8. Глава 8. Запоминающие элементы логических устройств

ЗЭ ЛУ могут быть построены на основе любой из ранее описанных логик: ТТЛ, ЭСЛ или К-МОП. Они принадлежат к последовательностным логическим устройствам (ПЛУ) или автоматам с памятью.

8.1. Описание функционирования ЗЭ ЛУ (спецификация простейшего ЗЭ ЛУ)

В логическом элементе (ЛЭ) "инвертор" выход сразу же реагирует на изменение сигнала на входе. ЗЭ должен вести себя не так. Если на вход ЗЭ подали сигнал установки "1", а затем перешли к стадии хранения информации в ЗЭ (то есть туда же подали сигнал перехода к стадии хранения), то на выходе единичная информация должна поддерживаться сколь угодно долго, несмотря на то, что на вход подается комбинация хранения, в отличие от ЛЭ ЛУ.

ЗЭ изображен на рис. 8.1 ($Y1$ — вход установки или запоминания на выходе "1", $Y0$ — вход запоминания "0").

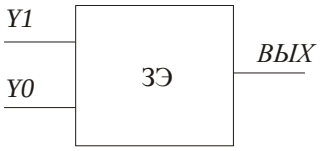

Рис. 8.1. Запоминающий элемент

Временные диаграммы функционирования стандартного ЗЭ ЛУ представлены на рис.8.2.

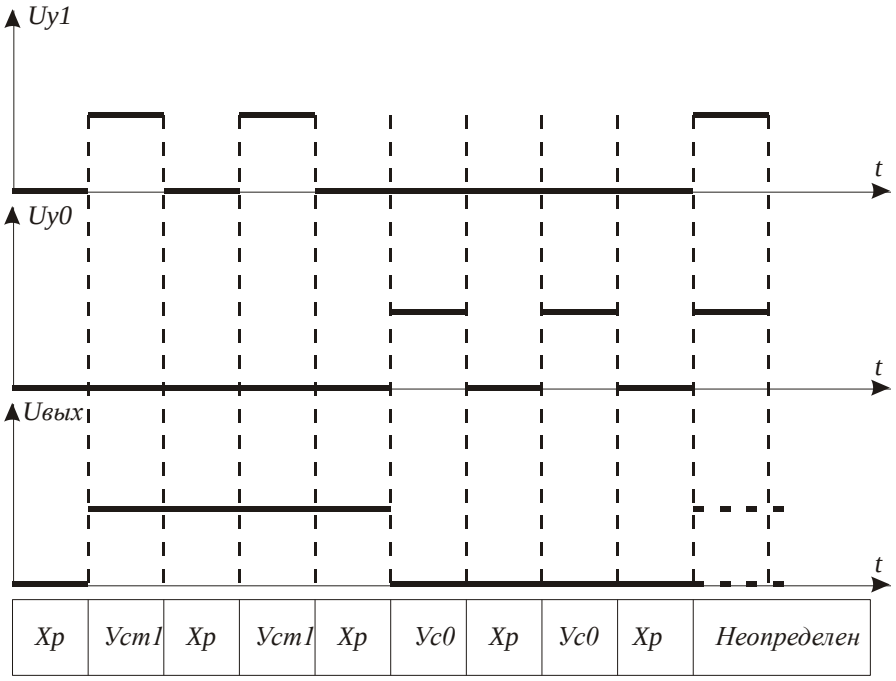

Рис. 8.2. Временные диаграммы ЗЭ

Здесь "*Хр*" — стадия хранения, "*Ус*0" — стадия установки "0", "*Ус*1" — стадия установки "1".

При неопределенности ЗЭ может растеряться, как и мы, когда нас дергают в разные стороны.

Электромеханическим аналогом работы ЗЭ является работа выключателя.

8.2. **Внутренняя структура ЗЭ**

Ответим на **вопрос**: за счет какого схемного усовершенствования возникает стабильное хранение "1" или "0"?

Для выключателя стабильность состояний задается внутренней связью ("пружинкой").

Для ЗЭ ЛУ стабильность состояний хранения "1" или "0" задается обратной связью (ОС).

Рассмотрим следующую схему с ОС (рис.8.3).

