Лабораторная работа № 3

ПЛИС: сумматоры и мультиплексоры

Цель работы: получить практические навыки разработки и размещения схем сумматоров в ПЛИС, а также приобрести опыт использования мультиплексоров при вводе и отображении данных.

1. Устройство сумматоров и мультиплексоров

Сумматор — логический операционный узел, выполняющий арифметическое сложение двух чисел. В процессе подготовки арифметического сложения возникает необходимость в дополнительных действиях: учёт знаков чисел, выравнивание порядков слагаемых и тому подобное. Весь комплекс операций со слагаемыми осуществляется арифметико-логическими устройствами, ядром которых являются сумматоры.

Одноразрядные двоичные сумматоры по внутренней структуре, числу входов и выходов подразделяются следующим образом:

- **четверть сумматоры;**
- **полусумматоры;**
- **полные сумматоры.**

Четверть сумматор является простейшим двоичным суммирующим элементом. Он характеризуется двумя входами складываемых бит и одним выходным значением их арифметической суммы. Четверть сумматор имеет в таблице истинности в два раза меньше выходов и в два раза меньше строк, чем полный двоичный одноразрядный сумматор. На основе его таблицы истинности (рис. 1) можно записать следующие логические формулы:

a	b	S
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 1. Таблица истинности четверть сумматора

а). $S = \overline{a} \cdot b + a \cdot \overline{b} = a \cdot \overline{a} + \overline{a} \cdot b + a \cdot \overline{b} + b \cdot \overline{b} = a \cdot \left(\overline{a} + \overline{b}\right) + b \cdot \left(\overline{a} + \overline{b}\right) = a \cdot \overline{a \cdot b} + b \cdot \overline{a \cdot b}$

$$S = \overline{\overline{a \cdot \overline{a \cdot b}} \cdot \overline{b \cdot \overline{a \cdot b}}} = a \oplus b$$

Функцию $a \oplus b$ по другому называют: «Исключающее ИЛИ». Условное обозначение этого логического элемента приведено на рисунке 2.

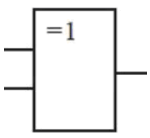


Рис. 2. Условное обозначение логического элемента «Исключающее ИЛИ»

Схема четверть сумматора, построенная с помощью элементов И-НЕ показана на рисунке 3-а.

b). $S = \overline{a} \cdot b + a \cdot \overline{b} = a \cdot \overline{a} + \overline{a} \cdot b + a \cdot \overline{b} + b \cdot \overline{b} = \overline{a} \cdot (a+b) + \overline{b} \cdot (a+b) = \overline{a + \overline{a+b}} + \overline{b + \overline{a+b}}$

Схема реализации четверть сумматора на элементах ИЛИ-НЕ показана на рисунке 3-б.

c). $S = \overline{a} \cdot b + a \cdot \overline{b} = a \cdot \overline{a} + \overline{a} \cdot b + a \cdot \overline{b} + b \cdot \overline{b} = \overline{a} \cdot (a+b) + \overline{b} \cdot (a+b) = (\overline{a} + \overline{b}) \cdot (a+b)$

$$S = \overline{a \cdot b} \cdot (a+b)$$

На рисунке 3-в показана схема четверть сумматора на элементах базиса (И, ИЛИ, НЕ).

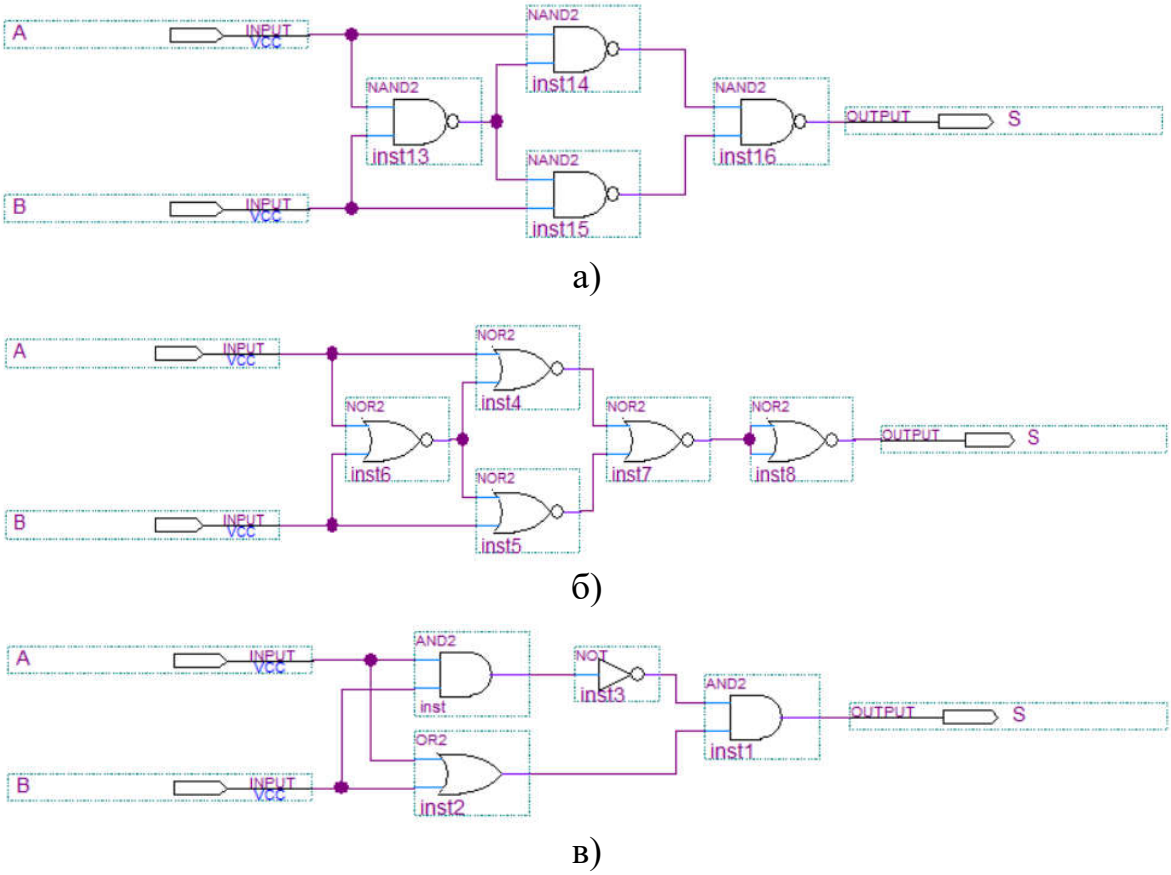


Рис.3. Схемы четверть сумматора в базисах Шеффера (а), Пирса (б), Буля (в)

Полусумматор характеризуется двумя входами складываемых бит и двумя выходами, на одном из которых реализуется арифметическая сумма, а на другом – перенос в следующий (более старший) разряд. Таблица истинности полусумматора на рисунке 4 позволяет записать формулу $P = f(a,b)$. Обозначением полусумматора служат буквы **HS** (half sum – полусумма).

a	b	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

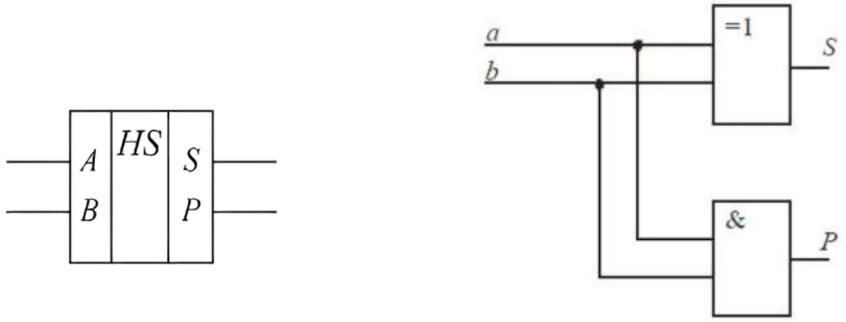
Рис. 4. Таблица истинности полусумматора

Формулы выходов полусумматора:

$$S = a \oplus b$$

$$P = a \cdot b$$

На рисунке 5 показано условное обозначение полусумматора и схема его реализации.



