Посвящается моей семье, Сэнди, Джону и Холланд

Оглавление

Предисловие: об этой книге

Язык Verilog появился зимой 1983–1984 годов и первоначально был коммерческим продуктом, предназначенным для моделирования и верификации цифровой аппаратуры. С тех пор он несколько раз подвергался стандартизации в Институте инженеров электротехники и электроники (Institute of Electrical and Electronics Engineers – IEEE); наконец, этот процесс завершился стандартом IEEE Std 1364™-2005, известным также как Verilog-2005[1]. SystemVerilog – это набор расширений Verilog-2005, определенных в стандарте IEEE Std 1800™-2005. И хотя это действительно надстройка над Verilog-2005, слово «расширение» не передает всей мощи нового языка. Получившаяся в результате комбинация того и другого описана в стандарте IEEE Std 1800™-2009[2].

SystemVerilog позволяет разработчикам работать с моделями более высокого уровня абстракции, что отвечает сложности современных цифровых систем. SystemVerilog – это не язык описания аппаратуры, с которым работали ваши родители, когда типичная микросхема содержала несколько тысяч транзисторов, ПЛИС (FPGA)[3] еще только маячили на горизонте, а логический синтез пребывал в младенческой стадии. В современных подходах к проектированию аппаратуры проверка модели (верификация) не менее важна, чем ее создание и имитационное моделирование (симуляция). SystemVerilog предлагает конструкции, позволяющие лучше отразить инженерный замысел в моделях, программные абстракции, упрощающие разработку тестовых окружений, утверждения, обеспечивающие проверку поведения сложных систем, а также средства измерения функционального покрытия в процессе верификации.

Хорошо это или плохо, но язык стал настолько большим, что охватить его целиком в одной книге трудно. В этой книге мы даже не пытаемся это сделать! У такого большого языка есть преимущество – это единая среда, объединяющая разработчиков и верификаторов и охватывающая многие уровни абстракции, используемые при проектировании интегральных схем. Книга начинается с описания новых конструкций SystemVerilog; при этом она содержит объяснения, позволяющие без труда читать унаследованные модели. Даже если вы никогда не сталкивались с языком Verilog, вам не придется изучать его перед освоением SystemVerilog; в книге есть все необходимое, чтобы начать работать с SystemVerilog.

---

[1] Предыдущими стандартами языка Verilog являются IEEE Std 1364™-1995 (так называемый Verilog-95) и IEEE Std 1364™-2001 (так называемый Verilog-2001). – *Здесь и далее прим. перев.*

[2] В феврале 2018 года стандарт языка SystemVerilog был обновлен – IEEE Std 1800™-2017.

[3] ПЛИС – программируемая логическая интегральная схема. Здесь и далее под ПЛИС понимается программируемая пользователем вентильная матрица (Field-Programmable Gate Array – FPGA).

Предполагается, что у читателя есть базовая подготовка в области схемотехники и программирования. Материал по языку дается вместе с материалом по логическому проектированию, так что книга может использоваться в качестве учебного пособия для курсов цифровой схемотехники и архитектуры компьютеров. Абстракции языка соответствуют абстракциям, используемым при проектировании, поэтому темы схемотехники и языка переплетаются. Книга ориентирована на следующие группы читателей:

- студентов, проходящих вводный курс цифровой схемотехники, на котором также преподается SystemVerilog;
- разработчиков, знакомых с Verilog или VHDL и желающих освежить свои навыки или нуждающихся в кратком справочнике по SystemVerilog;
- студентов, слушающих курсы по разработке СБИС/ПЛИС, затрагивающие дополнительные темы, в т. ч. вопросы верификации.

По сравнению с предыдущими книгами автора, написанными в соавторстве с Филипом Мурби (Philip Moorby)[4] («The Verilog Hardware Description Language, Fifth Edition», издательство Springer), в которых подробно описывался язык Verilog, эта книга в большей степени посвящена проектированию и верификации сложных цифровых систем.

Вы не найдете в книге полного описания SystemVerilog. Все верно – язык огромен! Задача книги – познакомить читателя с широким спектром возможностей языка; она дополняет вводные и продвинутые курсы по проектированию и верификации аппаратуры и закладывает фундамент для дальнейшего изучения. В книге имеются части, посвященные схемотехнике, в которых язык затрагивается лишь мимоходом (примером может служить глава 6). Такие части будут интересны студентам и преподавателям университетов.

В конце некоторых глав имеются задачи. За решениями, по крайней мере, некоторых из них можете обращаться ко мне по адресу dthomas@cmu.edu или dthomas1611@gmail.com. Возможно, я отвечу не сразу, но я постараюсь ответить. Если же вы студент, решайте самостоятельно. Только практикуясь, можно научиться!

Чтобы получить текст примеров для использования в учебных слайдах, обращайтесь ко мне по адресам, указанным выше.

Напоследок отметим, что в этом переработанном издании был обновлен материал о спецификации несущественных ситуаций в комбинационных схемах.

Получайте удовольствие от проектирования хороших систем!

– Дон Томас –

[4] Филип Мурби – один из создателей языка Verilog и разработчик первого Verilog-симулятора.

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте **www.dmkpress.com**, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

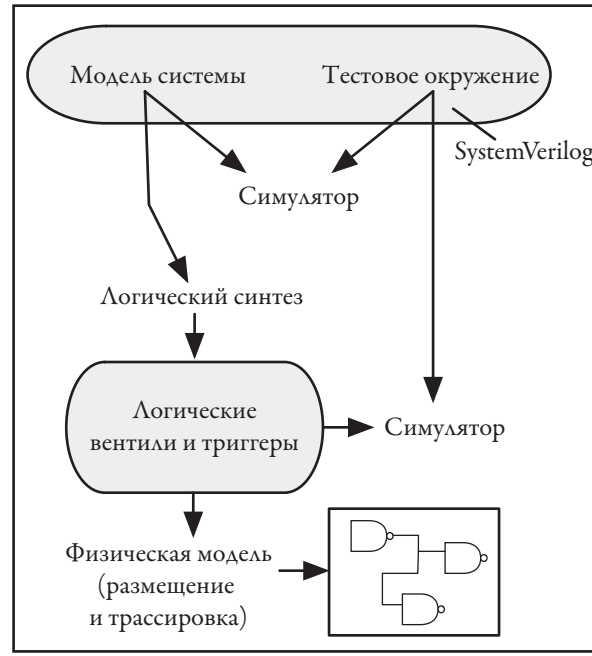
Пожалуйста, свяжитесь с нами по адресу электронной почты **dmkpress@gmail.com** со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Контекст: проектирование на уровне регистровых передач

Сейчас производятся цифровые системы с миллиардами транзисторов на кристалле. Любитель, конечно, может в качестве спецификации (для реализации на макетной плате) нарисовать несколько логических вентилей и соединить их проводами, но для коммерческих проектов это древняя история.

Современные системы специфицируются на языках описания аппаратуры, таких как SystemVerilog. Языки этого типа позволяют описывать аппаратуру в терминах ее функциональности. Система автоматизированного проектирования (САПР) получает описание аппаратуры на входе и предоставляет средства для автоматизированного (иногда автоматического) формирования детальной модели[5].



На рисунке справа показана упрощенная схема проектирования на уровне регистровых передач[6]. Модель, задающая функциональность системы, представляется на языке описания аппаратуры, например SystemVerilog (сверху). Для

[5] Имеется в виду модель, используемая для производства микросхем: топология интегральной схемы или прошивка ПЛИС.

[6] Подробно модели уровня регистровых передач рассматриваются в частях I и II (главах 2–6).

того чтобы проверить функциональную корректность модели, симулятор исполняет ее вместе с тестовым окружением. Тестовое окружение – это программа на языке описания аппаратуры, предназначенная для тестирования разрабатываемой системы.

