СХЕМОТЕХНИКА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ СХЕМ.

2.1.Способы описаний логических схем

Логические элементы могут быть соединены друг с другом, в результате чего образуется **логическая схема**. На **рис.2.1**. представлена схема, состоящая из двух соединенных элементов И.

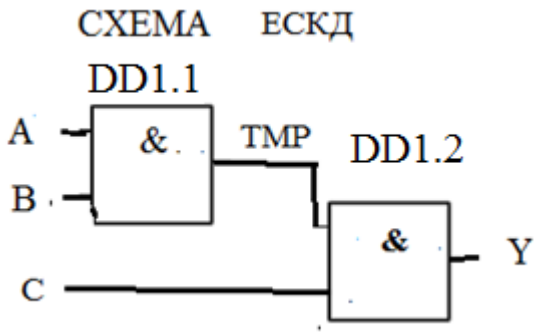


Рис.2.1. Схема, состоящая из двух элементов И.

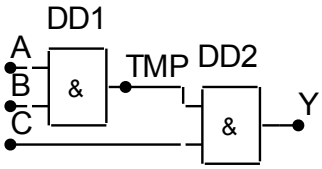
Обычно **под схемой цифрового устройства** понимают графическую форму его описания, содержащую имена входных и выходных сигналов, УГО элементов и отображение их связей. Схемы можно так же описывать, используя другие формы, например**, табличные, алгебраические, HDL описания** и т.д.

На **рис.2.2** представлены графический и табличный способы описания схемы, реализующей логическую функцию 3И (Y= A & B & C) на элементах 2И (Y= (A & B) & C). Слева на рисунке показаны примеры **графического** представления схемы в стандартах Единой Системы Конструкторской Документации (**ЕСКД**), принятой в России и ниже- Американского национального стандарта (**ANSY**). Слева входы схемы (A, B, C), справа выход Y.

Справа на рис.2.2 представлен **табличный способ** описания схемы и ниже- **алгебраическая форма** описаний схем- описание в форме Булевских уравнений. Сигнал TMP-промежуточный, связывающий два элемента 2И.

В табличной форме описания учитывается тот факт, что конструктивно несколько простых логических элементов реализуются в одном корпусе (микросхема ЛИ1 содержит 6 элементов 2И). В строке таблицы - слева тип микросхемы, потом имя используемого ее элемента, потом имена входных сигналов элемента и затем –имя выходного сигнала.

Схема ЕСКД



ТИП	ИМЯ	СВЯЗИ		
ЛИ1	DD1	**A**	**B**	TMP
ЛИ1	DD2	TMP	**C**	Y

СИСТЕМА УРАВНЕНИЙ

TMP = A & B

Y = TMP & C

Схема ANSI

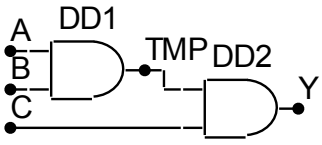


Рис.2.2. **Графический, табличный и алгебраический способы описания схемы, реализующей функцию 3И (Y= A & B & C) на элементах 2И (микросхема ЛИ1)**

.

В примере ниже приведены два варианта описания схемы **рис.2.2** на VHDL - **структурное** (аналог таблицы **рис.2.2,** справа вверху) и **потоковое** (аналог системы уравнений рис.**2.2**, справа, внизу).

```
entity AND_2 is  port (D1,D2 : in bit; Y : out bit); end;        [1][2]
architecture  BEHAVIOR of AND_2 is
   begin  Y <=  (D1  and D2) ;  end;

entity AND_3 is    port (A,B,C : in bit; Y : out bit); end;
--структурное описание  архитектуры  ниже ----------------
architecture  STRUCT of AND_3 is
 component  AND_2 is port (D1,D2 : in bit; Y : out bit);end component;
 signal TMP:bit;
begin
DD1:   AND_2  port map (A,B,TMP);
DD2:   AND_2  port map (TMP,C,Y);
end;
--потоковое описание  архитектуры ниже--------------
 architecture  DAT_FLOW of AND_3 is
 signal TMP:bit;
  begin
   TMP<= A and B;       Y<= TMP and C;
  end;
```

2.2. Проектирование комбинационных схем с использованием простейших логических элементов - логических вентилей

Схемы, выходные сигналы которых полностью определяются текущими значениями входных сигналов называются **комбинационными (КС).** Иными словами КС - это схемы, не имеющие памяти.

