

Продолжение. Начало в № 3 `2009

Александр ШАЛАГИНОВ  
shalag@vt.cs.nstu.ru

# Изучаем Active-HDL 7.1.

## Урок 2.

### Как задавать внешние воздействия с помощью стимуляторов

Прежде чем начать моделирование схемы, надо определить сигналы на ее входах. В среде Active-HDL 7.1 внешние воздействия задаются несколькими способами:

- с помощью виртуальных генераторов сигналов, называемых стимулами или стимуляторами (**Stimulators**);
- с использованием VHDL- или Verilog-описания входных сигналов (создание испытательных стендов — **Test Bench**);
- представлением силовых команд (**force**) в текстовом формате и вводом их в систему из командной строки окна **Console**;
- с применением существующих или рисованием новых временных диаграмм в графическом редакторе **Waveform Editor**.

Большинство разработчиков цифровой аппаратуры предпочитают моделировать свои проекты, ориентируясь на временные диаграммы, то есть привычные описания сигналов в графической форме (**waveform**). Кому-то больше нравится текстовое описание сигналов на языках HDL (испытательные векторы — **test vectors**). Однако для оптимальной верификации проекта, вероятнее всего, потребуются и те, и другие способы описания. Поэтому придется познакомиться со всеми разновидностями.

На данном уроке мы обсудим только работу со стимулами, остальные варианты будут рассмотрены на третьем уроке.

Схемотехнику, имеющему дело с реальной аппаратурой, ближе и понятнее всего стимуляторы (или стимулы) — виртуальные генераторы сигналов. В некоторых САПР, например в DesignLab 8.0 фирмы MicroSim, такие генераторы имеют даже графические образы и помещаются на схеме подобно другим реальным компонентам: логическим вентилям, триггерам, цифровым узлам и блокам.

В пакете Active-HDL 7.1 стимуляторы — это «генераторы-невидимки», они назначаются (приписываются) сигналам или портам, но на схеме не отображаются. Мы уже познакомились с ними на первом уроке. Теперь предстоит более серьезная работа.

Всего в среде Active-HDL 7.1 имеется восемь типов стимуляторов, что позволяет описать практически любую форму цифрового сигнала, какой бы сложной она не была.

#### Стимуляторы типа Value и Hotkey

Самые простые (и в чем-то похожие между собой) типы стимуляторов — это **Value** (значение) и **Hotkey** (горячая клавиша). С их помощью выбранному сигналу или порту задается постоянное значение, например '0' или '1'.

Эти стимуляторы напоминают работу обычного механического тумблера. В одном положении тумблера сигнал получает значение '0', в другом — '1'. Самое ценное в том, что значения можно менять непосредственно в процессе одного модельного эксперимента.

Виртуальные генераторы сигналов типа **Hotkey** меняют текущее значение при нажатии пользователем соответствующей горячей клавиши, генераторам типа **Value** можно задать новое значение на открытой диалоговой панели **Stimulators** в паузе между соседними шагами моделирования. Другими словами, сигналы разрешается изменять интерактивно в ходе моделирования. Посмотрим, как это делается на практике.

В том же самом рабочем пространстве **Lessons** (мы его организовали на первом уроке) создадим новый проект **Lesson\_2**. Это можно сделать многими способами. Проще всего раскрыть крайнюю левую пиктограмму на инструментальной панели **Standard** и выбрать кнопку **New Design** (рис. 1). Процедура создания нового проекта подробно рассматривалась на первом уроке, и мы не станем погружаться в детали. Отметим только, что новый проект автоматически станет активным (будет выделен жирным шрифтом).

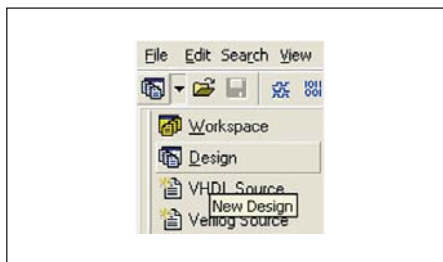


Рис. 1. Пиктограмма на инструментальной панели Standard

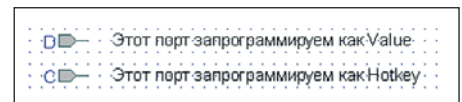





Рис. 2. «Схема» для тестирования стимуляторов типа Value и Hotkey

Вызовем схемный редактор **Block Diagram Editor** (с этой работой мы тоже познакомились на первом уроке), назовем схемный файл **value\_hotkey** и определим для него два входных порта — **D** и **C**. Мы даже не станем подписывать к ним проводники (рис. 2). Откомпилируем пустую схему.

Затем вызовем редактор **Waveform Editor** (иконка ) и добавим в него имеющиеся входные порты **D** и **C** (иконка ). Выделим первый из них и активизируем команду **Stimulators** (иконка ). В окне **Type** найдем нужный тип стимулятора **Value** и справа от него в поле **Force value** выберем желаемое значение, например '0'. После этого нажмем кнопку **Apply** — «применить» (рис. 3).

Порту **C** назначим стимулятор типа **Hotkey**. С этой целью, не закрывая панель **Stimulators**, щелкнем в окне редактора **Waveform Editor** на сигнале **C**. Он окажется в списке сигналов открытой панели **Stimulators**.

Выберем для него тип **Hotkey** и в поле **Press new hotkey** (рис. 4) определим горячую клавишу (например, **C**), которая будет им управлять. В данном примере мы назвали горячую клавишу тем же именем, что и порт, только для того, чтобы не запутаться. Вы можете «привязать» стимул к любой другой клавише.

По умолчанию горячая клавиша переключает сигнал между двумя уровнями: '0' и '1'

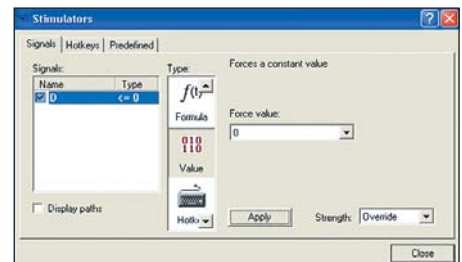


Рис. 3. Задаем постоянное значение входному порту D с помощью стимулятора Value

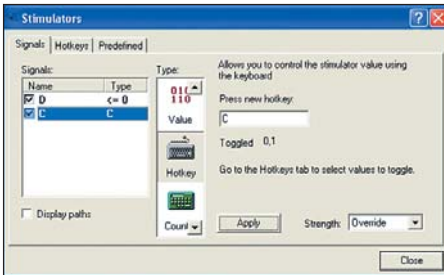


Рис. 4. Назначаем порту С стимулятор типа Hotkey с управляющей клавишей С

(Toggled 0,1). Зафиксируем сделанные назначения, нажав кнопку **Apply**. Диалоговую панель **Stimulators** оставим открытой, она нам пригодится в ходе моделирования.

Попробуем выполнить моделирование. Кнопкой **Run for** продвинем модельное время на один шаг (по умолчанию на 100 ns), и нажмем горячую клавишу С. Мы увидим, что сигнал С переключился с '0' на '1' (см. столбец **Value** на рис. 5). Другой сигнал D сохранил прежний уровень. Продвинемся еще на один шаг и снова нажмем клавишу С. Отметим, что сигнал С вернулся на прежний (низкий) уровень (рис. 5).

Теперь сделаем то же самое с сигналом D. Выделим его на панели **Stimulators**, в поле **Force value** определим для него новое значение '1' и нажмем кнопку **Apply**. Вы обнаружите, что сигнал D изменил свое значение, и последующий шаг моделирования подтвердит это. Проведите еще несколько экспериментов, чтобы получить временные диаграммы, показанные на рис. 5.

Работая со стимулами **Value** и **Hotkey**, вы, конечно, заметили, что кроме '0' и '1' они могут принимать еще семь значений: 'U', 'X', 'Z', 'W', 'L', 'H', '-' (определяя полный набор цифрового сигнала типа **std\_logic**).