Затем по модели синтезируется логическая схема. САПР автоматически определяет триггеры и вентили, необходимые для реализации модели, представленной на языке описания аппаратуры. Как правило, инструменты логического синтеза оптимизируют схему с учетом заданных ограничений, таких как занимаемая площадь, задержка распространения сигналов и потребляемая мощность. После этого начинается процесс физического проектирования, ориентированный на создание интегральной схемы или ПЛИС. Процесс называется *физическим*, поскольку его результатом является физическая реализация системы[7]. В случае интегральной схемы необходимо задать место расположения каждого логического вентиля и соединить их между собой (эта процедура называется размещением и трассировкой). Все это показано на рисунке. Схожие действия осуществляются, если конечной целью является ПЛИС.

Несмотря на то что это сильно упрощенный взгляд на процесс проектирования интегральных схем, общая суть понятна: в процессе задействуется множество инструментов, да и сам по себе процесс сложен. Без обеспечиваемой этими инструментами продуктивности было бы невозможно думать о создании систем, состоящих из миллиардов транзисторов! Отправной точкой всего процесса является язык описания аппаратуры, например SystemVerilog, – именно на нем описывается модель системы и тестовое окружение для ее верификации.

Проектирование цифровой системы сводится к использованию тех или иных абстракций, и разные части книги посвящены моделированию разных аспектов системы и соответствующим абстракциям.

- Сначала дается введение в языки описания аппаратуры и событийное моделирование.
- Затем, в первой части, описывается моделирование на уровне регистровых передач: комбинационные схемы, триггеры и конечные автоматы.
- Во второй части уровень проектирования повышается до вычислительных блоков – конечных автоматов с трактом данных. Мы называем их «аппаратными потоками».
- В третьей части рассматривается разработка тестового окружения, позволяющего верифицировать модель. Сюда же включены сведения об утверждениях и функциональном покрытии.
- В последней части обсуждаются детали, которым не нашлось места в других разделах книги.

Даже все эти части не охватывает язык целиком. С некоторыми деталями лучше знакомиться по справочному руководству.

---

[7] На самом деле результатом физического проектирования интегральной схемы является *топология* – описание (например, в формате GDSII) пространственно-геометрического расположения элементов схемы и соединений между ними. По такому описанию могут быть построены фотошаблоны, применяемые для производства микросхем.

Благодарности

Автор выражает благодарность всем студентам, которым пришлось иметь дело с ранними вариантами этой книги. Ваши полезные, порой болезненные замечания позволили улучшить объяснения и примеры. Профессор Билл Нейс (Bill Nace), с которым я имел удовольствие совместно преподавать в течение многих семестров, дал подробный отзыв о книге, доходя иногда до указания замеченных опечаток. Я благодарю также Габриэля Сомло (Gabriel Somlo) за корректуру и Дебру Острандер Виейра (Debra Ostrander Vieira) за дизайн обложки и подготовку к печати. Наконец, я признателен всем своим учителям, прошлым и нынешним, благодаря которым смог написать эту книгу.

Глава 1

Введение

Симулятором[8] называется программа, которая предсказывает, как состояние физической системы изменяется со *временем*. Симулятор погоды предсказывает погоду в будущий момент времени в зависимости от текущей погоды. Компьютерная игра SimCity™ – это симулятор, предсказывающий развитие города в зависимости от текущей ситуации и действий, например капиталовложений в строительство дорог и другой инфраструктуры. Написанная вами программа для моделирования скорости брошенного тела – тоже симулятор; она вычисляет новое значение скорости в зависимости от предыдущего, ускорения свободного падения, трения и времени, прошедшего с момента начала падения. Общим для всех этих симуляторов является тот факт, что они непосредственно моделируют время.

SystemVerilog – язык описания и симулятор электронной цифровой аппаратуры. Он позволяет *моделировать* цифровую систему, например соединенные между собой логические вентили, и сообщает, как в системе распространяются значения в зависимости от времени. Он помогает определить, правильно ли система реализует свою функциональность и обеспечивает ли заданную производительность.