а) б)

Рис. 5. Условное обозначение (а) и схема (б) полусумматора

Полный одноразрядный двоичный сумматор имеет три входа: слагаемые a и b, перенос из младшего разряда p_{in}. Выходами полного сумматора как и полусумматора является сумма в текущем разряде S и перенос в следующий (более старший) разряд P. Таблица истинности полного сумматора представлена на рисунке 6.

a	b	p_{in}	S	P
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. 6. Таблица истинности полного сумматора

Логические формулы, описывающие работу полного сумматора в виде СДНФ, имеют следующий вид:

$$S = \overline{a} \cdot \overline{b} \cdot p_{in} + \overline{a} \cdot b \cdot \overline{p}_{in} + a \cdot \overline{b} \cdot \overline{p}_{in} + a \cdot b \cdot p_{in}$$

$$P = \overline{a} \cdot b \cdot p_{in} + a \cdot \overline{b} \cdot p_{in} + a \cdot b \cdot \overline{p}_{in} + a \cdot b \cdot p_{in}$$

Введем обозначения:

$$S_1 = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b, \quad \overline{S}_1 = \overline{a} \cdot \overline{b} + a \cdot b,$$

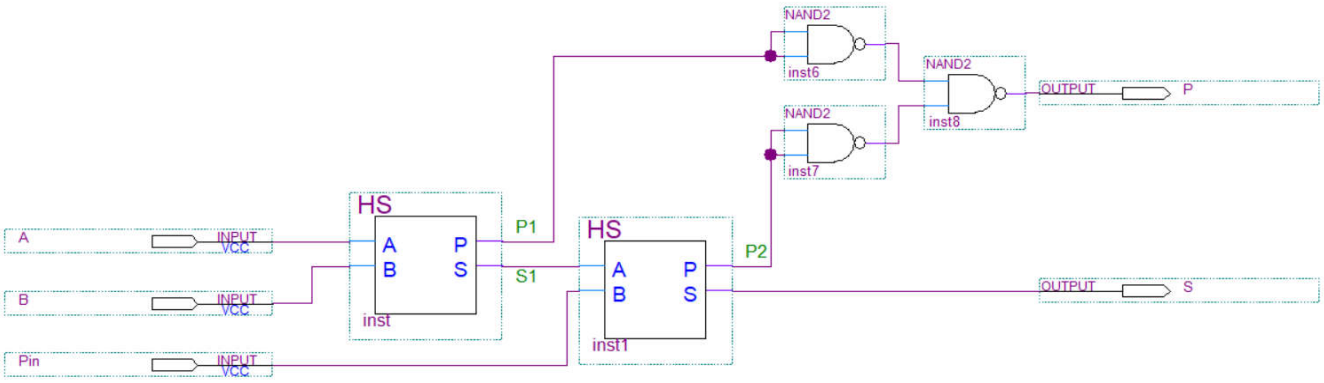
$$P_1 = a \cdot b, \quad P_2 = S_1 \cdot p_{in}$$

Тогда формулы полного сумматора можно записать в виде:

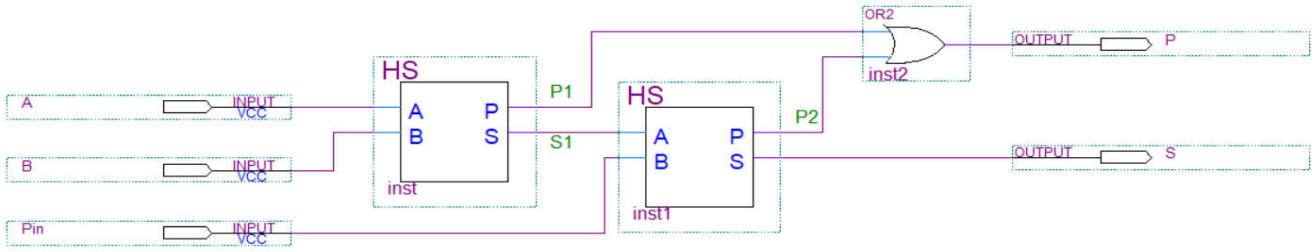
$$S = \left(\overline{a} \cdot b + a \cdot \overline{b}\right) \cdot \overline{p}_{in} + \left(\overline{a} \cdot \overline{b} + a \cdot b\right) \cdot p_{in} = S_1 \cdot \overline{p}_{in} + \overline{S}_1 \cdot p_{in} = S_1 \oplus p_{in} = a \oplus b \oplus p_{in}$$

$$P = a \cdot b + \left(\overline{a} \cdot b + a \cdot \overline{b}\right) \cdot p_{in} = a \cdot b + S_1 \cdot p_{in} = P_1 + P_2 = \overline{\overline{P_1} \cdot \overline{P_2}}$$

Полученные формулы показывают, что полный двоичный сумматор можно реализовать на двух полусумматорах и трех элемента И-НЕ (рис. 7-а), либо на двух полусумматорах и одном элементе ИЛИ (рис. 7-б).



а)



б)

Рис. 7. Схемы полного сумматора на двух полусумматорах

Условное обозначение полного сумматора в логических схемах приведено на рисунке 8.

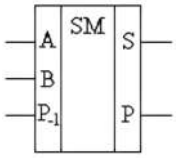


Рис. 8. Условное обозначение полного сумматора

Для построения многоразрядных сумматоров соединяют выход переноса (P) однобитного сумматора текущего разряда со входом переноса сумматора следующего разряда (рис. 9).

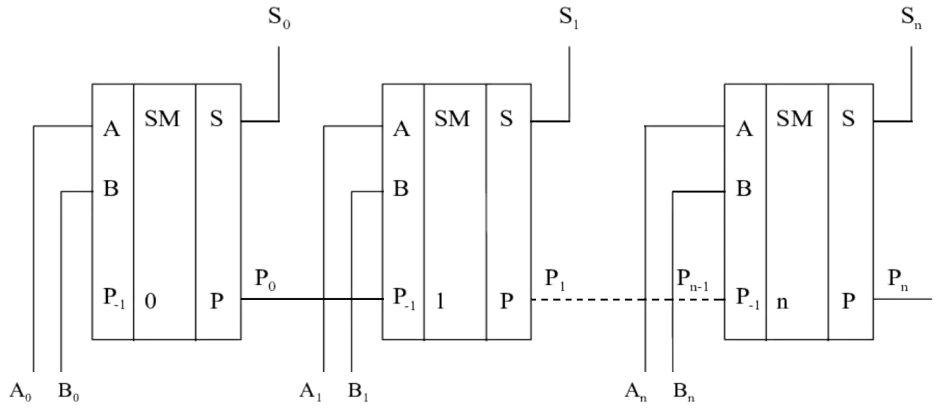


Рис. 9. Соединение элементов в многоразрядном сумматоре

Мультиплексор – это устройство, которое имеет несколько входов, и способно направлять сигнал с выбранного входа на выход. Простейшим мультиплексором является механический переключатель. В **аналоговых мультиплексорах** осуществляется электрическое соединение выбранного входа с выходом (соединение имеет минимальное электрическое сопротивление). В **цифровых мультиплексорах** отсутствует непосредственное соединение входа и выхода, но сигнал выхода является копией сигнала на выбранном входе. Базовым узлом цифровых мультиплексоров является логический элемент «И». Рассмотрим его таблицу истинности со следующими обозначениями: **DI** (Data Input) – информационный вход, **C** (Control) – управляющий вход и **DO** (Data Output) – информационный выход (рис. 10).