Чтобы просмотреть весь спектр возможных значений для стимула С типа **Hotkey**, выделим его на панели **Stimulators** и активизируем закладку **Hotkey**. Мы увидим, что заданная по умолчанию последовательность (**Sequence**) содержит только '0' и '1'.

Именно они и появлялись, циклически сменяя друг друга, в предыдущем эксперименте. Добавим в эту строку все оставшиеся значения и повторим моделирование (рис. 6). Обратите внимание, некоторые значения, например '0' и 'L', '1' и 'H', 'X' и 'W', отличаются только цветом, и при черно-белой печати станут практически неразличимы.

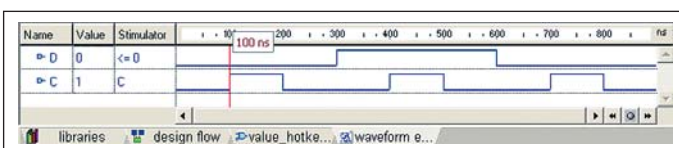


Рис. 5. Выполняем в пошаговом режиме моделирование пустой схемы, изменяя вручную сигналы на входах D и С

### Стимуляторы типа Clock (синхронизация) и Formula (формула)

Стимуляторы **Clock** применяются для моделирования периодических сигналов (рис. 7).

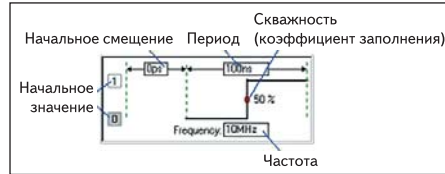


Рис. 7. Настраиваемые параметры периодического сигнала, задаваемого стимулятором типа Clock

Для настройки доступны все основные параметры такого сигнала: начальное значение, частота (или период повторения), скважность (коэффициент заполнения), начальное смещение (задержка).

С помощью стимулятора **Formula** можно описать сигнал произвольной формы как последовательность переключений (событий). На самом деле формулы как математического понятия здесь нет, есть просто текстовая строка в формате:

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

Из этой «формулы» видно, что в простейшем случае стимулятор **Formula** превращается в стимулятор **Value**. В общем случае — это цепочка событий (транзакций) «значение – время» произвольной длины. К тому же, созданную последовательность вы можете заставить повторяться (-r от слова **repeat** — повторение) с желаемым периодом (**period**). В частном случае легко задать обычный тактируемый сигнал, описываемый стимулом **Clock**. Другими словами **Formula** — это универсальный инструмент описания внешних воздействий.

Рассмотрим на практике, как работать с такими стимуляторами. В том же самом проекте **Lesson\_2** создадим еще один схемный файл и назовем его **clock\_formula**. В отличие от предыдущего эксперимента не станем даже задавать входные порты, пропустив в «мастере» эту процедуру.

В любом месте схемы нарисуем проводник (команда **Wire**) и назовем его **Clk**. Откомпилируем схему и, раскрыв рабочую библиотеку **Lesson\_2 library**, убедимся, что в ней появился откомпилированный модуль **clock\_formula**



Рис. 8. Окно просмотра проекта Design Browser

(**clock\_formula**). Щелкнем на нем ПКМ и исполним команду **Set as Top-Level**. Тем самым мы установили ему статус модуля верхнего уровня, и теперь он стал доступен программе моделирования.

Обратите внимание, в верхнем поле окна просмотра проекта **Design Browser** появилось (рис. 8) имя модуля верхнего уровня.

Добавим сигнал **Clk** в окно редактора сигналов **Waveform Editor**. Похожим образом мы действовали, исследуя стимулы **Value** и **Hotkey**.

Вызвав панель **Stimulators** (иконка ), активизируем стимулятор **Clock** и зададим параметры периодического сигнала, как показано на рис. 9.