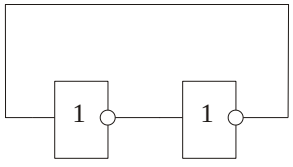

Рис. 8.3. Бистабильная ячейка из двух инверторов

Бистабильная ячейка имеет два устойчивых состояния.

Докажем это. Соответствующая электрическая схема выглядит так, как показано на рис. 8.4.

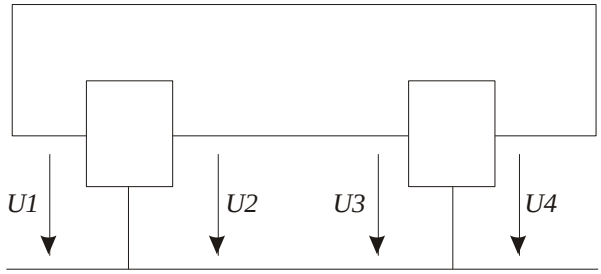

Рис. 8.4. Электрическая схема бистабильной ячейки

Здесь $U2=U3$ и $U4=U1$, этим равенствам соответствуют следующие точки пересечения ХПН инверторов — рис. 8.5.

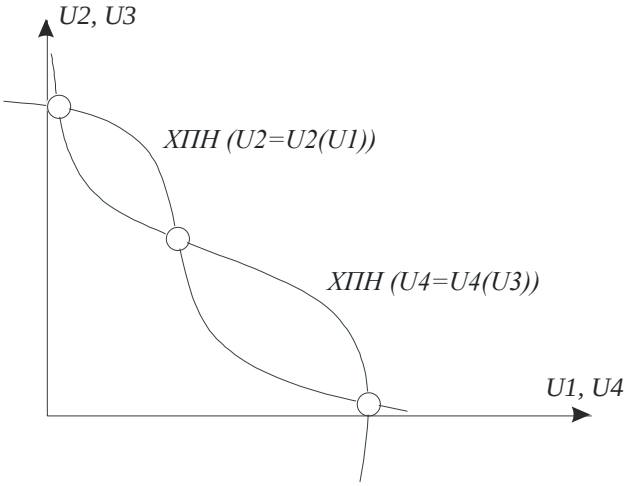

Рис. 8.5. Характеристика передачи напряжения бистабильной ячейки

Получили три точки пересечения или равновесия (средняя — неустойчивая, крайние — устойчивые), *но* пока имеем входы только для сигналов ОС, поэтому это еще не ЗЭ, так как нет входов установки "0" и "1".

8.2.1. **RS-триггер на "ИЛИ-НЕ"**

Объединим в едином ЗЭ входы установки "0" и "1" и его внутренние сигналы (сигналы ОС).

Если использовать элементы "ИЛИ-НЕ", то имеем рис.8.6.

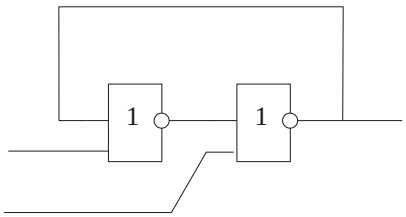

Рис. 8.6. RS-триггер

Для превращения этой схемы в схему бистабильной ячейки, хранящей информацию, на дополнительные входы нужно подать "нули", так как при этом из элементов "ИЛИ-НЕ" получаются инверторы. Поэтому комбинацией хранения в данном случае (случае с "ИЛИ-НЕ") являются два "нуля".

Обычно эту схему рисуют симметрично (на рис. 8.7 представлен временной, событийный процесс переключения триггера из состояния "0" в состояние "1" за время от $t0$ до $t4$, он связан с наличием задержек "ИЛИ-НЕ").

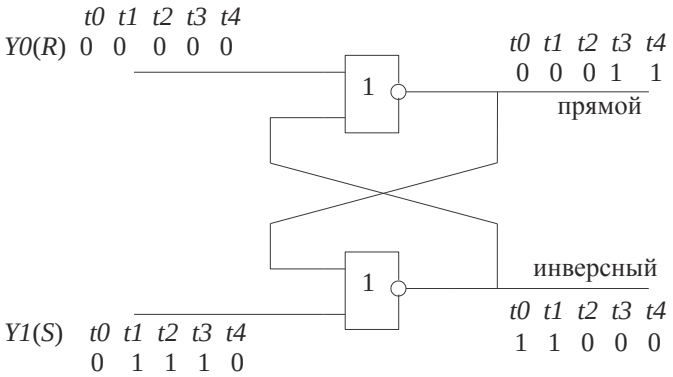

Рис. 8.7. RS-триггер с временным событийным процессом переключения

Сначала выберем прямой и инверсный выходы ЗЭ, так как по сигналу на прямом выходе судят о состоянии ЗЭ: единичное оно или нулевое. Пусть верхний выход схемы будет прямым выходом, а нижний — инверсным.

ЗЭ имеет две стадии работы:

- спокойная стадия или стадия хранения на выходе 0 или 1;
- стадия переключения:

  11. из состояния хранения "0" в состояние хранения "1";

  12. из состояния хранения "1" в состояние хранения "0" .

Рассмотрим комбинации сигналов, которые необходимо подавать на входы ЗЭ (S и R) для обеспечения работы на этих двух стадиях.

Для ЗЭ на "ИЛИ-НЕ" эти комбинации выглядят следующим образом:

- хранения: R — 0, S — 0;
- переключения из состояния хранения "0" в состояние хранения "1": R — 0, S — 1;
- переключения из состояния хранения "1" в состояние хранения "0": R — 1, S — 0.

Определим входы ЗЭ на "ИЛИ-НЕ": один вход назовем входом установки "1" ($Y1$), при подаче на него единичного сигнала, ЗЭ переходит в единичное состояние; другой вход назовем входом установки "0" ($Y0$), при подаче на него единичного сигнала, ЗЭ переходит в нулевое состояние.

Обозначение ЗЭ по ГОСТ представлено на рис. 8.8, здесь S (set) — установка "1", R (reset) — сброс "1", T — триггер:

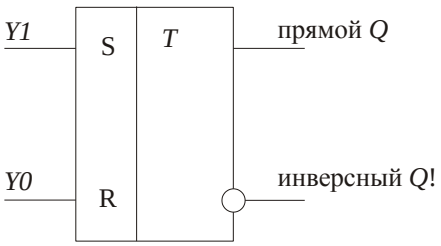

Рис. 8.8. УГО запоминающего элемента.

Этот ЗЭ называют RS-триггером

8.2.2. **RS-триггер на "И-НЕ"**

RS-триггер может быть построен на элементах "И-НЕ", тогда логическая схема выглядит так, как показано на рис. 8.9.

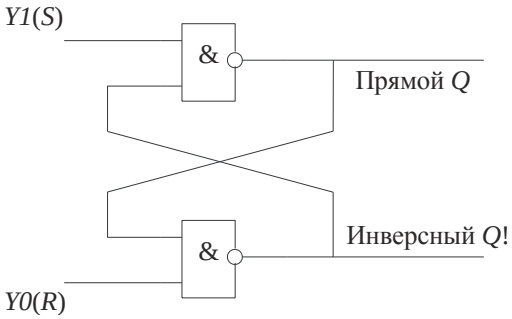

Рис. 8.9. Триггер на элементах "И-НЕ"

Комбинации сигналов необходимые для обеспечения работы:

- хранения: R — 1, S — 1;
- переключения из состояния хранения "0" в состояние хранения "1": R — 1, S — 0;
- переключения из состояния хранения "1" в состояние хранения "0": R — 0, S — 1.

Тот факт, что на установку и сброс подается "0" (нулевой) сигнал (а не единичный) в ГОСТ отображается инверсией (кружком) на входе.

8.3.**D-триггеры**

Для понимания использования RS-триггера рассмотрим часть схемы накапливающего или комбинационного сумматора последовательного действия. Здесь нужно помнить предыдущий перенос P_{K-1} при присутствующем на выходе сумматора следующем переносе P_K.

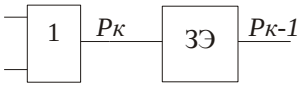

Рис. 8.10. Часть схемы накапливающего сумматора

Вопрос: как подключить вместо ЗЭ RS-триггер на "ИЛИ-НЕ"?

Решение

Заменим ЗЭ RS-триггером на "ИЛИ-НЕ" с инвертором на входе R. Входом такого ЗЭ будут объединенные входы инвертора и входа S триггера. Если на вход (P_K) такого ЗЭ придет "1", то на выходе (P_{K-1}) будем иметь "1",

если на вход ($Р_к$) такого ЗЭ придет "0", то на выходе ($Р_к-1$) будем иметь "0".

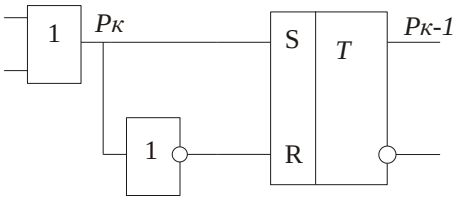

Рис. 8.11. Подключение RS-триггера в качестве ЗЭ

В данном случае триггер не работает как ЗЭ. Он работает как повторитель.

Для одновременного хранения предыдущего и следующего переносов к схеме через пару элементов "И" подключают синхронизирующие импульсы (СИ), которые разрешают считывание с элемента "ИЛИ" только в определенное время.

Эту пару ЛЭ "И" называют: симофором или защелкой.

На рис. 8.12 представлена схема, в состав которой входят инвертор, пара ЛЭ "И" и RS-триггер на "ИЛИ-НЕ".

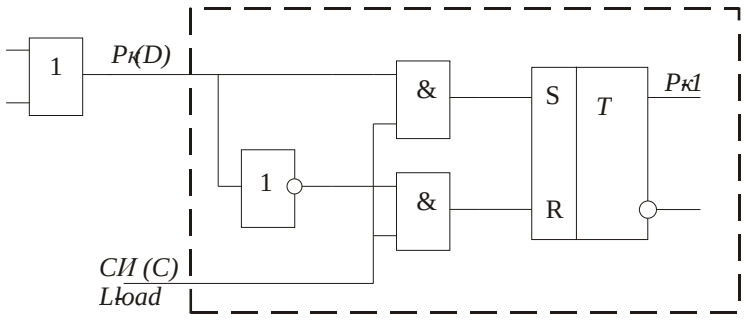

Рис. 8.12. Схема с защелкой

Обведенный участок схемы называется D-триггером (рис.8.13).

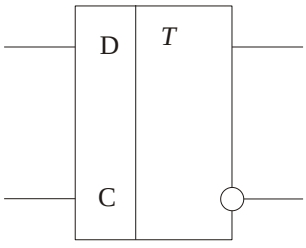

Рис. 8.13. УГО D-триггера

Схема D-триггера в базисе "И-НЕ" (реализованная на элементах "И-НЕ") показана на рис. 8.14.

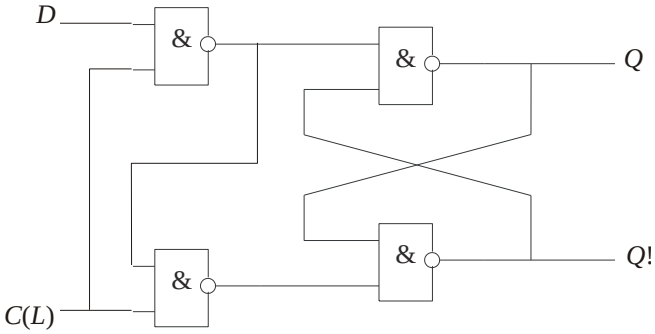

Рис. 8.14. Схема D-триггера в базисе "И-НЕ"

8.3.1. **DV-триггер**

Рассмотрим DV-триггер (рис. 8.15, 8.16). Здесь V-вход дополнительного управления (если V = 0, то входы отключены; если V = 1, то DV-триггер ведет себя как D-триггер)