Проектирование комбинационных схем традиционным способом сводится к следующим этапам.

1. Построение таблицы истинности логической функции, определяющей зависимость состояний выходных сигналов от входных.
2. Составление системы булевских уравнений, описывающих эти зависимости.
3. Минимизацию системы булевских уравнений.
4. Покрытие этих уравнений функциями элементов используемого логического базиса с учетом нагрузочных соотношений.

Совершенная дизъюнктивная нормальная форма (СДНФ) булевской функции представляет собой дизъюнкцию всех конъюнкций входных сигналов, дающих 1 в ее таблице истинности.

Совершенная конъюнктивная нормальная форма (СКНФ) – конъюнкция всех дизъюнкций, дающих 0 в таблице истинности.

Минимимизация Булевских уравнений преследует цель уменьшения количества логических операций и вхождений аргументов в булевские уравнения.

2.2.1. Реализация логической функции XOR

В качестве первого примера рассмотрим проектирование схемы, реализующей функцию **ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR**) с использованием логических вентилей **НЕ, И, ИЛИ**.

Таблица истинности логической функции **ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR**) представлена ниже (табл. 2.3), где X0, X1 – аргументы, Y- значение функции

Таблица 2.3. Таблица истинности логической функции

ИСКЛЮЧАЮЩЕЕ ИЛИ(XOR)

X0	X1	Y
0	0	0
0	1	1
1	0	1
1	1	0

Совершенная дизъюнктивная нормальная форма (СДНФ) булевской функции представляет собой дизъюнкцию всех конъюнкций входных сигналов, дающих 1 в ее таблице истинности.

В нашей таблице Y=1 во второй и третьей строках таблицы.

Соответственно ее СДНФ следующая

$$Y = \overline{X0} \cdot X1 + X0 \cdot \overline{X1}$$

Где * символ логической операции И-коньюнкции, а символ + символ операции ИЛИ-дизьюнкции.

Совершенная конъюнктивная нормальная форма (СКНФ) – конъюнкция всех дизъюнкций, дающих 0 в таблице истинности

В нашей таблице Y=0 в первой и четвертой строках таблицы.

Соответственно ее СКНФ следующая

$$Y = \overline{X0} + \overline{X1} \cdot X0 + X1$$

Этап так называемой минимизации мы пока опускаем, а этап покрытия функции элементами логического базиса И, ИЛИ, НЕ очевиден.

Например, при реализации СДНФ сначала входные сигналы надо пропустить через инверторы, потом через вентили И, потом через вентиль ИЛИ. Критический путь сигнала в этой схеме равен 3.

На **рис.2.3** иллюстрируется последовательность (слева-направо) этапов проектирования и представлены: Таблица истинности, СДНФ и схема, реализующая функцию ИСКЛЮЧАЮЩЕЕ ИЛИ в логическом базисе вентилей НЕ, 2И, 2ИЛИ. Стрелки, ведущие из таблицы к СДНФ (**рис.2.3**) показывают источник конъюнкций в совершенной дизъюнктивной нормальной форме. В данном примере существуют только две строки таблицы истинности, в которых значение выхода Y равно 1.

