МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

ОРГАНИЗАЦИЯ АРИФМЕТИКО-ЛОГИЧЕСКИХ УСТРОЙСТВ ЭВМ

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Самарский государственный аэрокосмический университет имени академика С.П. Королева (национальный исследовательский университет)» в качестве методических указаний

С а м а р а
Издательство СГАУ
2 0 1 3

УДК 004 (075)
ББК 32.97я7

Составители: ***О.А. Заякин, В.П. Павлов***

Рецензент д-р техн. наук, проф. П. К. Л а н г е

Организация арифметико-логических устройств ЭВМ [Электронный ресурс]: метод. указания к лабораторным работам по курсу Б3.Б.2 «ЭВМ и периферийные устройства» направления 230100.62 «Информатика и вычислительная техника» / сост. *О. А. Заякин, В. П. Павлов.* – Самара: Изд-во Самар. гос. аэрокос. ун-та, 2013. – 1 электрон. опт. диск (CD ROM).

В методических указаниях изучается структурная организация основной составляющей процессора ЭВМ – его арифметико-логического устройства.

Для изучения использованы графический редактор и эмулятор из системы автоматизированного проектирования MAX+PLUS II фирмы «Altera», разработанной ею для программирования собственных ПЛИС.

Предназначены для студентов, обучающихся по направлению «Информатика и вычислительная техника», а также по специальности «Автоматизированные системы обработки данных и управления». Могут быть полезны студентам, обучающимся по другим специальностям, связанным с информационными технологиями.

Работа подготовлена на кафедре информационных систем и технологий Самарского государственного аэрокосмического университета.

УДК 004 (075)
ББК 32.97я7

СВЕДЕНИЯ О СОСТАВИТЕЛЯХ

Заякин Олег Александрович – кандидат технических наук, научный сотрудник лаборатории моделирования и автоматизации лазерных систем Самарского филиала Физического института РАН (СФ ФИАН), доцент кафедры информационных систем и технологий (ИСТ) Самарского государственного аэрокосмического университета (СГАУ). Область научных интересов: лазерные измерения геометрических величин, когерентная оптика, жидкокристаллическая адаптивная оптика.

Результаты его работы нашли отражение в 40 научных работах, в том числе в патенте на изобретение, выступлениях более чем на 10 международных и региональных конференциях и симпозиумах.

Являлся руководителем более 40 дипломных проектов и работ.

Павлов Владимир Павлович – кандидат технических наук, доцент СГАУ. Область научных интересов: организация структур и вычислительных процессов в ЭВМ; научное направление: оценка производительности вычислительных систем; учебные курсы: организация ЭВМ, периферийные устройства систем автоматизации, устройства хранения информации.

Количество публикаций – более 60, из них одно авторское свидетельство. Участник многих международных, всесоюзных, а затем всероссийских, а также региональных конференций и симпозиумов.

Его школу прошли многие студенты и аспиранты.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АЛО – арифметико-логическая микрооперация;
АЛУ – арифметико-логическое устройство;
АМК – адрес микрокоманды;
БИС – большая интегральная микросхема;
БП – безусловный переход;
ВМ – вычислительная машина;
Г – закодированный граф микропрограмм;
ГТИ – генератор тактовых импульсов;
ЕП – естественный переход (то есть, по адресу следующей по порядку микрокоманды);
ЖЛ – жесткая логика, названа в противоположность программируемой, «мягкой» (soft) логике;
ЗУ – запоминающее устройство;
ЛУ – логическое условие;
ИСТ – кафедра информационных систем и технологий СГАУ;
КС – комбинационный сумматор;
КСЧ – комбинационный счетчик;
МК – микрокоманда;
МО – микрооперация;
МП – микропрограмма;
НЛУ – номер логического условия;
ОА – операционный автомат;
ОЗУ – оперативное запоминающее устройство;
ОПУ – комплекс операционных устройств;
ОС – операционная система;
ОУ – операционное устройство;
ПЗУ – постоянное запоминающее устройство (то есть, без возможности перезаписи);
ПЛ – программируемая логика;
ПЛИС – программируемая логическая интегральная схема;

ПМП – память микропрограммы;
ПНА – преобразователь начального адреса;
ПО – программное обеспечение;
РА – регистр адреса микрокоманды;
РАН – Российская академия наук;
РМК – регистр микрокоманды;
РОН – регистр общего назначения;
САПР – система автоматизированного проектирования;
СГАУ – федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Самарский государственный аэрокосмический университет имени академика С. П. Королёва (национальный исследовательский университет)»
СЛУ – список (множество) логических условий;
СМК – счетчик микрокоманд;
СМО – список (множество) микроопераций;
СФАМ – схема формирования адреса следующей микрокоманды;
СФ ФИАН – Самарский филиал федерального государственного бюджетного учреждения науки Физического института им. П. Н. Лебедева Российской академии наук;
ТИ – тактовый импульс;
УА – управляющий автомат;
УА с ПЛ – управляющий автомат с программируемой логикой;
УКС – универсальный комбинационный сумматор;
УП – условный переход;
УС – указатель стека;
ФК – формирователь кодов;
ФУС – формирователь управляющих сигналов;
ЦУУ – центральное устройство управления;
ЭВМ – электронная вычислительная машина;
ALO – арифметико-логическая операция;
BR – буферный регистр;
CISC – Complex Instruction Set Computer = архитектура с полным набором команд;
C – «Carry» - значение бита переноса из старшего разряда сумматора при выполнении результатов сложения или вычитания;
CF – флаг переноса;
FLU – формирователь логических условий;
DMC – управляющий демультиплексор;
I-автомат – ОА с максимальными характеристиками (с максимальной производительностью и максимальными затратами оборудования);
IBM PC – International Business Machines Personal Computer;
IM-автомат – ОА с промежуточными (между минимальными и максимальными) значениями характеристик между I- и M-автоматами;
M-автомат – ОА с минимальной производительностью (одна МО за один такт);
MAX+PLUS II - Multiple Array MatriX Programmeble Logic User System = Пользовательская Система Программирования Логики упорядоченных структур)

MS – мультиплексор;
MSA – мультиплексор A;
MSA – мультиплексор адреса;
MSB – мультиплексор B;
MSC – управляющий мультиплексор;
MSF – мультиплексор флагов;
MSL – мультиплексор левого сдвига;
MSR – мультиплексор правого сдвига;
NOP – «no operation» - пустой такт;
OVR – «Overflow» - признак (флаг, бит) переполнения, значение равно единице, если результат арифметической МО переполняет разрядную сетку, и нулю в противном случае;
OF – флаг переполнения, устанавливается в единицу, если в результате сдвига изменился знаковый (старший) разряд результата;
P – поле микрокоманды, указывающее тип этой микрокоманды;
RF – регистр флагов;
RISC – Reduced Instruction Set Computer = архитектура с сокращенным набором команд;
RON – регистр общего назначения;
S – «Storage» - регистр (обозначение на схеме), или его содержимое;
S – «Sign» - значение старшего разряда результата, при обработке чисел со знаком трактуется как знак результата;
SH – «Shift» - сдвигатель, то есть регистр сдвига;
SL – вход сдвигателя, используемый при левом сдвиге;
SM – сумматор;
SR – вход сумматора, используемый при правом сдвиге;
T – продолжительность такта работы цифрового устройства;
TC – счетный триггер;
TT – счетный триггер;
UKS – универсальный комбинационный сумматор;
W – «Write» - сигнал записи;
Z – «Zero» - признак (флаг, бит) нулевого результата.

ПРЕДИСЛОВИЕ

Данные методические указания составлены в соответствии с учебным планом СГАУ подготовки инженеров 090303.65-2012-О-П-5г00м (для набора студентов в 2012 г. – *примечание авторов*) для изучения дисциплины С3.Б.3 (номер дисциплины в плане – *примечание авторов*) «Организация ЭВМ и вычислительных систем» по специальности 090303.65 «Автоматизированные системы обработки данных и управления» (АСОИУ) и в соответствии с учебным планом СГАУ бакалаврской подготовки 230100.4.62-2012-О-П-4г00м для изучения дисциплины Б3.Б.2 «ЭВМ и периферийные устройства», относящейся к направлению 230100.62 «Информатика и вычислительная техника» (ИВТ).

В соответствии с рабочими программами названных выше дисциплин, описанные здесь лабораторные работы рассчитаны на один семестр, в течение которого студентам отводится 16 академических часов практических занятий и 12 часов самостоятельной работы.

Методические указания к лабораторным работам посвящены изучению структурной организации арифметико-логических устройств (АЛУ) электронно-вычислительных машин (ЭВМ). Изучение этих вопросов необходимо для понимания работы вычислительных систем. В брошюре приведено описание пяти лабораторных работ.

Методические указания дают студентам знания принципов устройства ядра современных ЭВМ, а также дают им типичные примеры практической реализации этих принципов на примере проектирования АЛУ – основного узла микропроцессора.

Рассматривается проектирование двухтактного конвейерного АЛУ на основе принципа микропрограммного управления, применяющегося сейчас практически во всех цифровых вычислительных устройствах. Используемый виртуальный макет АЛУ предполагает двухуровневое управление процессом вычислений, также широко применяемое в современных ЭВМ и контроллерах.

Обсуждаются различные способы кодирования микрокоманд, синхронизация комбинационных схем с обычно более медленными оперативными запоминающими устройствами (ОЗУ).

Для изучения использованы графический редактор и эмулятор из системы автоматизированного проектирования MAX+PLUS II фирмы «Altera», разработанной ею для программирования собственных ПЛИС. Это позволяет не только изучить проект и даже самостоятельно его создать, но и увидеть «изнутри» его работу, изучить реакцию системы во времени на множество факторов, которые заранее трудно учесть.

В лабораторных работах использованы проекты, разработанные в среде MAX+PLUS II студентами и преподавателями СГАУ. Они прошли успешную апробацию в образовательном процессе СГАУ – в курсовом и дипломном проектировании, а также в практических занятиях и лабораторных работах.

Данные методические указания к лабораторным работам также могут быть полезны студентам, обучающимся по специальностям, связанным с информационными технологиями.

Изучая описанные здесь лабораторные работы, студенты решают следующие основные задачи:

– ознакомление с принципами организации АЛУ;

– изучение основ работы с современными средствами проектирования цифровых вычислительных систем;

– изучение работы АЛУ.

Данный цикл лабораторных работ, совместно с двумя другими лабораторными работами, методические указания к которым издаются отдельно (это лабораторная работа, посвященная функциональной организации ЭВМ, на примере системы прерываний компьютеров IBM PC, а также лабораторная работа, посвященная системе ввода/вывода компьютеров этого семейства), образует практическую часть указанных выше курсов по специальности АСОИУ и направлению ИВТ.

Отметим, что вместе с другими названными здесь лабораторными работами практическая часть занимает несколько больший объем, в зависимости от конкретных учебных дисциплин. Так, например, согласно рабочей программе по курсу «ЭВМ и периферийные устройства» учебного плана 230100.4.62-2012-О-П-4г00м, на очные занятия отводятся 54 академических часа, и столько же - на самостоятельную работу.

При сокращенном объеме обучения по данным методическим указаниям можно опустить последнюю работу, а также совместить работу по ознакомлению с MAX+PLUS II с работой, посвященной знакомству с макетом АЛУ и принципами его действия.

Все использованные в данных работах материалы взяты из открытых источников и использованы (а также предназначены для использования в дальнейшем) только в учебных и образовательных целях.

Этот труд впервые выходит в тираж, хотя в практике работы со студентами на кафедре информационных систем и технологий СГАУ к настоящему времени использованы уже две его редакции. В ходе этой апробации был замечен и устранен ряд недостатков как самих методических указаний, так и используемых в практике проектов. Однако, как в любой сложной открытой системе, отдельные недостатки все же не могут не присутствовать. Это следует учесть тем читателям,

кто решится использовать данный материал «в железе». Авторы и редакция не могут нести ответственность за возможные последствия в этих случаях. Тем не менее, все замечания и конструктивные предложения будут рассмотрены и учтены.

Информация для контактов – адрес электронной почты: oleg_zayakin@inbox.ru; адрес обычной почты: СФ ФИАН, ул. Ново-Садовая, 221, г. Самара, 443011, Россия; номер служебного телефона: +7 846 335 95 83; номер служебного факса: +7 846 335 56 00; служебная веб-страница (строго модерируется!): http://www.fian.smr.ru/personal_page.php?id=48&lang=rus .

Хочется выразить признательность доценту кафедры ИСТ А. С. Овсянникову за любезное предоставление проектов для использования их в MAX+PLUS II.

ВВЕДЕНИЕ

В классической фон-неймановской вычислительной машине (ВМ) арифметическая и логическая обработка данных возлагается на АЛУ, которое относят к классу операционных устройств (ОУ). В реальных вычислительных машинах АЛУ обычно реализуется не как единое устройство, а в виде комплекса операционных устройств (ОПУ), каждое из которых ориентировано на определенные операции и определенные типы данных. Отметим, что в современных ЭВМ этот комплекс реализован, как правило, на одном чипе. Следуя [1], в общем случае можно выделить ОПУ, предназначенные для выполнения следующих операций:

– арифметическая обработка чисел в форме с фиксированной запятой;
– арифметическая обработка чисел в формате с плавающей запятой;
– логическая обработка данных;
– десятичная арифметика.

Специализированные ОПУ для логической обработки данных и десятичной арифметики в современных ЭВМ встречаются достаточно редко, поскольку подобную обработку можно достаточно эффективно организовать на базе ОПУ для чисел с фиксированной запятой. Таким образом, будем считать, что АЛУ образуют два вида операционных устройств: для обработки чисел в форме с фиксированной запятой и обработки чисел с плавающей запятой. В свою очередь, эти ОПУ также могут представлять собой совокупность операционных устройств, специализированных под определенную арифметическую операцию, например, ОПУ умножения, ОПУ деления и т. п.

В минимальном варианте АЛУ должно содержать аппаратуру для реализации лишь основных логических операций, сдвигов, а также сложения и вычитания чисел в форме с фиксированной запятой. Исходя из этого набора, можно программным способом обеспечить выполнение остальных арифметических и логических операций как для чисел с фиксированной запятой, так и для других форм представления информации. Однако подобный вариант не позволяет добиться высокой скорости вычислений, поэтому по мере расширения технологических возможностей доля аппаратных средств в АЛУ постоянно возрастает.

В данном цикле лабораторных работ студенты проходят, фактически, лишь первоначальное знакомство с АЛУ. Поэтому для (относительной!) простоты изложения в него не включены вопросы программной и аппаратной реализации арифметических операций для чисел с плавающей запятой. Тем не менее, даже в таком «целочисленном» виде этот цикл дает достаточное представление о том, как считает числа ЭВМ, для чего, собственно, ее и изобрели люди. Все современные приложения ЭВМ и систем, такие как администрирование баз данных, работа в сетях, мультимедиа (звук и видео), игры, экспертные интеллектуальные системы и т. д., в конечном счете, сводятся к цифровой двоичной арифметике и булевой логике.

Ознакомление с принципами организации АЛУ составляет одну из важных целей представленных лабораторных работ. Изучение работы АЛУ, его функционирование во времени составляет другую цель работ. И, наконец, студенты в процессе этого обучения также знакомятся на практике с современными средствами

проектирования цифровых вычислительных систем, основу которого составляют ПЛИСы. Последние все более широко применяются в цифровых контроллерах, конкурируя с универсальными микроконтроллерами и микропроцессорами.

Изучение предмета в данном цикле лабораторных работ основано на булевой логике, теории автоматов и цифровой схемотехнике.

В подготовке лабораторных работ использовались – в качестве ядра автоматизированной системы – компьютеры, совместимые IBM PC. Использованы компьютеры на основе микропроцессоров типа Pentium.

Программное обеспечение (ПО) компьютеров, использованное в лабораторных работах, следующее. Это операционные системы (ОС) Windows XP, либо Windows Vista, либо Windows 7. Прикладное программное обеспечение включает в себя типичный набор прикладных программ для офиса, обычно устанавливаемых на компьютере вместе с операционной системой.

Для выполнения лабораторных работ потребуется также и специализированное приложение – система автоматизированного проектирования MAX+PLUS II («Altera», США) в версии не ниже десятой. В данном цикле работ в последнее время использовалась версия 10.2.

По описанию САПР MAX+PLUS II и работе с ней издано немало публикаций. Наше изложение этого материала основано на трудах украинских специалистов В. Стешенко [2] и В. Поречного [3].

В методических указаниях достаточно подробно излагаются теоретические вопросы.

Успешное изучение предмета в данном цикле лабораторных работ требует знания основ цифровой схемотехники.

Кроме этого, студентам также потребуются базовые физико-математические знания в объеме технического вуза.

Знания, полученные из данного практикума, будут полезными в курсах «Микропроцессорные средства систем автоматизации», «Операционные системы» (ОС), «Интерфейсы АСОИУ». Эти знания также являются основой для курсового и дипломного проектирования.

В данном цикле лабораторных работ использованы проекты цифровых устройств, разработанные на MAX+PLUS II студентами и преподавателями СГАУ. В последние годы они успешно использовались в процессе курсового и дипломного проектирования студентами СГАУ специальности «АСОИУ». Созданные программы тщательно протестированы. Все эти проекты можно найти в Учебно-методическом комплексе дисциплины (УМКД) по курсу «Аппаратные средства вычислительной техники» по специальности 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем», составленном в соответствии с учебным планом 090105.65-10-О-П для набора 2010 на очную форму обучения на факультете «Информатика». Данный УМКД относится ко второму поколению таких комплексов. Он составлен на кафедре ИСТ СГАУ.

Надеемся, что сведения, изложенные в данных методических указаниях, будут интересны и полезны студентам и специалистам соответствующих специальностей.

1 ЛАБОРАТОРНАЯ РАБОТА №1. ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКОЙ РАБОТЫ С СИСТЕМОЙ MAX+PLUS II

Цель лабораторной работы – приобретение навыков работы в среде MAX+PLUS II.

1.1 Теоретические и практические основы лабораторной работы

1.1.1 Характеристика пакета MAX+PLUS II

MAX+PLUS II является системой проектирования устройств на ПЛИС фирмы «Altera». Программное обеспечение системы MAX+PLUS II, представляющее собой единое целое (так называемая «интегрированная среда»), обеспечивает пользователю управление средой логического проектирования и помогает достичь максимальной эффективности и производительности. Все пакеты этой среды способны работать на платформе IBM PC.

Данный пакет свободно распространяется фирмой «Altera» на условиях стандартной лицензии.

Для нормальной инсталляции и работы САПР MAX+PLUS II необходим компьютер IBM PC или совместимый с ним с процессором не хуже Pentium, объемом ОЗУ не менее 16 Мбит и свободным местом на жестком диске от 150 до 400 Мбит в зависимости от конфигурации системы.

В качестве операционной системы можно использовать не старее, чем Windows NT. Хорошо подходит Windows XP.

Отметим, что пакет не русифицирован. Это создает некоторые трудности при работе. Так, для названия файлов проекта требуется использовать только латинские буквы. Лучше также использовать не русифицированные версии Windows, хотя последнее не так критично.

Также существует ограничение на длину полного имени файлов, с которыми работает данная САПР.

Во время инсталляции системы MAX+PLUS II создаются два каталога: \maxplus2 и \max2work. Каталог \maxplus2 содержит системное ПО и файлы данных. Каталог \max2work содержит файлы обучающей программы и примеры.

Отметим, что папки, нужные пользователю для работы в данной САПР, лучше создать еще до ее запуска, используя для этого средства ОС компьютера.

Пользователю рекомендуется создавать свои папки в каталоге \max2work. В ряде случаев, для того чтобы добиться нормальной работы от данной САПР, помогает размещение папок пользователя непосредственно в каталоге maxplus2.

Если вы инсталлируете данную САПР самостоятельно, следует иметь в виду, что по умолчанию она записывается на системный диск (диск C:\) компьюте-

ра. В этом случае особенности инсталляции могут вызвать проблемы в работе с САПР. Это, очевидно, происходит из-за антивирусных программ, имеющихся на компьютере, причем как отдельных, так и входящих в ОС. Проблемы можно обойти, если при инсталляции задать для установки файлов САПР имя диска данных пользователя (обычно это диски компьютера D:\, E:\ и т. д.).

Название системы MAX+PLUS II является аббревиатурой от Multiple Array MatriX Programmeble Logic User System (Пользовательская система программирования логики упорядоченных структур).

Начнем с важного базового понятия, связанного с системой MAX+PLUS II – понятия проекта.

Проект (Project) – совокупность файлов, представляющая собой описание цифрового устройства с помощью средств, предоставляемых системой MAX+PLUS II. Проект обычно имеет иерархическую структуру (исключение – проект из одного файла), на верхнем уровне иерархии стоит файл, имя которого определяет имя проекта, на нижнем уровне – примитивы (логические вентили, триггеры, ячейки памяти, входные и выходные выводы и т. д.).

Отметим для справки, что в системе MAX+PLUS II отдельный файл в данной иерархической структуре носит название «логический дизайн» (Design). На более низком уровне иерархии стоят поддизайны (Subdesign). