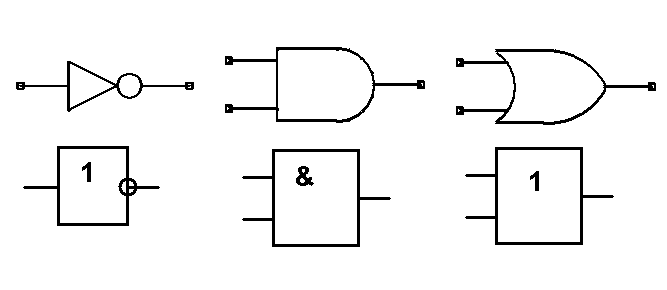
|  |  |
| --- | --- |
| ДИСЦИЛИНА | **Схемотехника устройств компьютерных систем Часть 2** |
|  |  |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **вычислительной техники** |
|  |  |
| ВИД УЧЕБНОГО | **Лекция** |
| МАТЕРИАЛА |  |
| ПРЕПОДАВАТЕЛЬ | **Тарасов И.Е.** |
|  |  |
| СЕМЕСТР | 3 |
|  |  |

4. Реализация базовых устройств цифровой схемотехники

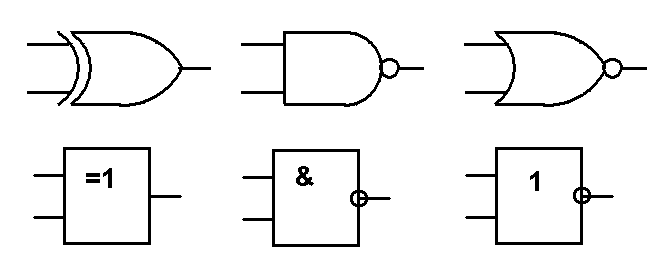
4.1. Основные цифровые элементы

В цифровой схемотехнике выделяют асинхронные (также называемся комбинационными) и синхронные элементы. Комбинационные элементы отличаются тем, что состояние их выхода однозначно определяется комбинацией значений на входах. Основными разновидностями комбинационных логических элементов являются элементы НЕ, И, ИЛИ. Их графическое изображение показано на рис. 4.1. Верхний ряд соответствует изображению по стандарту IEEE, нижний – по ЕСКД.



*Рисунок 4.1 Базовые логические элементы*

К дополнительным логическим элементам относят ИСКЛЮЧАЮЩЕЕ ИЛИ, И-НЕ, ИЛИ-НЕ. Они показаны на рис. 4.2. Аналогично предыдущему рисунку, верхний ряд соответствует изображению по стандарту IEEE, нижний – по ЕСКД.



*Рисунок 4.2 Дополнительные логические элементы*

Удобным способом описания работы логических вентилей является составление таблиц истинности. В такую таблицу записывается состояние выхода в зависимости от комбинации входных сигналов. Входы обычно обозначаются начальными буквами латинского алфавита, т.е., A, B, C, D, а для выхода используют обозначение Q. Для элементов, показанных на рис. 4.1, таблицы истинности будут выглядеть следующим образом.

Таблица 4.1. Таблицы истинности для базовых логических элементов

|  |  |
| --- | --- |
| Элемент НЕ | |
| A | Q |
| 0 | 1 |
| 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Элемент И | | |  | Элемент ИЛИ | | |
| A | B | Q | A | B | Q |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Таблицы истинности для элементов, показанных на рис. 4.2, приведены в табл. 4.2.