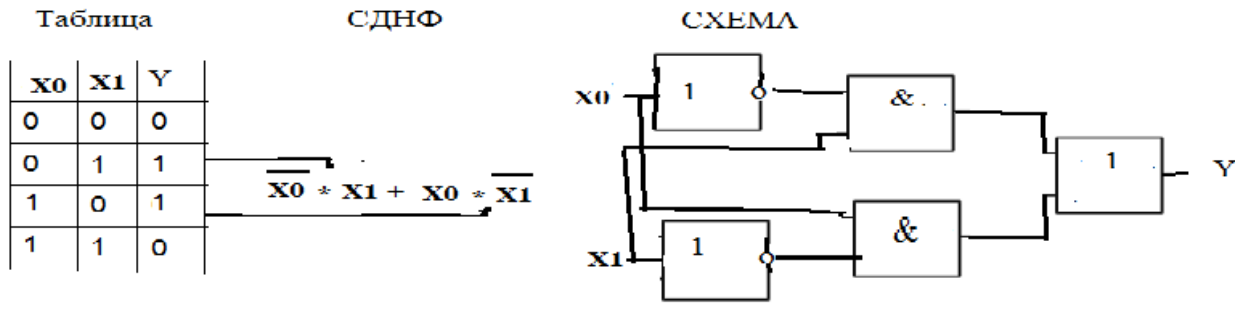


Рис.2.3. Таблица истинности, СДНФ и схема, реализующая функцию Исключающее ИЛИ(XOR) в логическом базисе НЕ, И, ИЛИ

Схема рис.2.3 содержит 5 логических вентилей, которые расположены в три яруса(каскада**).** Самый длинный путь распространения сигнала от входа до выхода схемы проходит через три элемента. **Этот самый длинный путь называется критическим и определяет максимальную задержку схемы (при равных задержках элементов) и так называемую глубину схемы**

2.2.2. Элементы Булевской алгебры

Напомним некоторые сведения из теории Булевской алгебры, используемые при минимизации Булевских уравнений, описывающих логические схемы (**табл 2.4 и табл.2.5**).

Аксиомы и теоремы Булевой алгебры подчиняются **принципу двойственности**. Если взаимно заменить символы 0 и 1, а также взаимно заменить операции И (&) и ИЛИ (|), то булево выражение останется верным. Например, аксиома 0 |0=0 по принципу двойственности эквивалентна аксиоме

1 &1=1.

Таблица. 2.4. **Аксиомы булевой алгебры**

~0=1	~1=0	0 | 0=0	1 | 1=1	0|1= 1	1 & 1= 1	0 & 1=0
0 |A=A	1 |A=1	A&A=A	0& A=0	A |A=A	A&~A=0	~ ~ A=A
A | ~A= 0						

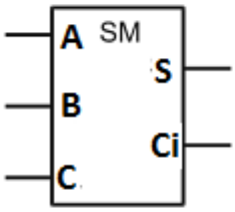
Таблица 2.5. Законы (соотношения, теоремы) Булевой алгебры

Номер	Соотношение	Двойственное соотношение	Название
1	A &B= B & A	A | B= B |A	Коммутативность
2	(A & B) & C= A &(B&C)	(A |B) | C=A|(B|C)	Ассоциативность
3	(A&B)|(A&C)=A&(B|C)	(A|B)&(A|C)=A|(B&C)	Дистрибутивность
4	A&(A|B)=A	A|(A&B)=A,	Поглощение
5	(A&B) |(A & ~B)=A	(A|B) &(A |~B)=A	Склеивание
7	~(A1&A2&A3 ..)= (~A1 |~A2|~A3 ..)	~(A1|A2|A3 ..)= (~A1&~A2&~A3..)	Теорема Де Моргана

Множество логических операций, с помощью которого можно отобразить любую логическую функцию называется полным. Например, полными являются множество операций **И, ИЛИ**, **НЕ** или множество из одной операции **И-НЕ** или множество из одной операции **ИЛИ-НЕ**.

2.2.4. Реализация логической функции двоичного суммирования

В качестве второго примера из области проектирования комбинационных схем рассмотрим проектирование схемы одноразрядного двоичного сумматора. На **рис.2.4** представлены его УГО и таблица истинности логической функции, где A, B, C- входы, S- выход суммы, Ci-перенос в следующий разряд.



A	B	C	S	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Рис.2.4. УГО и таблица истинности одноразрядного сумматора.