Модель проектируемой системы представляется на специальном *языке моделирования*. В языке имеются средства описания элементов модели и средства, позволяющие их использовать для построения больших систем. Например, язык SystemVerilog позволяет моделировать такие элементы, как логические вентили. Основные аспекты SystemVerilog, благодаря которым на нем можно описывать цифровую аппаратуру, – моделирование времени, функциональности и соединений. Мы уже отметили, что время – фундаментальная концепция любого симулятора, а стало быть, и языка моделирования. Межсоединения позволяют связывать вентили проводами (wires)[9]: когда изменяется выход одного вентиля, новое значение распространяется по проводам и попадает на входы других вентилей. Так физически устроены цифровые схемы, и язык

---

[8] Термин *simulator* переводится как *система* (*имитационного*) *моделирования*. Мы будем использовать более короткое и привычное разработчикам аппаратуры слово *симулятор*.

[9] Речь, очевидно, идет не о физических проводах, а о логической абстракции – переменных специального вида.

призван это отражать. SystemVerilog позволяет выразить куда более сложные и абстрактные конструкции, чем логические вентили, однако мы начнем с моделей уровня вентилей, поскольку с их помощью можно проиллюстрировать базовые возможности языка.

Язык SystemVerilog отличается от традиционных языков программирования. Предполагается, что читатель знаком с каким-нибудь языком программирования; в этой главе мы постараемся научить вас думать в терминах цифровой аппаратуры и объясним, какие языковые средства необходимы для описания и тестирования моделей аппаратуры.

1.1. Приступая к работе

Основным структурным элементом в языке SystemVerilog является *модуль*. У каждого модуля имеется интерфейс – входные и выходные порты, через которые осуществляется взаимодействие с другими модулями, – а также описание его содержимого. Модуль представляет собой блок, который можно описать, либо задав его внутреннюю структуру (например, указав, из каких логических вентилей он состоит), либо определив его поведение – так же, как в программах (в этом случае фокус смещается на функциональность модуля, а не на его реализацию с помощью вентилей). Модули соединяются проводами, благодаря чему они могут взаимодействовать и образуют более крупные системы.

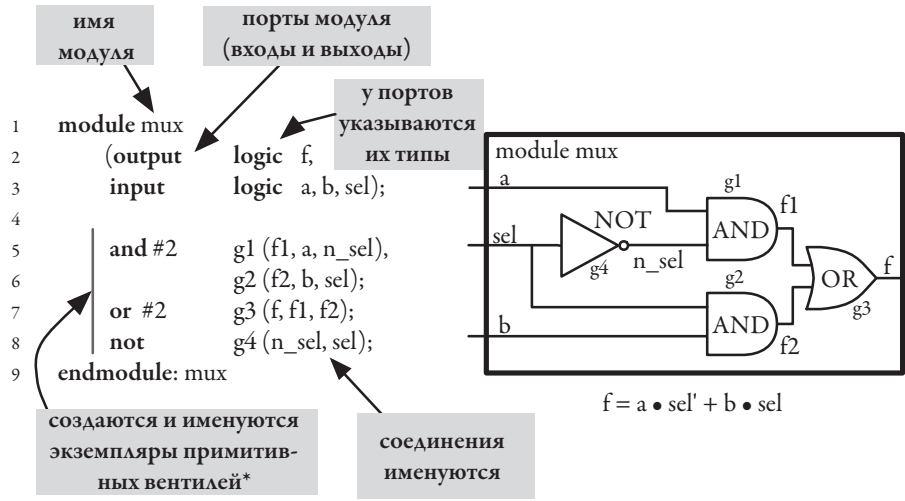
1.1.1. Структурное описание

Начнем с простой комбинационной схемы – двухвходного мультиплексора (2:1). В примере 1.1 приведена схема из логических вентилей, реализуемая ей булева функция и описание на языке SystemVerilog. Схема выбирает, какой из двух входов (`a` или `b`) определяет значение на выходе `f`. Если значение сигнала выбора (`sel`) равно TRUE, то на выход `f` подается значение `b`. Если же значение `sel` равно FALSE, то на выход `f` подается значение `a`.

