следующий график мы будем создавать на фоне ранее нарисованных, что решает все проблемы их взаимного согласования.

При выделении области редактирования есть некоторые нюансы. Например, вам требуется выделить определенный участок временной диаграммы от одного события до другого. Но как только указатель мыши приближается к фронту или срезу, автоматически включается режим перемещения событий (появляется двунаправленная стрелка) и не дает вам выполнить задуманное.

Оказывается, очень легко воспрепятствовать включению этого режима: нужно лишь во время операции выделения удерживать нажатой клавишу **Ctrl**. И таких деталей много. Обсудить их все нет никакой возможности, поэтому назовем лишь самые важные.

Выделить область редактирования можно не только мышью, но и управляющими клавишами. Если щелкнуть мышью в зоне временных диаграмм, то появится мигающий символ в виде небольшой вертикальной черточки (на рис. 12 внизу). Его можно перемещать с помощью управляющих клавиш по обеим осям. Дискретность перемещения определяется установленным шагом сетки (**Grid length**). Теперь в вашем распоряжении два указателя: один управляется мышью, другой клавиатурой.

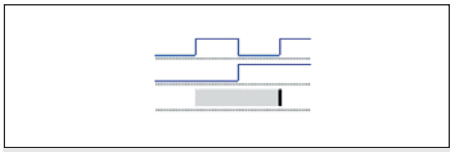

Рис. 12. Выделение области редактирования

При нажатой клавише **Shift** (или **Shift** + **Ctrl**) происходит формирование области выделения, которая может охватывать сразу несколько графиков. Добавим, что клавиша **Home** работает во всех случаях, тогда как противоположная клавиша **End** дает эффект лишь для существующих графиков.

В процессе построения временных диаграмм нередко возникает необходимость удалить «хвост» слишком длинного графика. Выделение и последующая команда **Delete** не дают ожидаемого результата: переключения исчезают, но уровень сигнала остается.

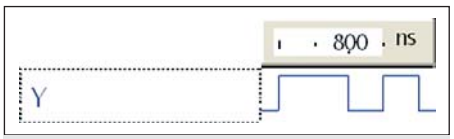

Рис. 13. Вставка фрагментов временных диаграмм в текстовый процессор

Чтобы все получилось, следует перевести клавиатуру в режим вставки, нажав клавишу **Insert**.

Очень удобно в процессе проектирования временных диаграмм использовать операции, связанные с буфером обмена (вырезать, скопировать, вставить).

Кстати, скопированные фрагменты временных диаграмм легко вставляются в текстовый процессор, в частности, в **Microsoft Word_XP** (рис. 13).

Еще одна интересная особенность волнового редактора — умение изменять размеры любых фрагментов графиков. Выделив один или несколько графиков, выполните команду **Stretch** из всплывающего меню. В результате выделенная часть диаграммы изменит свой масштаб (растянется или сожмется) в соответствии с установленным параметром **Scale** (масштаб). Таким способом можно изменить даже ширину импульса.

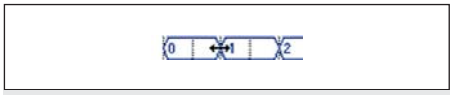

Рис. 14. Редактирование диаграмм шинных сигналов

Время от времени возникает необходимость проинвертировать сигнал. Например, активным уровнем при сбросе для одних устройств является '1', а для других '0'. Чтобы не перерисовывать весь график заново, достаточно выделить его и выполнить команду **Invert** из контекстного меню.

Временные диаграммы шинных сигналов редактируются теми же инструментами, что и для одиночных сигналов (рис. 14). При этом неважно, работаете вы с реальной или виртуальной шиной. Например, переключения на шине перемещаются точно так же, как мы это делали для одиночных графиков.

Если шина раскрыта и видны ее отдельные сигналы, то вносить изменения допустимо как в шинный сигнал, так и в одиночные сигналы, составляющие шину. Кстати, чтобы изменить значение шины, нужно дважды щелкнуть на ее литеральном значении и, убедившись, что режим редактирования включился (указатель мыши примет форму вертикальной черты и начнет мигать), ввести новое значение.

Для рисования шины оказывается пригодным и режим 0–1. Правда, в этом случае шина будет принимать значения 0…0 и F…F, то есть «заполняться» нулями или единицами. Аналогичным образом, нажимая на клавиши **U**, **X**, **Z**, **W**, **H**, **L** или '–', вы будете заполнять соответствующими значениями все разряды шины.

Заканчивая рассказ о волновом редакторе, напомним, что он сохраняет все результаты своей работы в текстовых файлах ***.awf**. Далеко не всегда это результаты моделирования. Вы можете нарисовать временные диаграммы входных сигналов, «повесить» на них стимуляторы типа **Custom** и сохранить их для будущих экспериментов, например в файле **Waveform Editor_input.awf**. Перед началом нового моделирования загрузите названный файл, и вам не придется заново выполнять уже проделанную работу.

Кроме того, волновой редактор позволяет экспортировать временные диаграммы в другие форматы, используемые обычно для создания эталонных тестовых наборов. Для языка VHDL наиболее часто применяется **HSL**-формат. Чтобы выполнить эту операцию, достаточно запустить команду **Export/Waveform** из выпадающего меню **File**.

Продолжение следует