Если использовать СДНФ для описания логической функции двоичного сумматора в терминах Булевской алгебры, то надо представить каждый выходной сигнал(S, Ci) как дизъюнкцию(ИЛИ) конъюнкций (**термов- И**) входных сигналов всех строк таблицы истинности (см. **рис. 2.4**), в которых выходной сигнал равен 1. Одноразрядный двоичный сумматор описывают следующие формулы СДНФ (булевские уравнения) в которых операция **И** обозначена символом &, **ИЛИ** символом |, **НЕ** символом ~, которые более удобны при наборе в word текста:

Формулы 1

S= (A & ~B&~C) | (~A&B &~C) | (~A&~B &C) | (A&B&C)

Ci= (~A&B&C) | (A&~B&C) | (A&B &~C) | (A&B &C)

Если без минимизации, прямо по **формулам 1** строить схему сумматора и использовать экзотический элементный базис, включающий вентили 3И с различными инверсными входами(они реализуют коньюнкции с тремя входами ,часть из которых инвертирует входной сигнал) и 4ИЛИ(они реализуют четырехвходовую дизьюнкцию), то получится двухкаскадная схема (общее для S и Ci выражение (A&B &C) в схеме не дублируется в целях минимизации оборудования) , состоящая из семи вентилей 3И(некоторые их входы инверсны) и двух 4ИЛИ.

Можно немного пожертвовать быстродействием схемы, увеличив число каскадов до трех и перейти к обычному элементному базису из вентилей 3И, 4ИЛИ, НЕ. Непосредственное воспроизведение формул 1 в трехкаскадной схеме потребует применения 3-х вентилей НЕ, 7-и вентилей 3И, 2-х вентилей 4ИЛИ (всего 12 вентилей).

Читателю предлагается самостоятельно, используя информацию, представленную в **табл. 2.4 и 2.5** , минимизировать формулу для переноса Ci на базе СДНФ, представленной в **Формулах 1**. Например, вынося за скобки общие множитель B&C (используется свойство дистрибутивности-(см. табл 2.5.-соотношение 3)) для первой и последней коньюнкций СДНФ и вынося общий множитель A из второй и третьей , получим следующую последовательность преобразований первого уравнения из Формул 1.

Ci= (~A&B&C) | (A&~B&C) | (A&B &~C) | (A&B &C) – исходная СДНФ

Ci=(B&C& (~A |A)) | (A&((~B&C) | (B&~C))) -вынос общих множителей

Учитывая, что (~A |A)=1 в результате получаем меньшее число операций И-&(не 8, а 4 операции) и ИЛИ - | (не 4 , а 3 операции).

Ci=(B& C) |(A&((~B&C) | (B&~C))) .

В данном простом случае проектирования одноразрядного сумматора, нетрудно заметить (см.таблицу истинности **рис.2.4**), что перенос Ci в сумматоре возникает только при наличии значений любых двух входных сигналов, равных 1. Сумма S равна 1 только тогда, когда отсутствует перенос и только один из входных сигналов равен 1, или все входные сигналы равны 1 или либо только один из входов равен 1, либо все три входа равны 1.Формулы 2 отражают эти выводы.

Формулы 2

Ci= (B&C) | (A&C) | (A&B)

S=~Ci & (A|B|C) | (A&B&C).

Схема, реализующая уравнения Формулы 2 представлена на **рис.2.5.**

Для реализации минимизированного описания сумматора (**Формулы** 2) в элементном базисе НЕ,2И,2ИЛИ потребуется использовать один вентиль НЕ, 6 вентилей 2И, 5 вентилей 2ИЛИ. Схема будет иметь 5 каскадов - при вычислении значения Ci сигнал проходит через 3 элемента-(2И,2ИЛИ,2ИЛИ), а при вычислении S дополнительно еще через элемент НЕ и элемент 2И. В элементном базисе 2-2-2И-ИЛИ-НЕ и 3-3-3-3И-ИЛИ-НЕ для инверсных выходов сумматора (~Ci,~S) согласно **формулам 2** имеем двухкаскадную схему **рис.2.5**.