В левой части рисунка показано *определение* SystemVerilog-модуля, соответствующего схеме справа. В данном случае модуль называется `mux`. Определение любого модуля начинается ключевым словом `module`, за которым следует имя модуля, а заканчивается ключевым словом `endmodule`. В строках 2 и 3 определения указаны имена и типы входных и выходных портов. Все входы и выходы имеют тип `logic`, который мы опишем ниже.

В строках 5 и 6 создаются *экземпляры*[10] двух вентилей AND. В строке 7 добавляется вентиль OR, а в строке 8 – вентиль NOT. Всем экземплярам вентилей даются имена (от `g1` до `g4`) в соответствии с тем, как они названы в схеме. Запись #2 у некоторых вентилей означает, что задержка распространения сигнала от входов до выхода составляет две единицы времени. Для вентиля NOT

[10] Создание экземпляра (instance) модуля также называют инстанцированием (instantiation).



* Примитивными называются вентили, реализующие простейшие булевы функции: NOT (НЕ, отрицание), AND (И, конъюнкция), NAND (И-НЕ, штрих Шеффера), OR (ИЛИ, дизъюнкция), NOR (ИЛИ-НЕ, стрелка Пирса), XOR (исключающее ИЛИ, сумма по модулю 2), XNOR (исключающее ИЛИ-НЕ, эквивалентность).

Пример 1.1. Модуль и соответствующая ему логическая схема

задержка не указана и, следовательно, равна 0. Имена в скобках в строках 5–8 обозначают соединения входов и выходов вентилей. Первое имя соответствует выходу, остальные – входам. Имена соединений и экземпляров вентилей указаны на рисунке, чтобы пояснить соответствие между логической схемой и эквивалентным описанием на SystemVerilog. Номера строк не являются частью описания, они включены только для удобства. Ключевые слова языка выделены полужирным шрифтом.

Несмотря на простоту, этот пример иллюстрирует несколько важных особенностей SystemVerilog.

- Модули – основные структурные элементы языка. В определении модуля описываются порты и внутренняя функциональность; из модулей можно конструировать более сложные системы. В примере показан простой мультиплексор, но в виде модуля можно описать и целый компьютер.
- Примитивные вентили – в языке предопределены такие вентили, как AND, NAND, OR, NOR, NOT и XOR. При создании экземпляра вентиля первое имя в скобках – выход, остальные – входы. У примитивных вентилей может быть несколько входов, они перечисляются в списке портов через запятую.
- Создание экземпляров – эту процедуру можно рассматривать как помещение нового элемента на макетную плату. В данном модуле созданы экземпляры четырех примитивных вентилей. В каждой строке описывается новый экземпляр вентиля определенного типа, и каждый

экземпляр добавляет в модуль новую логику. Система становится физически больше, потому что для реализации каждого вентиля необходимы транзисторы.

- Соединения – вентили соединены проводами с другими вентилями и с портами содержащего их модуля. В данном случае соединения названы `a`, `b`, `sel`, `n_sel`, `f1`, `f2` и `f`. Тем самым моделируются электрические соединения между вентилями (модулями).
- Сокрытие информации – экземпляр модуля можно создать в других модулях. При этом внутреннее устройство модуля может быть неизвестно, известны лишь имена и типы портов. Потенциально сложная внутренняя структура и имена внутренних сигналов скрыты от пользователя модуля.

1.1.2. Как интерпретируется описание модуля

У человека, знакомого с языками программирования, который никогда не видел языка описания аппаратуры, сразу возникает вопрос: «Как этот модуль исполняется?» Здесь нет ни привычных циклов `for`, ни операторов `if`. Программист, пишущий на C, спросит, где функция main. Для ответа на эти вопросы нужно рассмотреть несколько моментов.