DI	C	DO
0	0	0
1	0	0
0	1	0
1	1	1

Рис. 10. Таблица истинности элемента «И», управляющего цифровым входом

Пока на управляющий вход **C** подан логический «0», сигнал на выходе **DO** также равен нулю и не зависит от сигнала информационного входа **DI**. Если же на управляющем входе уровень сигнала соответствует логической «1», то сигнал на выходе **DO**, повторяет сигнал информационного входа **DI**. Таким образом, элемент «И» в мультиплексорах играет роль электронного ключа. Остается только соединить несколько подобных ключей с помощью элемента «ИЛИ» и обеспечить выбор одного из них с помощью дешифратора, который получает двоичный адрес ключа и генерирует сигнал его активации. Схема принципиального устройства цифрового мультиплексора приведена на рисунке 11.

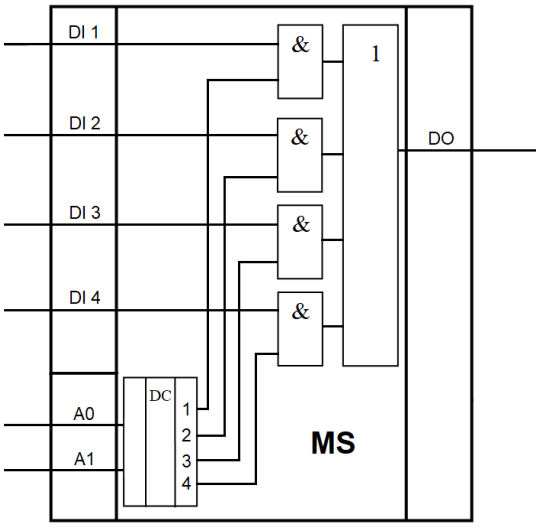


Рис. 11. Принципиальная схема цифрового мультиплексора

Цифровой мультиплексор имеет условное обозначение **MS** (Multi Selector).

Демультиплексор – это устройство, которое имеет один вход и способно направлять сигнал от него на выбранный выход. Аналоговые демультиплексоры функционально эквивалентны мультиплексорам, так как по открытому между входом и выходом каналу сигнал может распространяться в обеих направлениях. В следствие чего аналоговые мультиплексоры (демультиплексоры) называют **ключами** или **коммутаторами**, а для условного обозначения используют символ **MUX**. В цифровых системах выходной сигнал является логической функцией входного. Поэтому цифровой демультиплексор выполняет обратную функцию по отношению к мультиплексору. На рисунке 12 приведена схема принципиального устройства цифрового демультиплексора. Он имеет условное обозначение **DMS**.

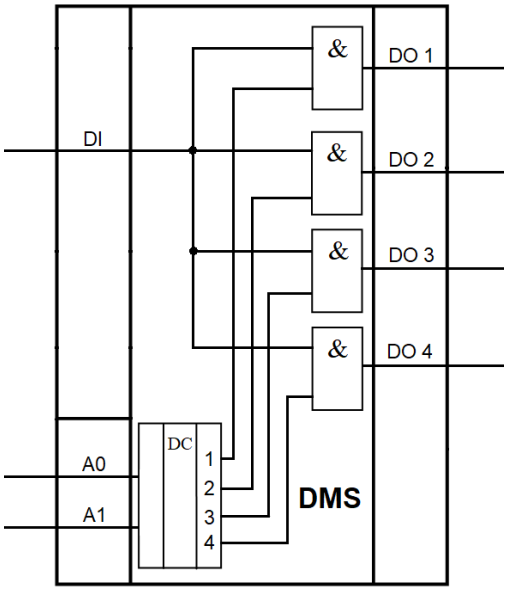


Рис. 12. Принципиальная схема цифрового демультиплексора

2. Задание

1. Создайте папку «LabFPGA(<фамилия на латинице>-<номер группы>)-3». Например, студент Иванов из группы 1191б должен создать папку: LabFPGA(Ivanov-1191)-3.
2. Внутри папки «LabFPGA…» создайте папки: «Task3_1», «Task3_2» и «Task3_3». В них далее следует размещать проекты среды Quartus Web Edition, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

2.1. Обучающая часть

Проект «Сложение и вычитание положительных чисел»: Построить арифметико-логическое устройство, способное складывать и вычитать двоичные положительные числа. Первым операндом, подаваемым на вход арифметического устройства должна быть константа $+7_{10}$, а значащие биты второго операнда должны определяться кнопками SW1 – SW3 лабораторного стенда. Кнопку SW4 следует использовать, чтобы задать вид операции. В случае отпущенной кнопки SW4 арифметическое устройство должно производить сложение операндов, а при

нажатой – вычитание второго операнда из первого. Результат арифметических действий должен отображаться на семисегментном индикаторе лабораторного стенда в виде шестнадцатеричного значения.

Построение схемы многоразрядного сумматора двоичных чисел

1. Запускаем приложение Quartus и строим новый проект в папке «Task3_1».
2. Создаем в проекте новую схему и в окне редактора «Block Diagram/Schematic» собираем четверть сумматор в булевом базисе, изображенный на рисунке 3-в. Логическая функция суммы любого одноразрядного сумматора реализуется операцией «Исключающее ИЛИ» с условным обозначением **XOR**. Даем имя файлу схемы в соответствие с шаблоном «XOR_<инициалы ФИО на латинице>3_1.bdf». Делаем файл главным в иерархии проекта и компилируем его. Строим в проекте файл для симуляции работы четверть сумматора, задаем входные сигналы в соответствие с его таблицей истинности и генерируем функциональную диаграмму сигналов для выхода S (рис. 13).

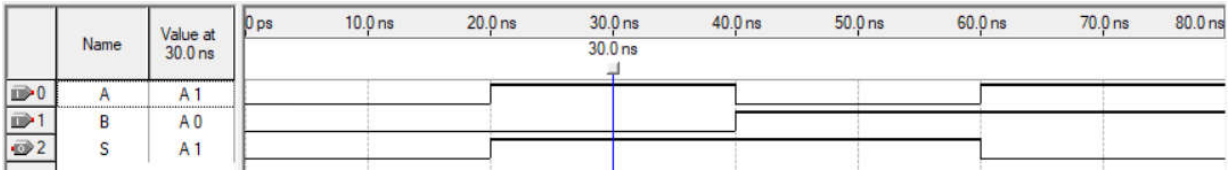


Рис. 13. Симуляция сигналов четверть сумматора

Сравнивая диаграмму сигналов с таблицей истинности четверть сумматора (рис. 1) убеждаемся в корректности разработанной схемы. Создаем в проекте Quartus модуль четверть сумматора с таким же названием как у файла схемы.

3. Создаем в проекте еще один файл типа «Block Diagram/Schematic». В окне редактора этого файла построим схему полусумматора. Для этого достаточно организовать выход переноса после элемента $(a \wedge b)$ из схемы четверть сумматора. Однако ранее построенный модуль XOR не подходит для этой цели, т.к. нужно подключиться к внутреннему элементу четверть сумматора. Поэтому копируем его схему целиком в окно редактора полусумматора и дополняем выходом переноса P, подключенного к логическому элементу $(a \wedge b)$ (рис. 14).

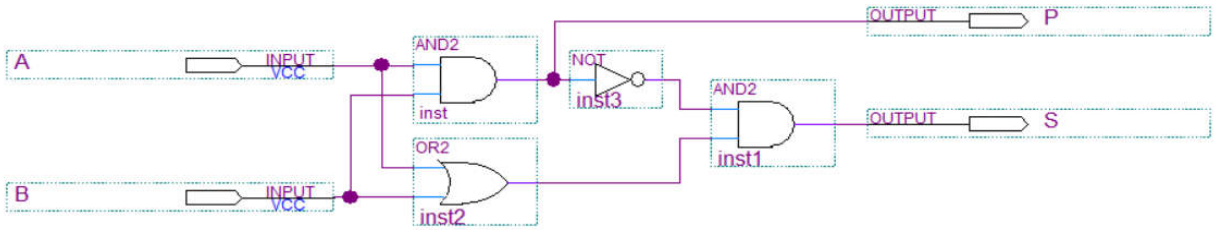


Рис. 14. Схема полусумматора в булевом базисе

Даем имя файлу схемы полусумматора в соответствие с шаблоном «HS_<инициалы ФИО на латинице>3_1.bdf». Делаем файл главным в иерархии проекта и компилируем его. Строим в проекте файл для

симуляции работы полусумматора, задаем входные сигналы в соответствие с его таблицей истинности и генерируем функциональные диаграммы сигналов для выходов S и P (рис. 15).