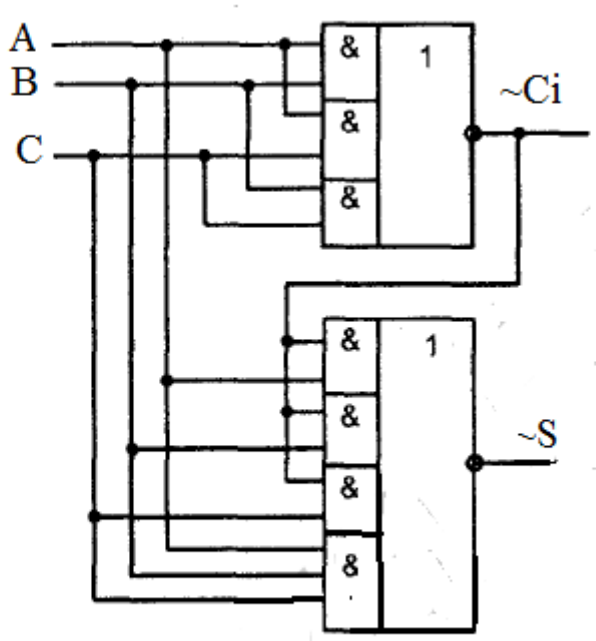


Рис. 2.5. Двухкаскадная схема сумматора на элементах И-ИЛИ-НЕ

Использование других элементных базисов часто позволяет получать более экономные по аппаратуре и быстродействующие решения. Например, в базисе «трехвходовое исключающее ИЛИ»- 3XOR имеем S=A ^ B^ C, т.е. достаточно одного трехвходового вентиля «Исключающее ИЛИ-3XOR» или двух вентилей 2XOR для получения суммы S.

В реальной жизни кроме традиционных критериев типа минимизации затрат аппаратуры и максимизации быстродействия схем системами автоматизации проектирования учитываются критерии минимизации используемой схемой площади кристалла, минимизации мощности потребляемой от источника питания, достижения максимальной регулярности схемы и т.п. Исходной информацией для САПР в этой работе

является представленное проектировщиком HDL-описание проекта, критерии оптимизации и заданный им логический базис.

2.2.5. Реализация логической функции F

Рассмотрим третий пример схемной реализации логической функции F от трех переменных с аргументами X0, X1, X2 и значением Y, заданной в виде таблицы истинности, приведенной в таб. 2.2.

Таблица 2.2. Таблица истинности логической функции F.

№	X0	X1	X2	Y	F(X0, X1, X2)	Минтерм	Макстерм
0	0	0	0	0	Y = F(0,0,0)	$\overline{X0}\cdot\overline{X1}\cdot\overline{X2}$	$X0+X1+X2$
1	0	0	1	1	Y = F(0,0,1)	$\overline{X0}\cdot\overline{X1}\cdot X2$	$X0+X1+\overline{X2}$
2	0	1	0	1	Y = F(0,1,0)	$\overline{X0}\cdot X1\cdot\overline{X2}$	$X0+\overline{X1}+X2$
3	0	1	1	0	Y = F(0,1,1)	$\overline{X0}\cdot X1\cdot X2$	$X0+\overline{X1}+\overline{X2}$
4	1	0	0	0	Y = F(1,0,0)	$X0\cdot\overline{X1}\cdot\overline{X2}$	$\overline{X0}+X1+X2$
5	1	0	1	1	Y = F(1,0,1)	$X0\cdot\overline{X1}\cdot X2$	$\overline{X0}+X1+\overline{X2}$
6	1	1	0	1	Y = F(1,1,0)	$X0\cdot X1\cdot\overline{X2}$	$\overline{X0}+\overline{X1}+X2$
7	1	1	1	0	Y = F(1,1,1)	$X0\cdot X1\cdot X2$	$\overline{X0}+\overline{X1}+\overline{X2}$

Запишем логическую функцию F, соответствующую этой таблице в виде СДНФ, используя для обозначения операции коньюнкции символ *, дизьюнкции +, отрицания- символ надчеркивания.

$$Y = \overline{X0}\cdot\overline{X1}\cdot X2 + \overline{X0}\cdot X1\cdot\overline{X2} + X0\cdot\overline{X1}\cdot X2 + X0\cdot X1\cdot\overline{X2};$$

Схемотехническая реализация функции F *на* ***элементах НЕ, И и ИЛИ***.

Каждому набору коньюнкцй (минитерму) ставим в соответствие логический элемент И с необходимым числом входов. Выходы этих элементов объединяем при помощи логических элементов ИЛИ. Инверсию входных сигналов обеспечивают инверторы. Такая трехуровневая реализация логической функции не оптимальна. В ней, во-первых, применяется излишнее число элементов И. Во-вторых, задержка распространения сигналов в элементах И и ИЛИ обычно больше, чем в элементах И-НЕ и ИЛИ-НЕ.