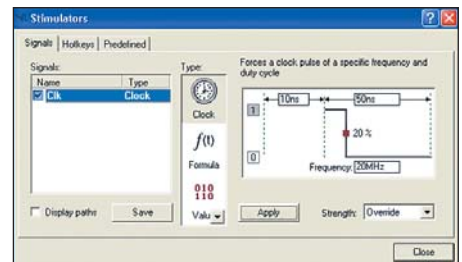


Рис. 9. Программируем периодический сигнал Clk с помощью стимула Clock

Мы специально поменяли все установленные по умолчанию значения, включая и коэффициент заполнения (в примере 20%). Промоделируем схему с шагом 25 ns и, включив режим измерения **Measurement Mode** (кнопка ) , проставим на временной диаграмме «размеры» всех ранее запрограммированных интервалов времени (рис. 10).

Обратите внимание, первые 10 ns (начальное смещение или задержка) сигнал **Clk** имел значение 'U', то есть не был инициализирован.

Попробуем воспроизвести в точности такую же форму временной диаграммы с помощью другого стимулятора — **Formula**. Нарисуем на той же схеме еще один проводник и назовем его **Form**. Проведя уже известную работу, запрограммируем сигнал **Form**, как показано на рис. 11.

Промоделировав новую схему, убедимся, что все было сделано правильно (рис. 12).

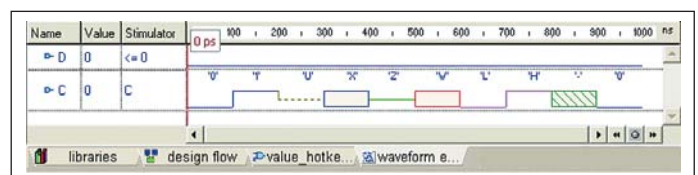


Рис. 6. Выводим на экран монитора все возможные значения сигнала С типа std\_logic

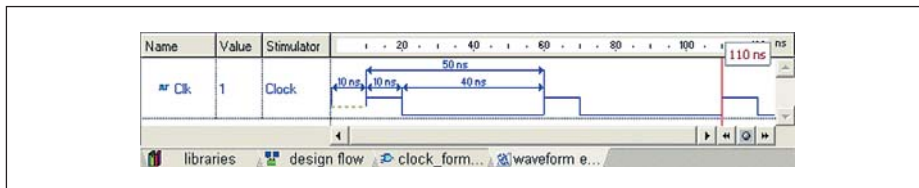


Рис. 10. С помощью измерения временных интервалов убеждаемся, что стимулятор Clock сформировал требуемую периодическую последовательность

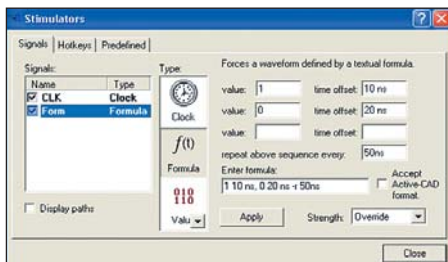


Рис. 11. Программируем периодический сигнал Form с помощью стимула Formula

Щелкнув правой кнопкой мыши в зоне временных диаграмм, исполним команду **Add Cursor**. Расставим курсоры так, как показано на рис. 12.

Это еще один способ проконтролировать интересующие нас временные интервалы. Заметим, что в отличие от некоторых других САПР, например **DesignLab 8.0** или **OrCAD 9.1**, среда **Active-HDL 7.1** не имеет ограничений на число курсоров.

Когда событий (переключений) не более трех, то их предпочтительнее вводить в специальные поля **value** и **time offset** (как в только что рассмотренном примере, показанном на рис. 11). По мере заполнения этих полей вы обнаружите, что копии событий дублируются и в поле **Enter formula**.

Если переключений более трех, вы можете просто дописать их в нижнее поле. Чтобы многократно повторить введенную последовательность, достаточно приписать в конце строки символ повторения (-) и указать период, например 50 ns.