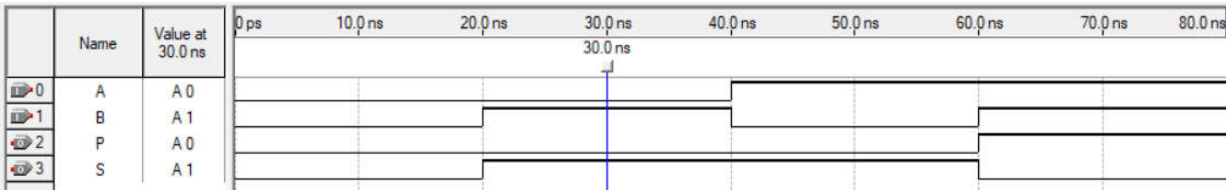


Рис. 15. Симуляция сигналов полусумматора

Сравнивая диаграммы с таблицей истинности полусумматора (рис. 4) убеждаемся в корректности разработанной схемы. Создаем в проекте Quartus модуль полусумматора с таким же названием как у файла схемы.

4. Создаем в проекте третий файл типа «Block Diagram/Schematic» и в окне его редактора строим схему полного одноразрядного сумматора в соответствие с рисунком 7-б. При этом используем модуль полусумматора, который был разработан ранее в п. 3. Даем имя файлу схемы в соответствие с шаблоном «SM_<инициалы ФИО на латинице>3_1.bdf». Делаем файл главным в иерархии проекта и компилируем его. Строим в проекте файл для симуляции работы полного одноразрядного сумматора, задаем входные сигналы в соответствие с его таблицей истинности и генерируем функциональные диаграммы сигналов для выходов S и P (рис. 16).

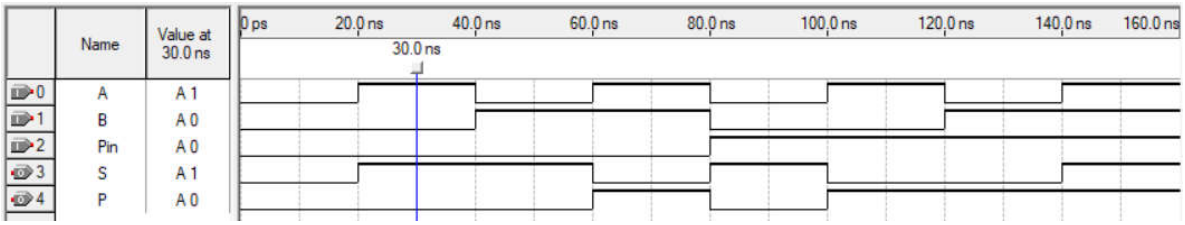


Рис. 16. Симуляция сигналов полного одноразрядного сумматора

Сравнивая диаграммы с таблицей истинности полного одноразрядного сумматора (рис. 6) убеждаемся в корректности разработанной схемы. Создаем в проекте Quartus модуль полного одноразрядного сумматора с таким же названием как у файла схемы. Замечаем, что в целом на реализацию полного одноразрядного сумматора потребовалось 9 логических элементов базиса Буля, а максимальное время задержки комбинационной схемы составило 12.189 нс. Таким образом, максимальная частота входного сигнала полного одноразрядного сумматора составляет порядка 40 МГц.

5. Теперь для решения поставленной задачи разработаем схему многоразрядного сумматора на основе модуля одноразрядного сумматора. Сначала определим количество необходимых разрядов сумматора/вычитателя в задаче. Максимальное значение модуля каждого операнда равно 7. Для представления этого числа в двоичном коде требуется 3 значащих разряда. Однако сумма максимальных значений операндов в двоичном коде потребует на 1 разряд больше. Кроме того, для реализации в АЛУ операций сложения и вычитания нужен дополнительные бит,

показывающий знак у операнда и результата операции. Он также принимает участие в операции сложения. Поэтому полное количество разрядов сумматора будет равно: количество значащих разрядов (3+1) + 1 знаковый бит = 5.

6. Создаем новый файл типа «Block Diagram/Schematic» для разработки схемы арифметико-логического устройства (АЛУ) на основе многоразрядного сумматора. Даем имя файлу схемы в соответствие с шаблоном «ALU_<инициалы ФИО на латинице>3_1.bdf». Для построения многоразрядного сумматора выход переноса текущего разряда нужно соединить со входом переноса следующего разряда. С учетом условий задания строим схему арифметико-логического устройства (АЛУ), включающую многоразрядный сумматор и выполняющую только операцию сложения (рис. 17).

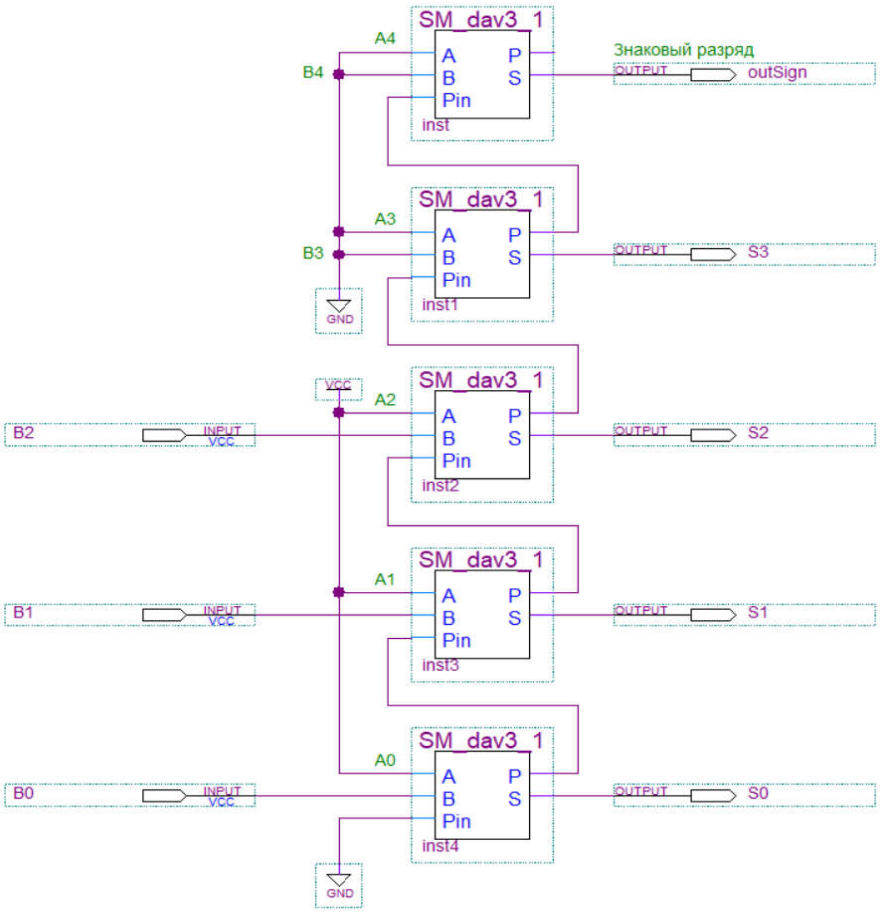


Рис. 17. АЛУ для сложения: $0111_2 + (0B2B1B0)_2$

Ядром АЛУ является пятиразрядный сумматор. Кроме сложения, АЛУ дополнительно выполняет следующие операции:

- Задает в виде константы первое слагаемое, биты которого подаются на соответствующие входы «А» многоразрядного сумматора. По условию задания первым операндом является число +7. Поэтому в двоичной системе счисления на входах: A0, A1 и A2 следует установить сигнал логической единицы, что реализовано подачей уровня питания на соответствующие входы одноразрядных сумматоров.

- Осуществляет выравнивание разрядности слагаемых до 4-х бит для получения корректного результата операции суммирования. Соответствующие биты слагаемых: A3 и B3, задаются в АЛУ равными нулю за счет подтяжки их к уровню «земли» (GND).
- Добавляет каждому операнду знаковый бит (A4 и B4) для представления их в прямом или дополнительном коде. В схеме, представленной на рисунке 17, выполняется только операция сложения. Поэтому оба операнда имеют знак «+», который кодируется значением 0 в знаковом бите числа, а операция сложения производится с операндами, представленными в прямом коде. С этой целью входы A4 и B4 подтянуты к уровню GND.
- На вход переноса нулевого разряда подается значение ноль.

7. Сохраняем файл схемы АЛУ, делаем его главным в иерархии проекта и компилируем. Затем с помощью симулятора тестируем работы АЛУ. В качестве входных данных используем числа от 0 до 7, подаваемые в виде бит двоичных чисел на соответствующие входы B0, B1, B2. Результаты симуляции представлены на рисунке 18.

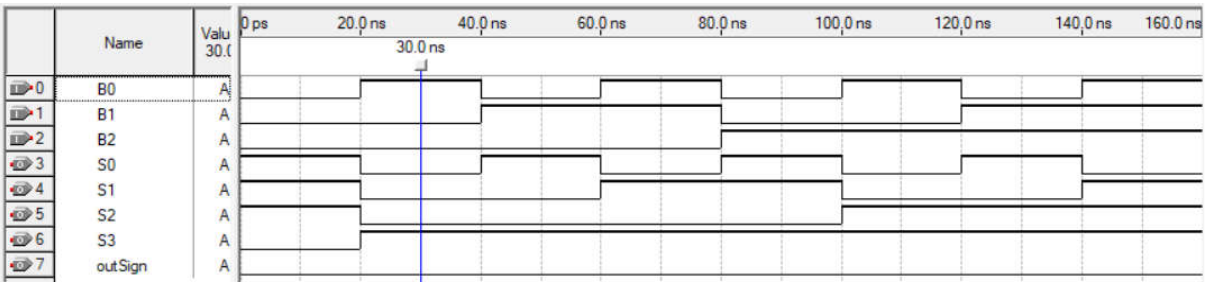


Рис. 18. Симуляция операции сложения в разработанном АЛУ

Прямой, обратный и дополнительный код числа

8. Получить разность двух положительных чисел x и y с помощью многоразрядного двоичного сумматора можно, если представить второе слагаемое в сумме отрицательным числом: $x - y = x + (-y)$. Для проведение такой операции нужно оба слагаемых записать в дополнительном коде. Алгоритм построения дополнительного кода включает следующие шаги:
   - Уровнять количество разрядов обеих слагаемых.
   - Записать оба слагаемых в прямом коде, поместив перед старшим разрядом двоичного числа знаковый бит и присвоить ему значение «0» в случае положительного слагаемого, либо «1» в случае отрицательного слагаемого.
   - Дополнительный код положительных чисел равен их прямому коду, а чтобы получить дополнительный код отрицательного числа в его прямом коде инвертируют все разряды, кроме знакового (обратный код), и получившееся значение увеличивают на единицу.

   После представления слагаемых в дополнительном коде выполняют побитовое сложение с переносом, включая знаковый разряд. В получившейся сумме анализируют знаковый бит. Если он равен нулю, то

результат положительный и представлен в прямом коде. Если в знаковом бите содержится единица, то результат отрицательный и представлен в дополнительном коде. Абсолютную величину, вычисленной суммы, можно определить после преобразования дополнительного кода в прямой и отбрасывания знакового бита. Для преобразования в прямой код у отрицательных чисел инвертируют разряды дополнительного кода, кроме знакового, и прибавляют единицу. Прямой код положительных чисел равен их дополнительному коду.

Пример 1: 6 – 3 = ?

Для двоичной записи числа $6_{10} = 110_2$ требуется 3 значащих разряда, а для записи числа $3_{10} = 11_2$ нужно 2 значащих разряда. В соответствие с алгоритмом после выравнивания разрядности слагаемых получим: $6_{10} = 110_2$ и $3_{10} = 011_2$. Для вычисления разности (6 – 3) второе слагаемое представляем в виде отрицательного числа, а результаты этапов построения дополнительного кода операндов занесем в таблицу (в кодах точкой отделен знаковый бит числа):

Число	Прямой код	Обратный код	Дополнительный код
$6_{10} = 110_2$	0.110_2	-	0.110_2
$-3_{10} = -011_2$	1.011_2	1.100_2	1.101_2

Далее выполняем сложение двоичных чисел в дополнительном коде, включая знаковый разряд:

$$\begin{array}{r} 0110 \\ +\ 1101 \\ \hline 0011 \end{array}$$

Перенос из знакового разряда суммы отбрасываем. Так как знаковый разряд суммы равен 0, то результат положительный и представлен в прямом коде. Значение суммы получаем из прямого кода числа, отбрасывая знаковый бит. Ответ: $011_2 = 11_2 = 3_{10}$.

Пример 2: 6 – 13 = ?

Для двоичной записи числа $6_{10} = 110_2$ требуется 3 значащих разряда, а для записи числа $13_{10} = 1101_2$ нужно 4 значащих разряда. В соответствие с алгоритмом после выравнивания разрядности слагаемых получим: $6_{10} = 0110_2$ и $13_{10} = 1101_2$. Для вычисления разности (6 – 13) второе слагаемое представляем в виде отрицательного числа, а результаты этапов построения дополнительного кода операндов занесем в таблицу (в кодах точкой отделен знаковый бит числа):

Число	Прямой код	Обратный код	Дополнительный код
$6_{10} = 0110_2$	0.0110_2	-	0.0110_2
$-13_{10} = -1101_2$	1.1101_2	1.0010_2	1.0011_2

Далее выполняем сложение двоичных чисел в дополнительном коде, включая знаковый разряд:

$$\begin{array}{r} 00110 \\ +\ 10011 \\ \hline 11001 \end{array}$$

Перенос из знакового разряда суммы отбрасываем. Так как знаковый бит суммы равен 1, то результат отрицательный и представлен в дополнительном коде. Для преобразования в прямой код инвертируем биты дополнительного кода, кроме знакового, и прибавляем единицу. В итоге прямой код суммы: 1.0111_2, а результат вычисления равен $–0111_2 = –7_{10}$. Ответ: $–111_2 = –7_{10}$.

9. Работа четверть сумматора основана на логической функции «Исключающее ИЛИ» (XOR). Таблица истинности операции XOR позволяет применять данный логический элемент в качестве **управляемого инвертора**. Используем в таблице следующие обозначения: **DI** (Data Input) – информационный вход, **C** (Control) – управляющий вход и **DO** (Data Output) – информационный выход (рис. 19).

DI	C	DO
0	0	0
1	0	1
0	1	1
1	1	0

Рис. 19. Таблица истинности операции «Исключающее ИЛИ»

Когда на управляющий вход **C** подано значение логического «0», то сигнал со входа **DI** передается на выход **DO** без инверсии. Если же на управляющем входе **C** установлена логическая «1», то логический элемент производит операцию отрицания и со входа на выход поступает инвертированный сигнал. Это свойство элементов XOR используют в АЛУ для преобразования чисел из прямой кода в обратный, и наоборот.