Схемотехническая реализация логической функции F, соответствующей табл. 2.2, на ***элементах НЕ, И и ИЛИ*** представлена на рис. 2.6.

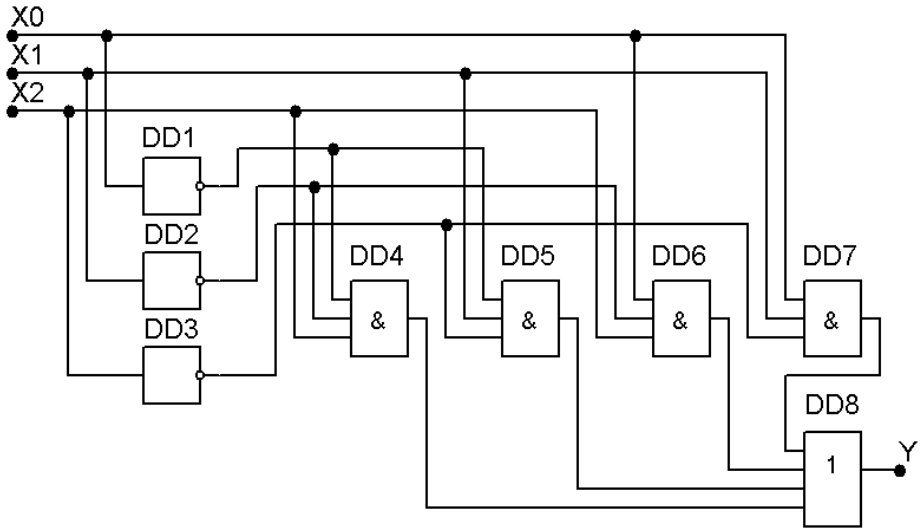


Рис. 2.6. Схемотехническая реализация логической функции F, соответствующей табл. 2.2, *на элементах НЕ, И и ИЛИ*

1. Схемотехническая реализация логической функции F *на логических элементах НЕ,* И *и* ИЛИ *на основе ее минимизированного представления*

Минимизация обычно преследует цель уменьшить количество операций и вхождений аргументов.

После минимизации получим следующую логическую функцию.

$$Y = \overline{X1} \cdot X2 + X1 \cdot \overline{X2};$$

В результате получим следующую комбинационную схему рис.2.7.

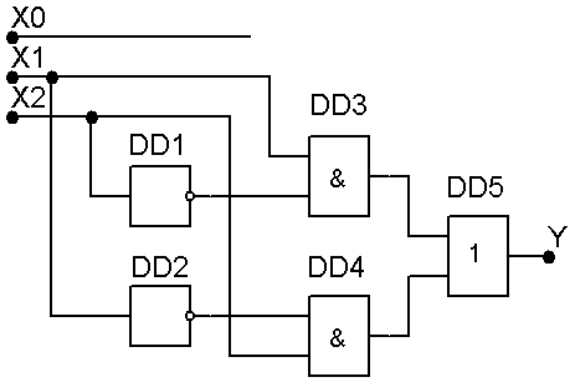


Рис. 2.7. Схемотехническое описание минимизированной логической функции на элементах И и ИЛИ, заданной в табл. 2.2.

Новая минимизированная схема более экономична, так как имеет на 3 логических элемента меньше, хотя максимальная задержка сигнала от входа до выхода схемы осталась неизменной.

Однако и эта схемотехническая реализация не оптимальна, так как время задержки распространения сигналов в логических элементах И и ИЛИ обычно больше, чем в логических элементах И-НЕ и ИЛИ-НЕ.

Применение этих элементов позволяет сделать схему с меньшими задержками.

2. *Формальная реализация логических функций на элементах* И-НЕ *и* ИЛИ-НЕ.