## Предопределенные стимуляторы типа Predefined

Названный тип стимулятора удобен тем, что не требует программирования. Достаточно сослаться на его имя. Допустим, нам нужен генератор импульсов с частотой 25 МГц,

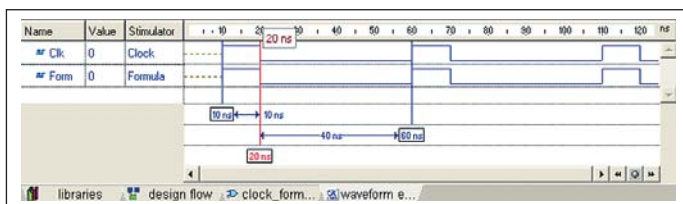


Рис. 12. С помощью дополнительных курсоров убеждаемся, что оба стимула (Clock и Formula) воспроизводят одинаковую форму сигналов

В списке поставляемых с системой предопределенных стимулов он находится под именем **Clock B2** (рис. 13). Указав это имя на закладке **Signals**, мы решаем задачу одним щелчком мыши.

Если подходящего стимулятора нет, его легко создать самостоятельно, внести в список предопределенных стимулов и в дальнейшем ссылаться на него, когда возникает необходимость. Обычно рассматриваемый прием эффективен для часто используемых сигналов с неизменной временной диаграммой. Такими сигналами могут быть сброс (**Reset**), установка (**Set**), синхронизация (**Clock**) и т. п.

Создадим предопределенные стимуляторы **Reset** и **Reset\_n** с временными диаграммами, описываемыми «формулами»: 0 0 ns, 1 30 ns, 0 50 ns и 1 0 ns, 0 30 ns, 1 50 ns соответственно. В первом случае сброс осуществляется высоким уровнем, во втором — низким. Мы уже умеем программировать стимулы типа **Formula**, поэтому никаких проблем возникнуть не должно.

Откройте на панели **Stimulators** закладку **Predefined** и нажмите кнопку **Add** (рис. 13). Введите имя **Reset** и в поле **Stimulator type** укажите тип стимула **Formula**. В поле **Enter formula** введите требуемый текст:

```
0 0ns, 1 30ns, 0 50ns
```

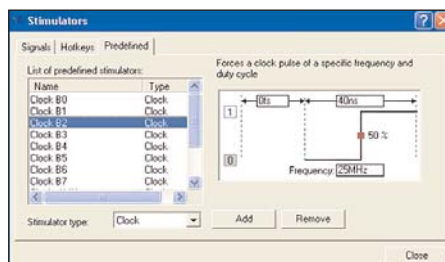


Рис. 13. Просматриваем список предопределенных стимуляторов

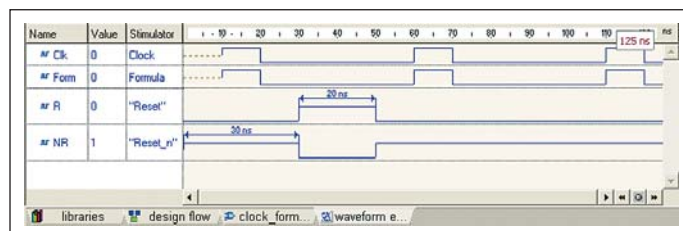


Рис. 14. Тестируем предопределенные стимуляторы Reset и Reset\_n

Переключитесь на закладку **Signals**, и в окне **Predefined** вы увидите созданный стимул.

Аналогичным образом создайте стимул **Reset\_n**. Протестируйте полученные стимулы так, как мы это делали ранее. Надеюсь, что предлагаемая работа не вызовет у вас особых затруднений.

Ожидаемые результаты показаны на рис. 14.

Заметим, что предопределенные стимуляторы сохраняются в системных регистрах, а не в волновых файлах, имеющих расширение \*.awf.

Поэтому при переносе проекта на другой компьютер могут возникнуть определенные проблемы: вы их просто не обнаружите в наличии. По этой причине увлекаться такими стимулами не следует.