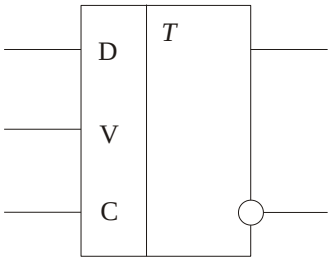

Рис. 8.15. УГО DV-триггера

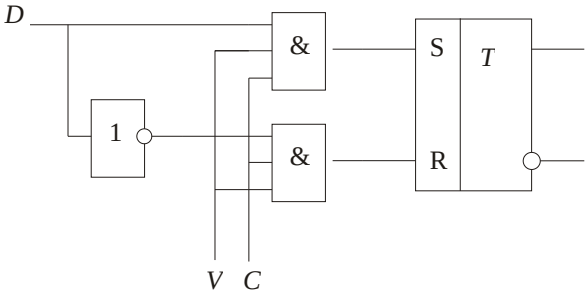

Рис. 8.16. Схема DV-триггера

8.3.2. **Двухтактный (двухфазный) D-триггер**

Если необходимо одновременно хранить два вида информации (например: предыдущий перенос и новое значение переноса при сложении), то

используют двухтактный или двухфазный D-триггер, состоящий из двух D-триггеров (рис. 8.17, 8.18).

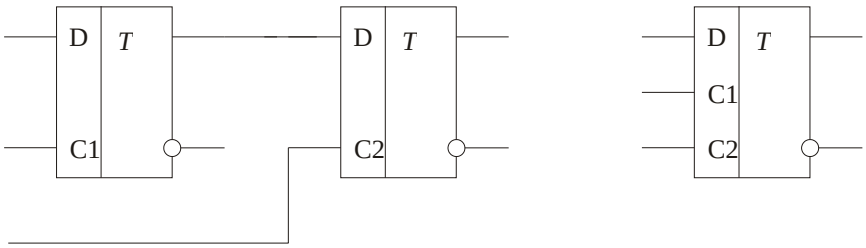

Рис. 8.17. Двухтактный (двухфазный) D-триггер

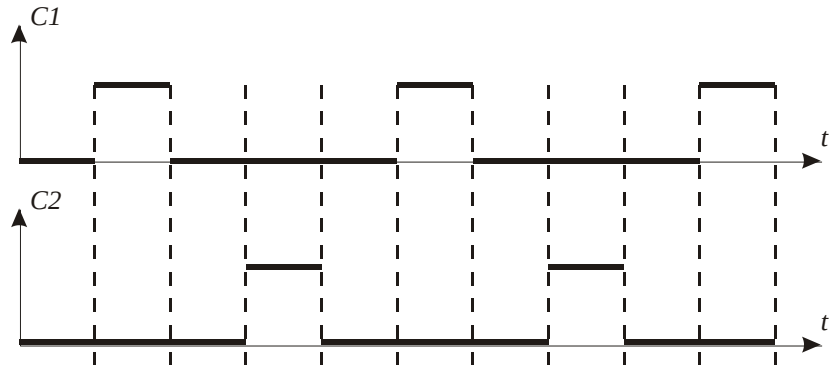

Рис. 8.18. Временная диаграмма, тактовые импульсы C1 и C2 сдвинуты друг относительно друга во времени

8.4. **Начальные сведения об описании ЛУ на языке VHDL**

Название языка VHDL расшифровывается как: VHSIC (Very High-Speed Integrated Circuits) HDL (Hardware Description Languages) (Язык описания аппаратуры очень высокоскоростных интегральных схем).

В данной главе дается пояснение к проектам RS-триггера на VHDL, представленным далее в лабораторной работе. В этой работе рассматриваются два варианта описания простейших логических устройств с использованием языков программирования (в нашем случае VHDL).

Первый вариант — создание схемы, когда разработка ведется, начиная с мельчайших элементов схемы (компонентов), которые в дальнейшем

соединяются в схему более высокого уровня иерархии (*структурный подход*).

Второй вариант — описание поведения устройства в целом в зависимости от значений сигналов на его входах (*поведенческий подход*).

Программа (проект) на языке VHDL состоит из нескольких блоков.