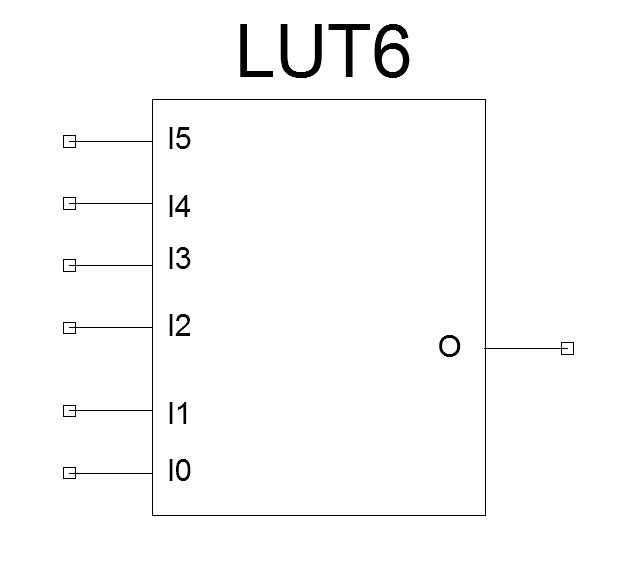
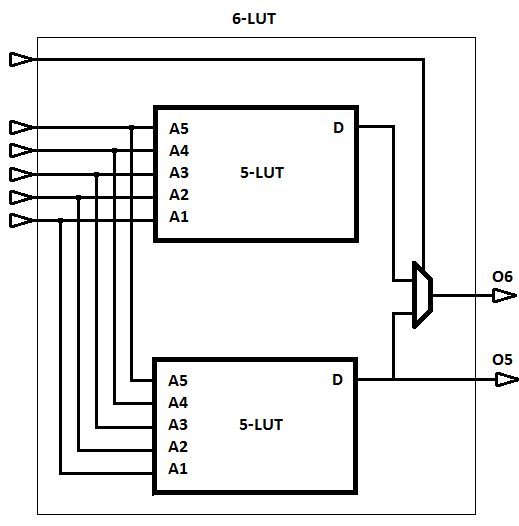
Таблица 4.2. Таблицы истинности для вспомогательных логических элементов

|  |  |  |
| --- | --- | --- |
| Элемент ИСКЛЮЧАЮЩЕЕ ИЛИ | | |
| A | B | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Элемент И-НЕ | | |  | Элемент ИЛИ-НЕ | | |
| A | B | Q | A | B | Q |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |

4.2. Реализация логических выражений в ПЛИС

Особенностью ПЛИС является отсутствие показанных выше элементов в явном виде. Вместо этого в логических ячейках имеются генераторы комбинационных выражений (LUT, Look-Up Table), которые непосредственно хранят таблицу истинности.

*Рисунок 4.3 Таблица истинности в составе ПЛИС*

Реализация комбинационных выражений в цифровой электронике производится с помощью оператора assign (в языке Verilog). Он также называется *оператором непрерывного присваивания* (continuous assignment operator). Пример выражения для элемента И показан ниже:

assign c = a & b;

В отличие от языков программирования, где знак «равно» воспринимается как «присвоить переменной значение и перейти к следующему оператору», в языках описания аппаратуры такие операторы удобнее интерпретировать как «соединить». После такого соединения назначить другое выражение сигналу невозможно, поскольку это означало бы короткое замыкание между двумя проводниками.

Это различие очень важно для понимания языков описания аппаратуры. Программистам привычнее воспринимать операторы как однократные действия, после которых можно совершить другие действия с этой переменной. Попытки же воспринимать тексты на Verilog в подобном ключе приведут к неустранимым ошибкам синтезатора, который не сможет создать схему, в которой один и тот же сигнал будет описываться разными правилами.

Для оператора assign необходимо искать такие выражения, которые будут справедливы на протяжении всего времени работы схемы. Последовательная смена состояний, аналогичная выполнению программы, реализуется с помощью тактируемых схем – например, конечных автоматов.

Ниже показан более сложный пример, использующий основные логические операторы, имеющиеся в Verilog. Из примера видно, что выражения в Verilog записываются в целом по тем же правилам, что и в языках программирования. Обозначения операторов аналогичны языку Cи (используются символы &|^~, а не буквенные операторы and or xor not).

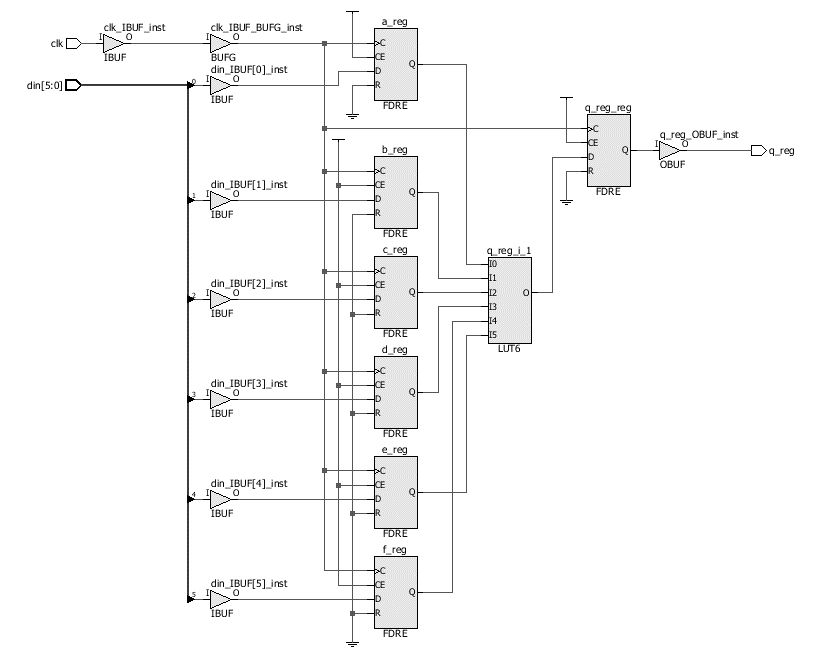
assign q = (~a | b) ^ (c & d);

Реализация логических функций в виде таблицы истинности предполагает анализ не сложности выражений, а количества входов в этом выражении.

Рассмотрим выражение на Verilog:

assign q = (a & b) | ((c ^ d) & (e | f));

Оно выглядит громоздким, однако использует 6 входных аргументов. Поэтому его реализация потребует всего одной LUT, как показано на рис. 4.4. На нем видно, что входные и выходной сигналы, которые хранятся в триггерах FDRE, подключены к единственной LUT. Вне зависимости от количества операторов в выражении, 6 входов гарантируют, что любая таблица истинности поместится в 6-входовой LUT.

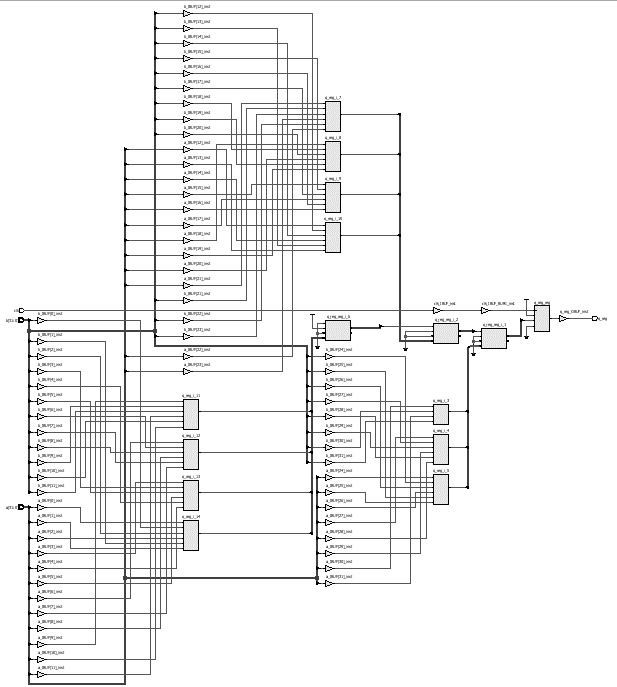


*Рисунок 4.4 Реализация комбинационного выражения*

Для сравнения рассмотрим выражение, которое выглядит более простым:

q <= '1' when a = b else '0';

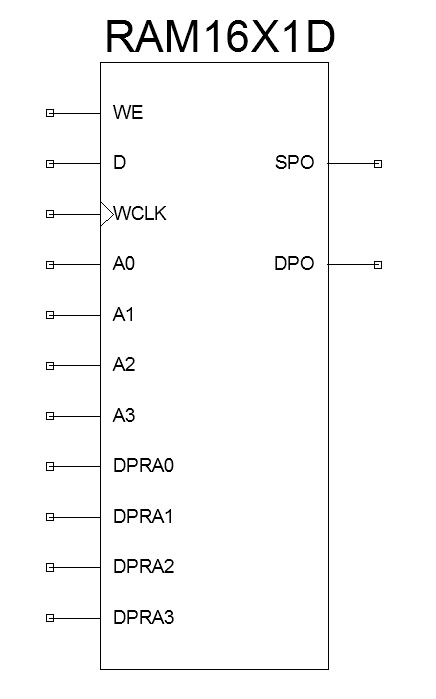
Тем не менее, сигналы a, b объявлены в модуле как 32-разрядные. Поэтому количество входов в логическом выражении равно 64, что очевидно не помещается в одну LUT. Результаты синтеза такого выражения для ПЛИС показаны на рис. 4.5.