## Описание шинных сигналов стимуляторами типа Counter и Formula

В принципе все стимуляторы, кроме **Clock**, позволяют программировать как одиночные, так и шинные сигналы. Значения шинных сигналов могут представляться как символьной литеральной строкой, например 1101, так и числами в двоичной (2#1101), восьмеричной (8#15), десятичной (10#13) или шестнадцатеричной (16#D) системе счисления. В данном примере описано одно и то же число 13 всеми возможными способами. Для периодических шинных сигналов предпочтение следует отдать стимулятору типа **Counter**, весьма мощному и эффективному средству. Одиночные сигналы он не поддерживает.

Создадим новый схемный файл **stim\_counter.bde**. Нарисуем в нем 4-разрядную шину A(3:0) и запрограммируем ее с помощью стимулятора **Counter** как двоичный суммирующий счетчик (рис. 15).

В левом верхнем поле **Count type** укажем тип счетчика **Binary** (двоичный), а в соседнем справа поле **Count Direction** — направление счета **Up** (суммирующий). Далее зададим начальное значение счетчика (поле **Starting value**), шаг изменения (**Modify by**) и время сохранения каждого состояния (**Count every**).

Наконец, надо задать систему счисления, в которой будет восприниматься стартовое значение (возможны двоичный, восьмеричный, десятичный и шестнадцатеричный форматы). Впрочем, описанный режим установлен по умолчанию, так что можно получить результат без всяких усилий (рис. 16).

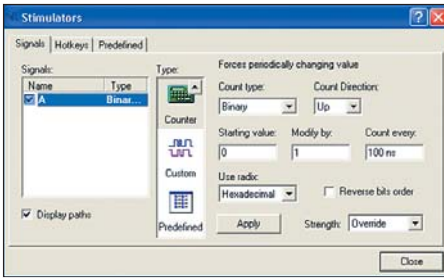


Рис. 15. Задаем для шины A(3:0) параметры двоичного суммирующего счетчика

Очевидно, что для получения вычитающего счетчика достаточно сменить в поле **Count Direction** значение **Up** (вверх) на **Down** (вниз). Причем сделать это можно непосредственно в ходе моделирования.

Для получения двоично-десятичного счетчика следует поступить так. Когда в пошаговом режиме моделирования в счетчике окажется число 9, надо на диалоговой панели **Stimulators** нажать кнопку **Apply** и «заставить» счетчик начать счет заново (по умолчанию — с нуля).

Кроме двоичного счетчика (**Binary**) вы можете имитировать работу счетчиков Грея (**Gray Code**), Джонсона (**Johnson Code**) или организовать циклический счет, гоняя по кругу '1' (**Circular One**) или '0' (**Circular Zero**) в одну или в противоположную ей сторону (счет в унитарном коде — кольцевой счетчик).

Для описания неперiodических шинных сигналов предпочтение следует отдать стимулятору типа **Formula**. Показанная на рис. 16 временная диаграмма шинного сигнала **B(3:0)** получена с помощью следующей строки:

```
0 0ns, 16#F 250ns, 10#13 500ns, 8#16 750ns, 0101 1000ns, 2#0111 1250ns, 2#1 1500ns
```

Она специально «запутана», чтобы показать все возможные форматы записи значений шинного сигнала.

### Стимуляторы типа Random

Стимулятор типа **Random** (случайный) основывается на использовании генератора случайных чисел, который возвращает значения целых чисел, распределенные согласно стандартным вероятностным функциям. В поле **Distribution** (рис. 17) можно установить желаемый закон распределения, в частности рав-

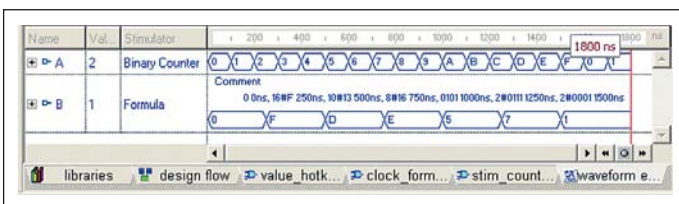


Рис. 16. Результаты моделирования шинного сигнала типа Counter на примере двоичного суммирующего счетчика

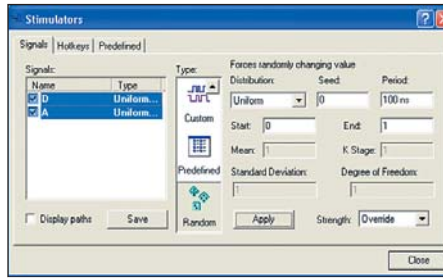


Рис. 17. Для программирования сигналов со случайными числовыми последовательностями используем стимулятор Random

номерный (**Uniform**), нормальный (**Normal**), экспоненциальный (**Exponential**) и некоторые другие (всего восемь распределений).