10. Добавим в АЛУ многоразрядный управляемый инвертор на основе модулей однобитных четверть сумматоров. В качестве управляющего сигнала для инвертора применим знак второго операнда. Если АЛУ производит сложение, то знак второго операнда положительный, а в знаковом разряде числа находится «0». Поэтому многоразрядный управляемый инвертор передаст исходные биты двоичного числа на вход сумматора без изменения, и сумматор будет проводить операцию с прямым кодом числа. Когда АЛУ должен выполнить операцию вычитания второго операнда из первого, то знак второго операнда отрицательный, а в знаковом разряде числа находится «1». Поэтому многоразрядный инвертор передаст на вход сумматора обратный код двоичного числа. Остается только добавить к нему единицу, чтобы образовался дополнительный код второго операнда. Реализуем преобразование обратного кода второго операнда в дополнительный внутри

сумматора. Для этого на вход переноса младшего разряда сумматора передадим сигнал знакового бита. Таким образом, при выполнении операции вычитания многоразрядный сумматор будет складывать первый операнд в дополнительном коде (т.к. для положительных числе прямой и дополнительный коды совпадают), второй операнд в обратном коде и единицу, поступающую через вход переносам суммирующего элемента младшего разряда. На рисунке 20 изображена схема АЛУ, которая способна выполнять как сложение, так и вычитание двух положительных чисел в соответствие с заданием.

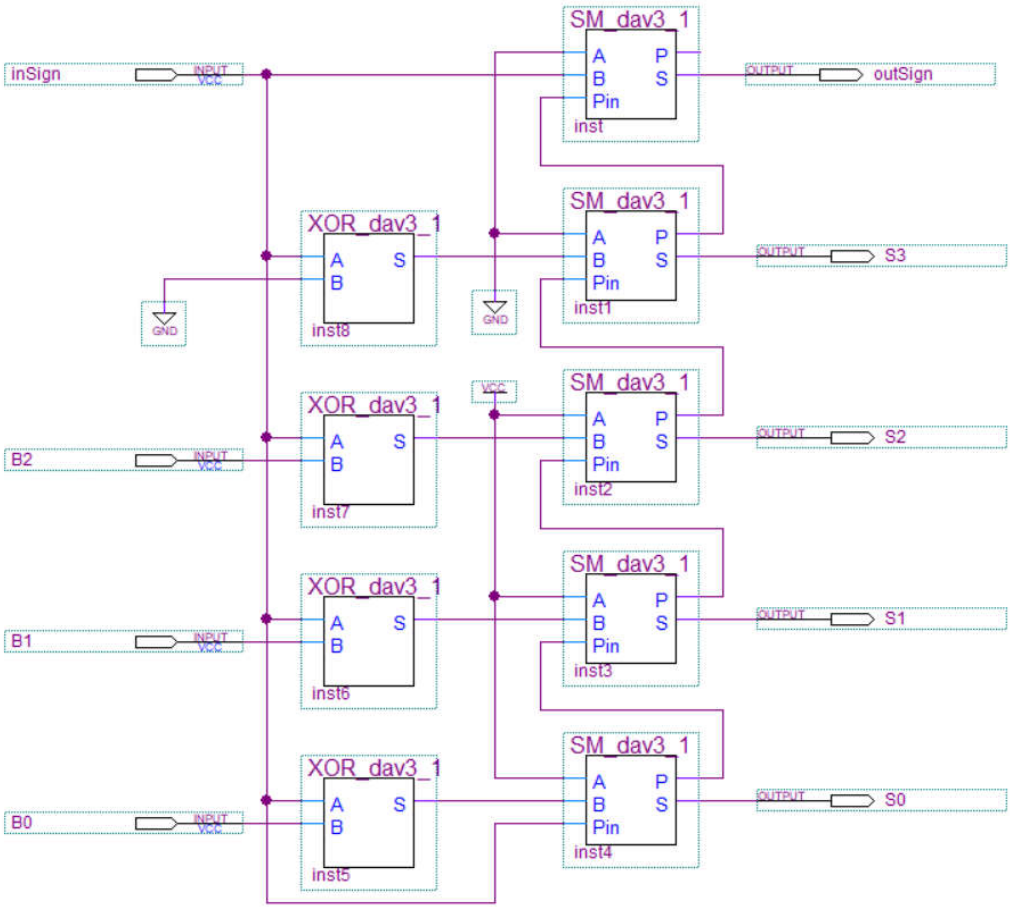


Рис. 20. Схема АЛУ для сложения и вычитания положительных чисел

Замечание. Когда требуется сложить два отрицательных числа, то использовать вход переноса младшего разряда сумматора уже не удастся. В таком случае нужно подавать на вход многоразрядного сумматора операнды в сразу дополнительном коде. Для этого на основе четверть сумматоров и полусумматоров строят специальный дешифратор, которые выполняет преобразование кода из прямого в дополнительный. Более того, алгоритм преобразования «дополнительный – прямой код» совпадает с алгоритмом «прямой – дополнительный код». Поэтому при возникновении на выходе сумматора отрицательного значения подобный преобразователь кода способен представить результат в прямом коде. В дешифраторе «прямой – дополнительный код» четверть сумматоры выполняют роль управляемых

инверторов, а на полусумматорах строиться схема добавления единицы к обратному коду числа.

11. Сохраняем файл с обновленным АЛУ, компилируем его и с помощью симулятора тестируем работу в процессе сложении и вычитании положительных чисел. В качестве входных данных используем числа от 0 до 7, подаваемые в виде бит двоичных чисел на соответствующие входы B0, B1, B2, и знак операции на входе inSign. Результаты симуляции представлены на рисунке 21.

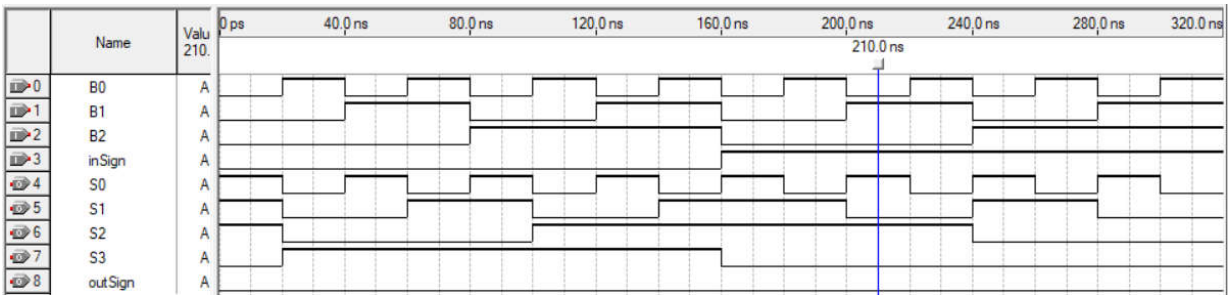


Рис. 21. Симуляция работы АЛУ для сложения и вычитания положительных чисел

12. Создаем в проекте Quartus модуль АЛУ для сложения и вычитания положительных чисел. Даем ему такое же название как у файла схемы.
13. Создаем в проекте Quartus новый файл типа «Block Diagram/Schematic» и, используя модули АЛУ и дешифратора шестнадцатеричных цифр в код семисегментного индикатора (разработан в третьем задании лабораторной работы «ПЛИС: шифраторы и дешифраторы»), строим схему сопряжения аппаратного обеспечения лабораторного стенда с разработанным арифметико-логическим устройством. Даем имя файлу схемы в соответствие с шаблоном «ALU_Stend_<инициалы ФИО на латинице>3_1.bdf». Используя «Assignment Editor» соединяем выводы микросхемы ПЛИС с выводами схемы сопряжения (рис. 22). Для отображения знака суммы задействуем на лабораторном стенде сегмент «DP» («точку») индикатора. «Точка» индикатора загорается в том случае, когда результат вычислений отрицательный. Символ шестнадцатеричной цифры выводим на крайний правый индикатор. Поэтому его управляющий сигнал подтягиваем к GND. Управляющие сигналы остальных индикаторов нужно подтянуть к $V_{пит}$, чтобы гарантировать их полное гашение. Не забываем, что не использованные выводы микросхемы ПЛИС следует перевести в состояние с высоким сопротивлением. Делаем файл схемы сопряжения главным в иерархии проекта и компилируем его. Наконец, с помощью меню «Tools - Programmer» загружаем разработанную логическую схему в ПЛИС.
14. Тестируем работу АЛУ на стенде. Подготавливаем отчет о выполнении первого задания в соответствие с требованиями раздела 3 настоящей лабораторной работы.

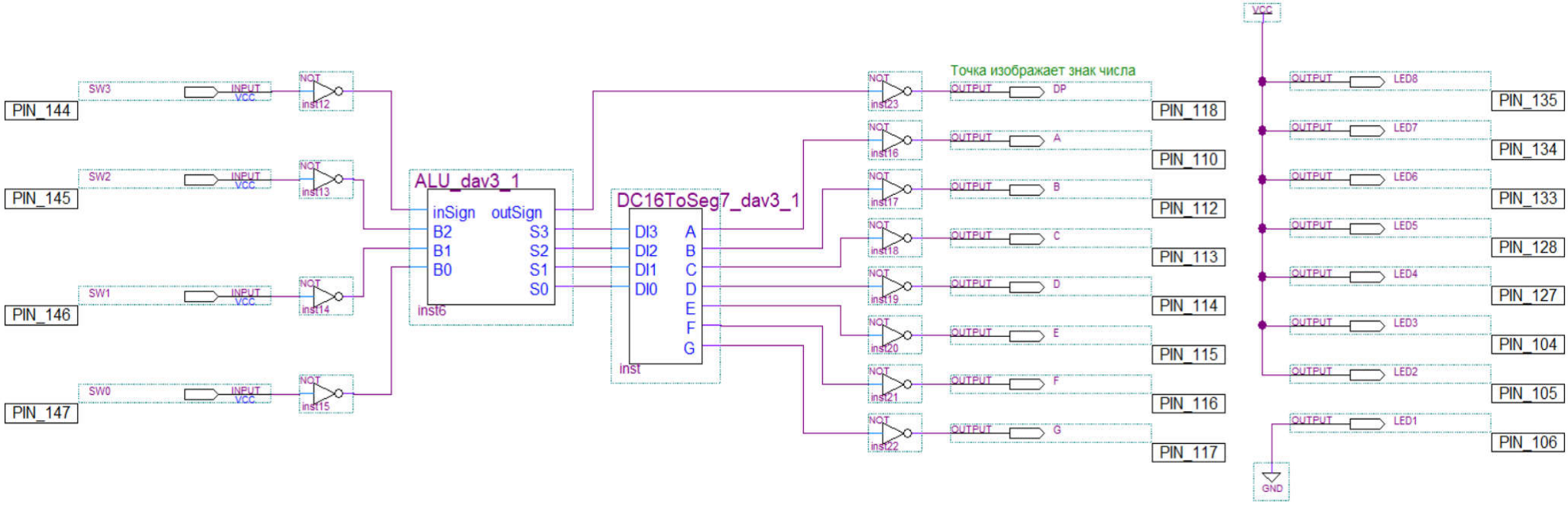


Рис. 22. Схема сопряжения АЛУ и дешифратора результата вычислений
с аппаратным обеспечением лабораторного стенд

2.2. Самостоятельная работа

Задача: Разработать АЛУ для сложения положительных и отрицательных двоичных чисел. Кнопки SW1 и SW2 лабораторного стенда должны определять знаки 1 и 2 операнда соответственно. Модуль первого операнда должен быть постоянен и равен 5_{10}. Модуль второго операнда в двоичной системе счисления должен задаваться с использованием кнопок лабораторного стенда SW3 и SW4 следующим образом: $((SW3)1(SW4))_2$. Прямой код результата вычислений должен отображаться семисегментным индикатором. Светящуюся «точку» индикатора следует использовать для изображения знака «–». Базис для построения комбинационных схем выбирается студентом произвольно.

1. Разработать модуль дешифратора, выполняющий преобразование кода числа из прямого в дополнительный и обратно. При построении дешифратора можно использовать только четверть сумматоры и полусумматоры.
2. Использовать модули дешифратора «прямой – дополнительный код» для представления операндов на входе многоразрядного сумматора в коде, который соответствует их знаку.
3. Использовать модуль дешифратора «прямой – дополнительный код» для представления результата вычислений в прямом коде.
4. Использовать дешифратор шестнадцатеричных цифр в код семисегментного индикатора (разработка 3 задания лабораторной работы «ПЛИС: шифраторы и дешифраторы») для отображения прямого кода результата вычислений на лабораторном стенде.
5. Разработать схему сопряжения АЛУ с аппаратным обеспечением лабораторного стенда.
6. Протестировать АЛУ на лабораторном стенде и продемонстрировать работоспособность схемы преподавателю.
7. Подготовить отчет о выполнении задания в соответствие с требованиями раздела 3 настоящей лабораторной работы.

2.3. Индивидуальное задание

Задача: Разработать многоразрядный мультиплексор для селективного отображения на семисегментном индикаторе прямого кода слагаемых и результата вычислений. Кнопки SW1 и SW2 лабораторного стенда должны определять знаки 1 и 2 слагаемых соответственно. Модули операндов АЛУ определены вариантом индивидуального задания, представленного в **Приложении 1**. Кнопки SW3 и SW4 лабораторного стенда должны использоваться для выбора отображаемого значения: первый операнд, либо второй операнд, либо результат вычислений. Схема кодирования выбора приведена в варианте индивидуального задания. Базис для построения комбинационных схем выбирается студентом произвольно.

1. Разработать модуль многоразрядного мультиплексора и интегрировать его с АЛУ, выполняющего сложение положительных и отрицательных чисел. Мультиплексор должен в соответствие с кодом, генерируемым внешним

шифратором, направлять на вход дешифратора одно из двоичных значений операндов или результата вычислений. Входные данные дешифратора должны быть представлены в прямом коде. Дешифратор преобразует шестнадцатеричные цифры в код семисегментного индикатора.

2. Разработать модуль шифратора, который в соответствие с вариантом индивидуального задания преобразует сочетания нажатий кнопок SW3 и SW4 в код для выбора значения, отображаемого на семисегментном индикаторе.
3. Использовать дешифратор шестнадцатеричных цифр в код семисегментного индикатора (разработка 3 задания лабораторной работы «ПЛИС: шифраторы и дешифраторы») для отображения прямого кода чисел на лабораторном стенде.
4. Разработать схему сопряжения специализированного АЛУ, включающего мультиплексор, шифратор выбора отображаемого значения и дешифратор семисегментного индикатора, с аппаратным обеспечением лабораторного стенда.
5. Протестировать специализированное АЛУ на лабораторном стенде и продемонстрировать его работоспособность преподавателю.
6. Подготовить отчет о выполнении задания в соответствие с требованиями раздела 3 настоящей лабораторной работы.

3. Подготовка отчета, представление и оценка работы

Структура отчета

1. Для каждой части задания требуется создать отдельный отчет. Файл отчета должен именоваться также как файл соответствующего проекта Quartus и иметь расширение «.doc». Отчет по части задания следует размещать в папке проекта (Task3_1, Task3_2 или Task3_3).
2. Для каждого разрабатываемого в проекте модуля и схемы сопряжения с аппаратным обеспечением лабораторного стенда требуется включить в отчет описание со следующий структурой:
   - В разделе «Дано» следует привести исходную таблицу входных и выходных сигналов разрабатываемого модуля и базис для построения его логической схемы.
   - В раздел «Решение» нужно поместить:
     a. карты Карно с обозначенными группами покрытия;
     b. формулы минимальных нормальных форм логических функций. Обосновать выбор дизъюнктивной или конъюнктивной формы в соответствие с базисом;
     c. изображение окна приложения Quartus со схемой разрабатываемого модуля;
     d. изображение окна «Simulation Report»;
     e. результаты симуляции работы модуля, представленные в виде таблицы со следующей структурой:

t, ns	Аргументы функции				Значение функции		
	x_1	x_2	x_3	…	f_0	f_1	…

f. изображение символа модуля в библиотеке элементов проекта Quartus, включающего название самого модуля и названия его входов и выходов.

- В разделе «Заключение» следует сделать вывод об особенностях реализации схемы, обосновать работоспособность созданного устройства и указать его быстродействие.

Представление и защита работы

3. Каждая часть задания лабораторной работы может быть представлена отдельно. В ходе представления студент должен дать пояснения по выводу формул МНФ, продемонстрировать в приложении Quartus схему логической функции или модуля, навыки работы с симулятором сигналов и средствами программирования ПЛИС.

4. Преподаватель оценивает знания по оптимизации логических функций на основе карт Карно и навыки разработки и симуляции работы схем в приложении Quartus. Студент должен быть готов ответить на дополнительные вопросы преподавателя, связанные с построением проекта Quartus, в рамках материалов лабораторной работы.

5. После получения оценки за представленные проекты лабораторной работы их вместе с отчетами следует загрузить на сайт Eluniver. Отчет по каждой части задания оценивается отдельно.

6. После представления обучающей и самостоятельной частей работы студент получает право ответить на 2 контрольных вопроса из обозначенного в **разделе 4** списка.

Структура оценки лабораторной работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
	Итого:	**100**

4. Контрольные вопросы

1. Какие задачи можно решать с помощью сумматора?
2. В чем состоит отличие сумматора от АЛУ? Приведите пример.
3. Что обозначает понятие «операнд»?
4. Зачем нужно выравнивание разрядности операндов и результата в АЛУ?
5. Какая логическая функция является простейшим суммирующим элементом?
6. Как называется и обозначается простейший суммирующий элемент?
7. Какова таблица истинности простейшего суммирующего элемента?
8. В чем состоит отличие между четверть и полусумматором?
9. Можно ли без дополнительных логических элементов четверть сумматор преобразовать в полусумматор?
10. В каком базисе схема полусумматор имеет наименьшее количество логических элементов?
11. Сколько строк содержит таблица истинности полусумматора?
12. Можно ли построить многоразрядный сумматор только на основе полусумматоров?
13. Как называется логическая операция, обозначаемая «XOR»?
14. Какой логический элемент способен выполнять функцию управляемого инвертора?
15. Является ли равенство верным: $a \oplus b \oplus c = c \oplus b \oplus a$? Поясните.
16. Какая логическая функция используется для формирования сигнала переноса в полусумматоре?
17. Какая логическая функция определяет сигнал переноса в одноразрядном полном сумматоре, построенном на полусумматорах?
18. Какие структурные элементы можно выделить в схеме цифрового мультиплексора?
19. В чем заключается различие между мультиплексором и демультиплексором?
20. Почему аналоговый мультиплексор может выполнять роль демультиплексора?
21. Какое обозначение имеет аналоговый демультиплексор?
22. Какое количество входов и выходов имеет цифровой демультиплексор с трехразрядной адресной шиной?
23. Почему для построения управляемого инвертора можно использовать четверть сумматор?
24. Как с помощью сумматора можно выполнять операцию вычитания?
25. Каков алгоритм преобразования дополнительного кода в прямой?
26. Код 100101_2 является обратным для 011010_2?
27. Является ли код 011001 прямым кодом числа -25? Поясните.
28. Можно ли использовать преобразователь «прямой – дополнительный код» для получение значений на выходе АЛУ в прямом коде? Докажите.
29. Если требуется складывать числа из диапазонов: [-6,9] и [-8,12], то какова должна быть разрядность операндов и суммы в АЛУ?
30. Если требуется вычитать числа из диапазонов: [-8, 11] и [-18,0], то какова должна быть разрядность операндов и разности в АЛУ?

Приложение 1. Варианты индивидуальных заданий

№ варианта	Модуль десятичного значения		Схема кодирования выбора		
	Операнд 1	Операнд 2	Код выбора	Сочетания нажатий (SW3, SW4)	Отображаемая величина
1.	3	6	0	(1, 0)	Операнд 1
			1	(0, 1)	Операнд 2
			3	(0, 0), (1, 1)	Результат вычислений
2.	2	7	0	(0, 0)	Операнд 1
			2	(0, 1), (1, 1)	Операнд 2
			3	(1, 0)	Результат вычислений
3.	1	5	1	(0, 0), (1, 1)	Операнд 1
			2	(0, 1)	Операнд 2
			3	(1, 0)	Результат вычислений
4.	4	6	3	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
5.	5	7	1	(1, 1)	Операнд 1
			0	(0, 0), (0, 1)	Операнд 2
			3	(1, 0)	Результат вычислений
6.	1	6	1	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
7.	2	4	3	(0, 0)	Операнд 1
			1	(0, 1), (1, 1)	Операнд 2
			2	(1, 0)	Результат вычислений
8.	5	4	2	(0, 1)	Операнд 1
			3	(1, 0)	Операнд 2
			0	(0, 0), (1, 1)	Результат вычислений
9.	7	3	3	(1, 0)	Операнд 1
			0	(0, 0), (1, 1)	Операнд 2
			1	(0, 1)	Результат вычислений
10.	6	5	1	(1, 1)	Операнд 1
			3	(0, 0), (1, 0)	Операнд 2
			0	(0, 1)	Результат вычислений
11.	4	7	2	(1, 0), (0, 1)	Операнд 1
			1	(0, 0)	Операнд 2
			3	(1, 1)	Результат вычислений
12.	5	6	1	(1, 1), (0, 1)	Операнд 1
			3	(1, 0)	Операнд 2
			2	(0, 0)	Результат вычислений
13.	7	4	2	(1, 0)	Операнд 1
			0	(0, 0)	Операнд 2
			3	(1, 1), (0, 1)	Результат вычислений
14.	1	6	2	(0, 1)	Операнд 1
			3	(1, 1)	Операнд 2
			0	(1, 1), (0, 0)	Результат вычислений
15.	3	5	1	(0, 1)	Операнд 1
			3	(0, 0), (1, 0)	Операнд 2
			2	(1, 1)	Результат вычислений
16.	2	7	2	(0, 1)	Операнд 1
			1	(1, 1)	Операнд 2
			0	(1, 1), (0, 0)	Результат вычислений

17.	5	6	1	(0, 1)	Операнд 1
			3	(0, 0), (1, 0)	Операнд 2
			0	(1, 1)	Результат вычислений
18.	4	5	3	(1, 0)	Операнд 1
			0	(0, 0), (1, 1)	Операнд 2
			1	(0, 1)	Результат вычислений
19.	6	3	2	(0, 1)	Операнд 1
			3	(1, 0)	Операнд 2
			0	(0, 0), (1, 1)	Результат вычислений
20.	4	6	3	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
21.	3	5	1	(0, 1)	Операнд 1
			3	(0, 0), (1, 0)	Операнд 2
			2	(1, 1)	Результат вычислений
22.	7	4	3	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
23.	1	6	2	(0, 1)	Операнд 1
			3	(1, 1)	Операнд 2
			0	(1, 1), (0, 0)	Результат вычислений
24.	7	4	1	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
25.	4	6	3	(0, 0)	Операнд 1
			1	(0, 1), (1, 1)	Операнд 2
			2	(1, 0)	Результат вычислений
26.	7	7	0	(1, 0)	Операнд 1
			1	(0, 1)	Операнд 2
			3	(0, 0), (1, 1)	Результат вычислений
27.	5	3	3	(0, 0), (0, 1)	Операнд 1
			2	(1, 1)	Операнд 2
			0	(1, 0)	Результат вычислений
28.	2	4	1	(1, 1)	Операнд 1
			0	(0, 0), (0, 1)	Операнд 2
			3	(1, 0)	Результат вычислений
29.	6	5	2	(0, 1)	Операнд 1
			1	(1, 1)	Операнд 2
			0	(1, 1), (0, 0)	Результат вычислений
30.	5	5	3	(1, 0)	Операнд 1
			0	(0, 0), (1, 1)	Операнд 2
			1	(0, 1)	Результат вычислений