Преобразуем логическую функцию F (табл.2.2) по теореме Де Моргана. Получим [поэтапное упрощение]

$$\overline{Y} = \overline{\overline{X0}\cdot\overline{X1}\cdot X2} + \overline{\overline{X0}\cdot X1\cdot\overline{X2}} + \overline{X0\cdot\overline{X1}\cdot X2} + \overline{X0\cdot X1\cdot\overline{X2}};$$

Это представление логической функции позволяет осуществить ее схемотехническую реализацию на элементах И-НЕ и ИЛИ-НЕ. Очевидным и очень полезным свойством этого преобразования является следующий простой факт: *необходимо в схеме на* рис. 2.7. *формально заменить элементы* И *и* ИЛИ *на элементы* И-НЕ.

3. ***Реализация функции F (см. табл.2.2) на логических элементах* И-НЕ *и* ИЛИ-НЕ *на основе ее минимизированного представления***

. Преобразуем минимизированную логическую функцию на основе теоремы Де Моргана. Получим

$$\overline{Y} = \overline{\overline{X1}\cdot X2} + \overline{X1\cdot\overline{X2}};$$

Схемотехническая реализация этой логической функции приведена на рис. 2.8.

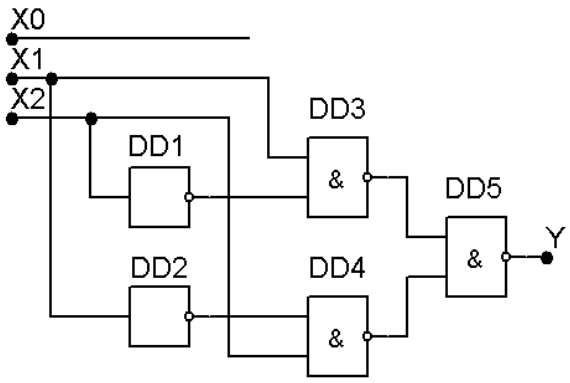


Рис. 2.8. Схемотехническая реализация заданной в табл. 2.2. минимизированной логической функции F на элементах И-НЕ.

Замечание. Также просто проводится схемотехническая реализация логических выражений, заданных в конъюнктивной нормальной форме. В

этом случае в первом слое схемы используются логические элементы ИЛИ, а объединение их выходов проводится на логических элементах И.

Замечание. Очень часто таблицы истинности комбинационной схемы являются неполными. Это случается, когда часть входных комбинаций логической функции никогда не встречается на практике. Этот факт можно использовать для упрощения схемотехнической реализации логической функции.

Замечание. Не следует тратить больших усилий на минимизацию функций с большим числом переменных вручную. Если выполнить описание логической функции на VHDL , то имеется возможность с помощью САПР автоматически минимизировать заданную логическую функцию и реализовать ее в заданном элементном базисе.

Приведем описание на VHDL узла(**рис.2.9**), соответствующего логической функции F, заданной в табл. 2.2.

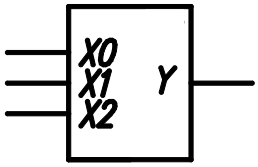


Рис. 2.9. Условное графическое обозначение устройства Log_F_1.

```
entity Log_F_1 is
        port(X0 : in BIT;
             X1 : in BIT;
             X2 : in BIT;
             Y : out BIT);
  end Log_F_1;
```

Устройство с именем Log_F_1 имеет входы X0, X1, X2 и выход Y.

Логическая функция устройства описана в виде СДНФ в разделе Архитектура с именем Log_F_ARC.

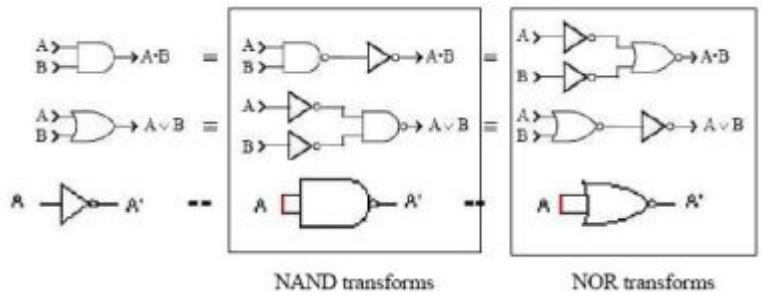
```
architecture Log_F_ARC of Log_F_1 is
  begin
Y<=(not (X0) and not (X1) and X2) or (not (X0) and X1 and not (X2))
  or (X0 and not (X1) and X2) or (X0 and X1 and not (X2));
end Log_F_ARC;
```

Описание логической функции выполнено при помощи логических операций языка VHDL **and, or и not**. Логическая функция не минимизирована, так как при автоматизированном проектировании предполагается, что это сделает САПР.