По умолчанию будут генерироваться псевдослучайные числа с равномерным распределением. Для настройки данного генератора доступно несколько полей.

В поле **Seed** задается начальное число, порождающее случайную числовую последовательность. В соседнем справа поле **Period** устанавливается временной интервал, с которым меняются числа.

В полях **Start** и **End** задается диапазон изменения случайных чисел (по умолчанию — отрезок 0–1).

Создадим схемный файл **Random.bde**, разместим в нем один шинный сигнал **D(3:0)** и два одиночных сигнала **A** и **B**. Подключим к первым двум стимулятор **Random** с параметрами, заданными умолчанием, а для сигнала **B** зададим диапазон 2–3.

Выполним моделирование на интервале 2000 ns (рис. 18).

Временная диаграмма шинного сигнала **D(3:0)** показывает последовательность целых чисел: 10001100111111100010. На этом интервале число 0 выпало 9 раз, а 1 — 11 раз. Даже на такой небольшой выборке уже проявляется основное свойство данного распределения — равная вероятность появления случайных чисел.

А вот эпюра одиночного сигнала **A** выглядит несколько неожиданно: вместо битовой последовательности нулей и единиц мы наблюдаем хаотичную смену двух значений сигнала — 'U' и 'X'.

Впрочем, все легко объясняется: это первые два значения цифрового сигнала переисчисляемого типа **std\_logic**, который может принимать 9 значений: 'U', 'X', '0', '1', 'Z', 'W',

'L', 'H', '-'. Целому числу 0 соответствует значение цифрового сигнала 'U', а числу 1 — 'X'. Можно предположить, что числу 2 будет поставлено в соответствие значение '0', числу 3 — значение '1', числу 4 — значение 'Z' и т. д.

Наши догадки подтверждает эксперимент с одиночным сигналом **B** (рис. 18). С ним тоже ассоциирован стимулятор типа **Random** с равномерным законом распределения, но диапазон генерируемых чисел установлен иным — 2...3. Как видите, все получилось наилучшим образом: перед нами действительно битовая последовательность 10001100111111100010 двоичных чисел.

Для нормального закона распределения доступными становятся другие поля: **Mean** (среднее значение) и **Standard Deviation** (стандартное отклонение).

Заканчивая разговор о генераторах псевдослучайных чисел, заметим, что они широко применяются в процедурах тестового диагностирования цифровой аппаратуры и, в частности, в сигнатурных анализаторах.

### Логическая сила сигнала (Strength)

Основное назначение стимуляторов — генерировать значения сигналов на входных цепях цифровых устройств. Но этим область их применения не ограничивается. В принципе стимуляторы позволяют подменить любой сигнал, вырабатываемый в схеме, невзирая на его истинные значения, вычисляемые моделью. Такая необходимость возникает, например, при разработке проверочных векторов (**test vectors**) для процедур внутрисхемного тестирования печатных плат с помощью подпружиненных игольчатых контактов.

Допускается даже принудительное (**Override**) задание тестового сигнала на выходе вентиля, вырабатывающего противоположное значение сигнала. Неприятностей удастся избежать, если тестовые сигналы в реальных схемах подавать в течение короткого времени.

Понятно, что тестовый сигнал должен «побороть» выходной сигнал вентиля, то есть иметь силу большую, чем логический выход. По умолчанию стимулятору как раз и назначается такой уровень силы — **Override**. Вы без труда обнаружите это значение в раскрываемом списке **Strength** (сила) в правом нижнем углу диалоговой панели **Stimulators** (рис. 17).

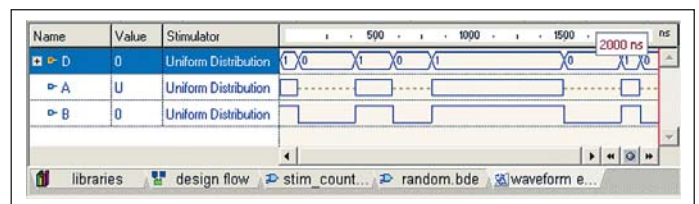


Рис. 18. Тестируем стимулятор Random, генерирующий псевдослучайные последовательности

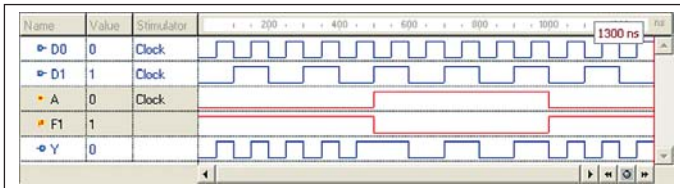


Рис. 19. Выходной сигнал F1 является инверсией входного сигнала A и автоматически вычисляется в процессе моделирования

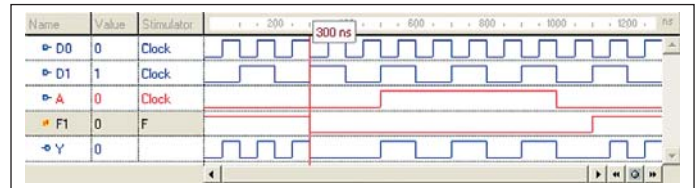


Рис. 20. Стимул, подключенный к внутренней цепи F1, преодолевает сигнал, вырабатываемый моделью в этом узле

Проведем простой эксперимент, подтверждающий только что сказанное. В окне просмотра проектов **Design Browser** щелкнем ПКМ на проекте **Lesson\_1** и сделаем его активным (команда **Set as Active Design**). Откроем файл с временными диаграммами **mux2\_schema.awf** и добавим в него сигнал **F1**, формируемый на выходе инвертора **U3** (см. урок 1, рис. 11).

Промоделируем схему еще раз, чтобы убедиться, что сигнал **F1** является инверсией входного сигнала **A** (на рис. 19 они выделены красным цветом).

А теперь «повесим» на сигнал **F1** стимулятор типа **Hotkey** (или любой другой) и повторим моделирование, управляя значением сигнала на проводнике **F1** с помощью клавиши **F** (рис. 20).

Пока значение, выдаваемое стимулом, повторяет истинное значение, вычисляемое моделью в узле **F1** (до момента времени 300 ns), реакция на выходе мультиплексора не искажается. Но потом мы нажимаем клавишу **F** и стимул принудительно (**Override**) задает в узле **F1** значение 0, перекрывая вычисляемый моделью сигнал. Очередной импульс с входа **D0** не попадает на выход **Y** мультиплексора, то есть оба его канала оказываются отключенными.

Повторим эксперимент еще раз, изменив уровень логической силы стимула, прикрепленного к узлу **F1**, на значение **Deposit** (хранение). В этом режиме «навязанное» стимулом значение отменяет текущее значение сигнала, вычисленное в имитационной модели. Но его действие будет распространяться только до ближайшего события в данном узле,

после чего стимул потеряет силу, если его не инициировать новым значением.

В третьем режиме **Drive** присоединенный стимул ассоциируется с дополнительным драйвером сигнала, действующего в данном узле, например в узле **F1**. Теперь стимул не в состоянии преодолеть сигнал, вычисленный моделью. Если они имеют разные значения, то в управляемом ими узле возникнет конфликт и результирующее состояние станет равным 'X'.

Практическое применение этот режим находит при моделировании двунаправленных сигналов и тристабильных шин.

У нас остался не рассмотренным только стимулятор типа **Custom** — заказной стимулятор. Он имеет особое назначение, и о нем мы поговорим на следующем уроке. ■