Во-первых, при соединении вентилей (а также модулей, как мы скоро увидим) мы делаем только одно – задаем именованные соединения компонентов. Создание экземпляра – это добавление в модуль еще одного компонента. Значения передаются по соединениям с выхода одного вентиля на вход другого. Определять компоненты и соединения между ними можно в любом порядке. Таким образом, оба варианта модуля `mux`, показанные в примере 1.2, определяют ту же самую логику, что и модуль из примера 1.1.

```
1 module mux
2   (output logic f,
3    input logic a, b, sel);
4
5    and #2 g1 (f1, a, n_sel),
6               g2 (f2, b, sel);
7    not       g4 (n_sel, sel);
8    or #2   g3 (f, f1, f2);
9 endmodule: mux
```

```
1 module mux
2   (output logic f,
3    input logic a, b, sel);
4
5    or #2   g3 (f, f1, f2);
6    not        g4 (n_sel, sel);
7    and #2 g1 (f1, a, n_sel),
8               g2 (f2, b, sel);
9 endmodule: mux
```

Пример 1.2. Два эквивалентных модуля

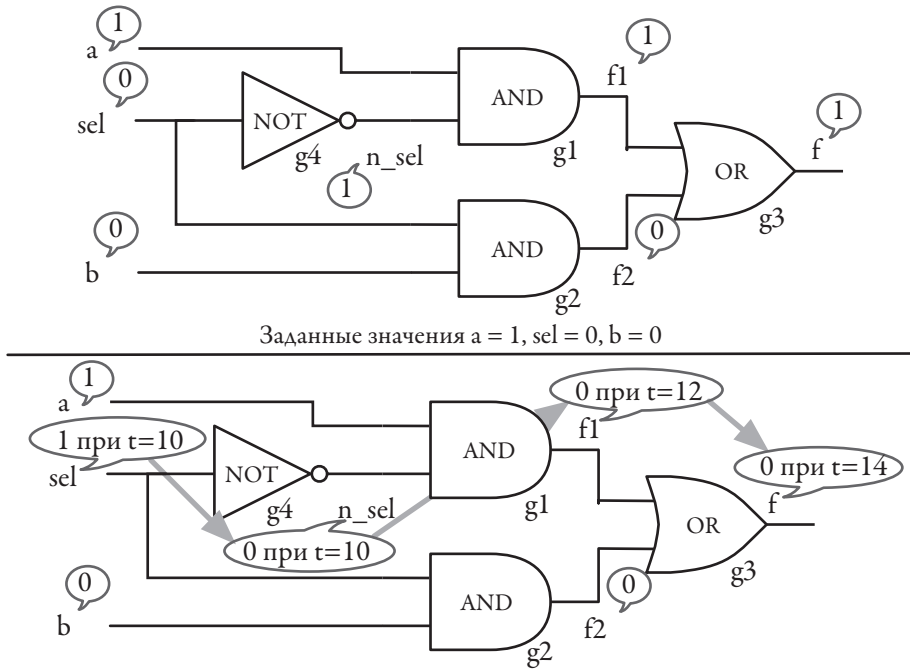


Рис. 1.1. Распространение значений во времени

Во-вторых, нужно понимать, что модуль является исполняемым, но *модель исполнения* не такая, к какой мы привыкли в языках программирования. В SystemVerilog при изменении значения входа вентиля симулятор перевычисляет значение его выхода. Если значение изменилось, симулятор распространяет это изменение на соединенные с ним вентили, возможно, с задержкой во времени. Исполняя модуль таким способом, симулятор моделирует распространение электрических сигналов между соединенными между собой компонентами. Это модель исполнения на уровне вентилей.

Поток значений от входа вентиля к его выходу, затем на вход следующего вентиля и т. д. не описывается вызовами функций, как можно было бы подумать, т. е. строки 5–8 в примерах выше не вызовы функций. Еще раз повторим, что эти строки описывают экземпляры вентилей и соединения между ними. Когда значение входа вентиля изменяется, симулятор вычисляет значение выхода и смотрит, изменилось ли оно. Если да, измененное значение распространяется по соединениям на входы других вентилей.

Рассмотрим, как значения распространяются в логической схеме на рис. 1.1. Предположим, что в течение длительного времени входы схемы не менялись и имеют следующие значения: `a=1`, `b=0` и `sel=0`. Легко видеть, что `n_sel=1`, `f1=1` и, следовательно, `f=1`, как показано на верхней диаграмме. Если в момент 10 значение `sel` станет 1, некоторые значения в схеме изменятся, как показано