*Рисунок 4.5 Реализация устройства сравнения двух операндов*

4.3. Дополнительные возможности логических ячеек ПЛИС

Поскольку логический генератор представляет собой элемент статической памяти, его можно использовать и в этом качестве. Этот режим поддерживается не всеми производителями ПЛИС, например, Xilinx обеспечивает работу LUT в режимах распределенной памяти (distributed memory) и сдвиговых регистров.



*Рисунок 4.6 Таблица истинности в режиме распределенной памяти (distributed memory)*

Распределенная память может работать в однопортовом или простом двупортовом (simple dual-port) режимах. Простой двупортовый режим имеет то ограничение, что только один порт может использоваться для чтения и записи, а второй предназначен только для чтения. Показанный на рис. 4.6 элемент распределенной памяти имеет следующие сигналы:

– we – сигнал разрешения записи;

– d – данные для записи;

– wclk – вход тактового сигнала;

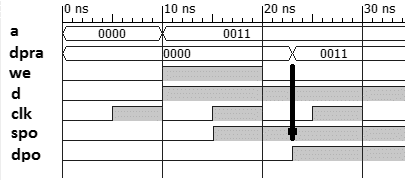
– a – адрес для записи;

– dpra – адрес для чтения, второй порт (dual port read address);

– spo – выход первого порта (single port output);

– dpo – выход второго порта (dual port output).

Временные диаграммы работы распределенной памяти показаны на рис. 4.7.

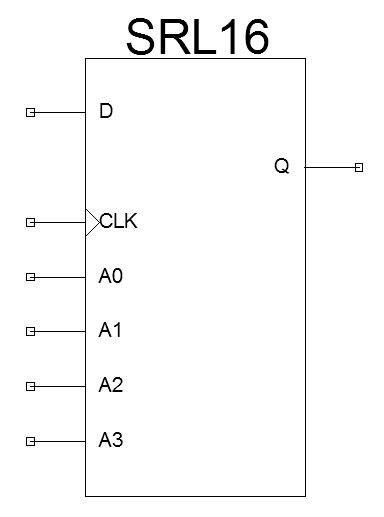


*Рисунок 4.7 Временные диаграммы работы распределенной памяти*

На рис. 4.7 видно, что запись в память происходит синхронно (в момент времени 15 нс, когда данные d = 1 записываются по адресу addr = 00112 = 310), а чтение – асинхронно. В момент времени 23 нс изменение адреса на входе dpra приводит к соответствующему изменению выхода dpo. Этот момент (без фронта тактового сигнала) был выбран специально, чтобы продемонстрировать, что память реагирует на изменение адреса асинхронно. Аналогично, изменение состояния входа a немедленно приводит к появлению на выходе spo значения из ячейки с адресом a.

Распределенная память удобна для организации небольших блоков данных – буферов, линий задержки, небольших таблиц. Реализация больших блоков на распределенной памяти в общем случае нецелесообразна из-за сильной фрагментации такого блока.

Другой вариант использования логического генератора – реализация на его базе сдвигового регистра. Графическое изображение такого компонента показано на рис. 4.8.



*Рисунок 4.8 Таблица истинности в режиме сдвигового регистра*

Сдвиговый регистр имеет следующие сигналы:

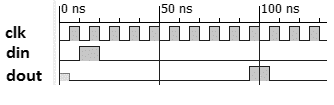
– d – данные для записи в сдвиговый регистр;

– clk – тактовый сигнал;

– a – адресный вход;

– q – выход данных.