2.2.6. Реализация схем в логических базисах И-НЕ и ИЛИ –НЕ

Хотя традиционные методы представления и минимизации логических функций базируются на базисе И,ИЛИ,НЕ , следует учитывать, что логические базисы И-НЕ и ИЛИ-НЕ являются полными и многие функции реализуются в этих базисах проще, чем в традиционном базисе И,ИЛИ,НЕ. Ниже представлен один из способов преобразования

схем из традиционного логического базиса в базис И-НЕ или ИЛИ-НЕ с использованием теоремы Де Моргана.



- .
- **Рис. 2.10. Преобразование схем с применением теоремы Де Моргана**

- Первые два столбца иллюстрируют теорему Де Моргана при реализации в базисе И-НЕ, третий столбец иллюстрирует реализацию в базисе ИЛИ-НЕ.
- Последовательность действий по переводу схемы в нетрадиционный логический базис И-НЕ примерно следующая.
- На базе Таблицы истинности логической функции составляется ее описание в СДНФ, которое минимизируется, после чего рисуется схема в традиционном логическом базисе И, ИЛИ, НЕ.
- Применяется теорема Де Моргана. (см. предыдущий рисунок). При переводе в И-НЕ используется средняя колонка, при переводе в ИЛИ-НЕ правая.
- Исключаются избыточные инверторы (цепочки из двух последовательно соединенных инверторов)..
- 4.Оставшиеся инверторы замещаются Эквивалентными И-НЕ и ИЛИ
- Ниже иллюстрирующий пример преобразования в базис И-НЕ.
- А) Исходная Схема в традиционном базисе И, ИЛИ, НЕ
- В) Промежуточная Схема (в пунктирных прямоугольниках подсхемы, заменяемые эквивалентными И-НЕ),
- С) Окончательная Схема в базисе И-НЕ.

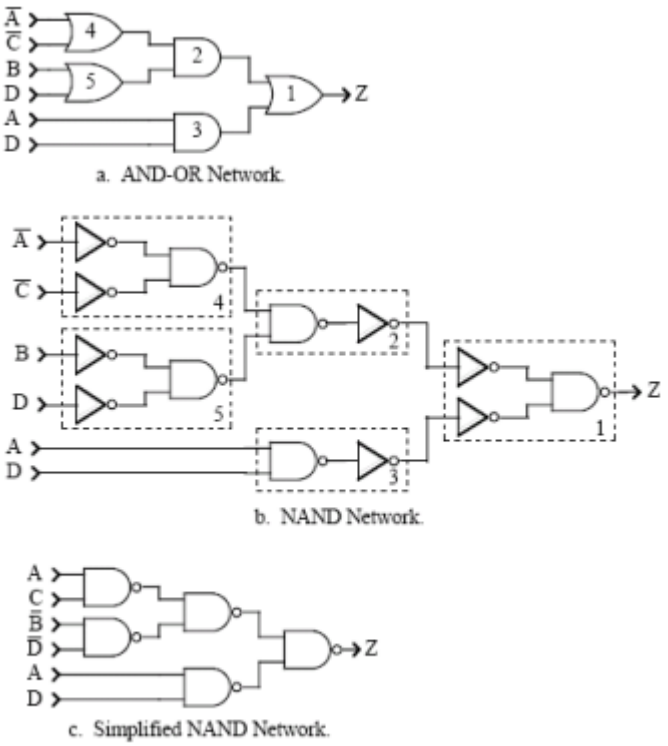


Рис. 2.11. Пример преобразования схемы в логический базис И-НЕ (пунктиром выделены фрагменты И-НЕ)

В заключение следует снова отметить, что традиционная минимизация логических функций, преследующая цель уменьшить количество вхождений переменных и количество операций не учитывает множества других критериев, как например минимизации задержки сигналов и т.п.