- Для подключения библиотечных функций и описания типов переменных языка используются операторы:

  LIBRARY <*имя библиотеки*>;

  USE <*имя модуля в библиотеке*>;

- Для описания входов и выходов устройства применяется блок:

  ENTITY <*имя проекта*> … END <*имя проекта*>;

- Для описания модели функционирования устройства во времени для различных комбинаций значений входных сигналов используют блок:

  ARCHITECTURE <*имя модели поведения*> OF <*имя проекта*> IS BEGIN … END <*имя модели поведения*>;

Модели могут быть двух типов: поведенческие и структурные, что соответствует двум подходам к описанию схем, описанным ранее.

- Следующий блок используется в случае структурного подхода и устанавливает соответствие компонента и реального проекта, полностью описывающего этот компонент:

  CONFIGURATION <*имя конфигурации*> OF <*имя проекта*> IS … END <*имя конфигурации*>;

Основными операторами языка являются:

- оператор присваивания:

  c <= a;

- оператор сравнения:

  IF … THEN … ELSE …END;

- оператор цикла:

Метка: FOR <*переменная цикла*> IN <*порядок*> LOOP … END LOOP *метка*.

Могут быть использованы следующие логические операции: AND, OR, NOT.

Операторы языка VHDL отделяются друг от друга точкой с запятой.

Кроме константы и переменной в VHDL введено понятие *сигнал*, ему соответствует соединительный провод в реальном устройстве. Сигнал в VHDL можно задерживать на заданное время так же, как реальный сигнал.

Для возможности описания параллельной во времени работы отдельных узлов сложного ЛУ в VHDL введен оператор process: внутри него команды на VHDL выполняются последовательно, если его нет, то все команды выполнятся параллельно (одновременно).

Более полное описание языка VHDL представлено в *приложении 1*.

В дальнейшем, при рассмотрении сложных устройств ЦТ (например, процессоров), изучение VHDL будет продолжено.

8.4.1. **Синтезируемость кода на языке VHDL**

Под синтезируемостью VHDL-кода понимается возможность перевода VHDL-проекта в схему, состоящую из логических элементов.

Для перехода к синтезируемости кода на VHDL рассмотрим основные этапы, которые включает проектирование с использованием языков описания аппаратуры и реализации их на ПЛИС:

1. Создание алгоритмического HDL-описания проекта.
2. Моделирование проекта (функциональное тестирование).
3. Автоматический синтез логической схемы.
4. Физическое проектирование
5. Моделирование логической схемы вентильного уровня с учетом задержек сигналов в проводах (временная симуляция).

Современные системы автоматического проектирования состоят из нескольких подсистем, автоматизирующих выполнение этапов проектирования цифровых систем.

- Редакторы ввода проектов.
  - Текстовые редакторы — предназначены для создания поведенческого или иерархического функционально-структурного описания проекта на HDL.
  - Графические редакторы — для ввода проектов в виде иерархических блок-схем, логических схем, автоматов состояний. Некоторые САПР автоматически генерируют HDL-описание по графическому представлению.
- Симуляторы — служат для тестирования проектов на соответствие спецификации.
- Средства автоматического синтеза (синтезаторы) — переводят поведенческое описание проекта в логическую схему, которая может быть представлена RTL или вентильным уровнем в технологическом базисе выбранной микросхемы.

- Register transfer level (RTL) — уровень описания, в котором поведение проекта явно описано в терминах передачи данных между запоминающими элементами и комбинационной логикой, которая может представлять любую вычислительную или арифметико-логическую схему.
- Вентильный уровень (Gate level) — уровень логических ячеек (из них состоит ПЛИС) может конфигурироваться из набора логических элементов (вентили и более сложные по функциональности ЛЭ), данный набор определяет базис проекта, или библиотеку, которую предоставляет производитель данной микросхемы. В соответствии с

этой библиотекой синтезатор строит логическую схему вентильного уровня по поведенческому описанию проекта.

- Средства размещения и трассировки — с их помощью выполняется размещение логической схемы проекта на кристалле, назначение выводов микросхемы входам-выходам, а также создается файл временных задержек для тестирования.
- Программаторы — средства программирования или конфигурирования ПЛИС.

Синтезируемое подмножество языков HDL

На этапе алгоритмического описания проект можно создавать не только на HDL, но и на других языках программирования (например, C, C++, Паскаль), а затем переводить это описание на язык HDL. Необходимо выполнять перевод, так как синтезаторы принимают на вход только HDL-описания, и не все конструкции, поддерживаемые данными языками, могут быть представлены синтезируемым HDL-кодом.

Существуют стандартные правила поддерживаемых HDL-конструкций для всех синтезаторов. В свою очередь производители программ синтеза могут включать свои правила.

Основные синтезируемые конструкции языка VHDL

Поддерживаемые (синтезируемые) конструкции

- Описание объектов: entity, architecture, package, function и procedure.
- Все библиотеки IEEE, включая:

  std_logic_1164

  std_logic_unsigned

  std_logic_signed

  std_logic_arith

  numeric_std and numeric_bit

  standard library package (std);

- Описание портов: in, out, inout, buffer.
- Описание связей, констант, переменных: signal, constant, variable.
- Типы сигналов и переменных: integer, bit, Boolean, std_logic, std_ulogic.
- Все операции (-, -, *, /, **, mod, rem, abs, not, =, /=, <, <=, >, >=, and, or, nand, nor, xor, xnor, sll, srl, sla, sra, rol, ror, &).
- Последовательные операторы: назначение сигналов и переменных, wait, if, case, loop, for, while, return, null, function..
- Параллельные операторы: назначение сигнала, process, block, generate (for и if), создание экземпляра компонента, function, и вызов процедуры.
- Предопределенные атрибуты: t'base, t'left, t'right, t'high, t'low, t'succ, t'pred, t'val, t'pos, t'leftof, t'rightof, integer'image, a'left, a'right, a'high, a'low, a'range, a'reverse_range, a'length, a'ascending, s'stable, s'event.

Не поддерживаемые конструкции

- Типы доступа access, и файлы file.
- Инициализация значений сигналов и портов.
- Тип переменных real.
- Нельзя изменять значение переменной в разных процессах process.
- Запрещается использовать значения X, Z в выражениях.

Игнорируемые конструкции

Игнорируются задержки after.

Лабораторная работа. Программирование на VHDL в среде Max+Plus II RS- и D-триггеров

Цель работы

Освоение работы в текстовом редакторе и получение начальных навыков программирования на языке VHDL. Одновременно с этим исследуется работа таких устройств, как RS- и D-триггеры. Изучение основ языка проводится на примерах программных моделей RS-триггера. Закреплением полученного учебного материала является самостоятельное написание двух типов моделей для D-триггера.

Программа работы

1. Создать в графическом редакторе проект схемы RS-триггера на элементах "И-НЕ" (элемент nand2), откомпилировать и промоделировать его работу. Зарисовать временные диаграммы. Схема RS-триггера приведена на рис. 8.19.

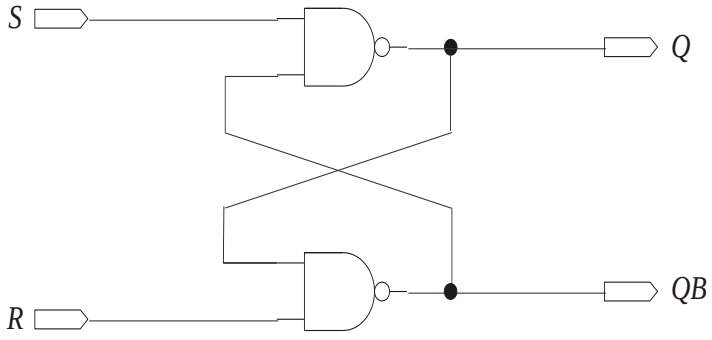

Рис. 8.19. Схема RS-триггера

2. Создать проект схемы RS-триггера в текстовом редакторе, используя язык проектирования схем VHDL.

- Для этого после выполнения команды меню **File | New** в открывшемся диалоговом окне выбрать текстовый редактор **Text Editor**.

- Создать проект схемы логического элемента "И-НЕ" на языке VHDL. Сохранить файл схемы с расширением vhd. Имя проекта схемы должно совпадать с именем, указанным в блоке ENTITY. Текст программы приведен далее:

```
LIBRARY ieee;                -- подключение библиотеки ieee
USE ieee.std_logic_1164.ALL;  -- использование библиотечного
                              -- модуля, содержащего
                              -- дополнительные типы переменных.
ENTITY notand IS
PORT( a : IN std_logic;       -- описание входов и выходов
    b : IN std_logic;         -- устройства (IN — вход, OUT —
    c : OUT std_logic );      -- выход, INOUT — двунаправленный
END notand;                   -- сигнал)
ARCHITECTURE behavior OF notand IS
BEGIN
  C <= NOT ( a AND b );
END behavior;
```

- Откомпилировать проект и получить временные диаграммы. Сравнить с таблицей истинности элемента "И-НЕ".
- Создать проект схемы RS-триггера на основе логических элементов "И-НЕ", используя язык VHDL. Пример программы, описывающей работу этого триггера, приведен ниже (структурная модель):

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY rstr IS
PORT( s : IN std_logic;
    r : IN std_logic;
    q : INOUT std_logic;
```

```
    qb : INOUT std_logic );
END rstr;
ARCHITECTURE behav OF rstr IS
COMPONENT notand    -- описание используемого компонента
PORT( a : IN std_logic;
      b : IN std_logic;
      c : INOUT std_logic);
END COMPONENT;
BEGIN
u1: notand                -- указание u1, как компонента notand
   PORT MAP ( s, qb, q);   -- указание входов и выхода для u1
u2: notand
   PORT MAP (q, r, qb);
END behav;
CONFIGURATION con OF rstr IS
   FOR behav
      FOR u1, u2: notand
          USE ENTITY work.notand (behavior); -- определяет интерфейс
END FOR;    -- и модель компонента notand
 END FOR;
END con;
```

Примечание

Обратите внимание, что соединение двух одинаковых частей RS-триггера (u1 и u2), описываемых одним компонентом notand, задано в описании карт портов (Port map) этих частей u1 и u2.

- Откомпилировать проект и получить временные диаграммы. Сравнить с результатами моделирования, полученными в п.1.

- Реализовать проект RS-триггера, используя поведенческую модель. Для этого в текстовом редакторе можно выбрать шаблон программной модели триггера (**Full design: Flipflop**) в меню **Templates | VHDL template** и внести соответствующие изменения или набрать текст следующей программы:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY rstr1 IS
PORT( s : IN std_logic;
     r : IN std_logic;
     q : OUT std_logic );
END rstr1;
ARCHITECTURE behav OF rstr1 IS
   SIGNAL qs:std_logic;
BEGIN
PROCESS (s,r)
   BEGIN
      IF s='1' THEN
         IF r='1' THEN qs<=qs;
         ELSE qs<='0';
         END IF;
      ELSE qs<='1';
      END IF;
END PROCESS;
q<=qs;
END behav;
```

- Откомпилировать проект и получить временные диаграммы. Сравнить с результатами моделирования, полученными в п.1.

3. Создать в графическом редакторе проект схемы D-триггера на элементах "И-НЕ" (элемент nand2), откомпилировать и промоделировать его работу. Зарисовать временные диаграммы. Схема D-триггера приведена на рис. 8.20. В качестве элемента RS-триггера использовать схему, приведенную в п.1.

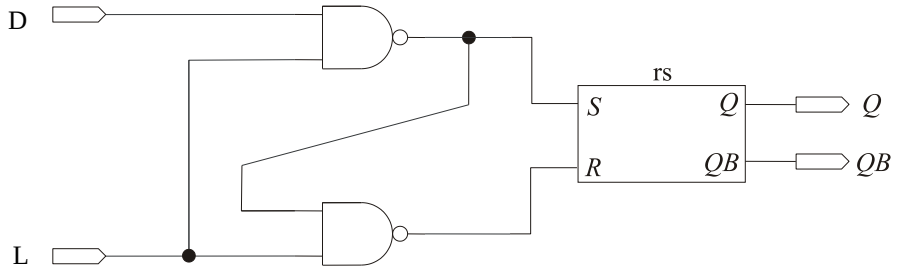

Рис. 8.20. Схема D-триггера

4. Реализовать проект D-триггера на языке программирования VHDL, используя структурную и поведенческую модели функционирования устройства. Откомпилировать проекты и получить временные диаграммы. Сравнить их с диаграммами, полученными в п.3.