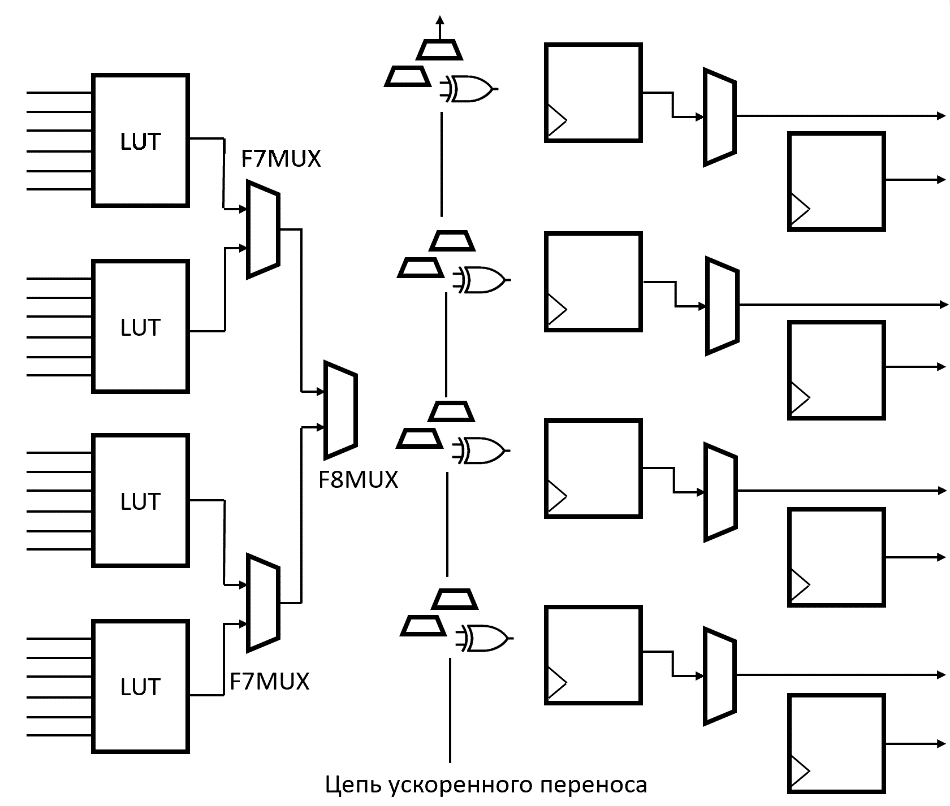
В простейшем варианте сдвиговый регистр может представлять собой модуль, задерживающий входной сигнал din на N тактов. Временные диаграммы работы 8-разрядного сдвигового регистра показаны на рис. 4.9.



*Рисунок 4.9 Временные диаграммы работы сдвигового регистра*

В составе матрицы ресурсов FPGA есть не только LUT и триггеры. Для реализации некоторых часто используемых цифровых узлов возможности LUT избыточны. Поэтому к группе LUT и триггеров добавляются дополнительные компоненты, выполняющие единственную функцию, которая часто оказывается полезной в проектах. Набор LUT, триггеров и дополнительных компонентов называется в FPGA *секцией* (slice). Этот термин используется Xilinx и не всегда применяется другими производителями в том же качестве.

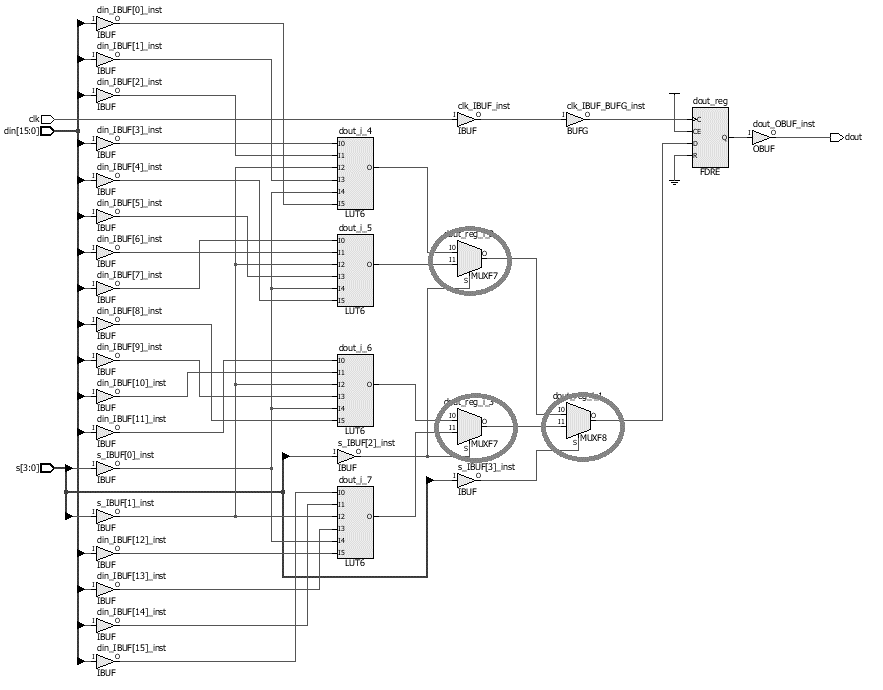
На рис. 4.10 показаны основные компоненты логической секции ПЛИС Xilinx.



*Рисунок 4.10 Основные компоненты логической секции ПЛИС*

К дополнительным компонентам можно в первую очередь отнести мультиплексоры и цепи ускоренного переноса. В левой части рис. 4.10 видно, что выходы LUT объединяются мультиплексорами F7MUX, а выходы этих мультиплексоров – дополнительно мультиплексором F8MUX.

Цифры 7 и 8 в обозначении мультиплексоров показывают, что они помогают расширить количество входов с 6 в LUT до 7 и 8 соответственно. На рис. 4.11 показано, как синтезатор использует эти мультиплексоры для реализации 16-входового мультиплексора. Для 16 входов требуется 4 разряда управляющего сигнала, поэтому общее количество входов становится равным 20. Это потребует 4 логических генератора, выходы которых нужно чем-то объединить. Мультиплексоры F7MUX и F8MUX позволяют завершить построение схемы в рамках одной секции, не привлекая еще одну LUT.



*Рисунок 4.11 Применение дополнительных компонентов логической секции (мультиплексоров) для реализации более сложного комбинационного выражения*

Синтезаторы применяют дополнительные мультиплексоры по мере возможности. Следует использовать оператор switch по мере возможности, чтобы облегчить распознавание шаблона мультиплексора. Пример:

always @ \*  
begin  
case (sel)   
 2’b00 : q = a;  
 2’b01 : q = b;  
 2’b10 : q = c;  
 2’b11 : q = d;  
 default : q = 1’bx;  
endcase   
end

В этом примере сигнал sel, очевидно, является сигналом выбора входа мультиплексора. Подобные шаблоны легко распознаются синтезаторами.

На рис. 4.10 также показана цепь ускоренного переноса. Она используется для реализации схем сложения и вычитания, которые будут рассмотрены далее. Как и мультиплексоры, эта цепь занимает немного места в секции, но помогает создавать схемы, для которых в противном случае были бы применены LUT.

4.4. Выводы по главе

Базовые логические операции реализуются в языках описания аппаратуры с помощью оператора непрерывного присваивания. В языке Verilog это оператор assign. В отличие от языков программирования, присваивание должно быть единственным в модуле для каждого сигнала. Удобно рассматривать такой оператор как описание соединений, которые будут оставаться в схеме на всем протяжении ее работы.

Особенностью ПЛИС является применение таблиц истинности для реализации логических выражений. Поэтому при проектировании нужно следить не за сложностью выражений, а за количеством его входов.

Ячейки ПЛИС имеют дополнительные возможности, которые автоматически используются синтезаторами. Такие схемы можно создать и на базе только логических генераторов в таблицах истинности, но дополнительные компоненты делают их компактнее и быстрее. К дополнительным функциям относятся:

- распределенная память;

- сдвиговые регистры;

- аппаратные мультиплексоры на выходах логических генераторов;

- цепи ускоренного переноса.

Контрольные вопросы:

1. Какие виды логических элементов существуют в цифровой схемотехнике?

2. Что такое таблица истинности и как она используется в ПЛИС?

3. Влияет ли сложность выражения на количество таблиц истинности, необходимых для его реализации?

4. Какие дополнительные компоненты имеются в секции ПЛИС? Какие функции они помогают реализовать?

5. Почему с помощью оператора assign нельзя реализовать двоичный счетчик?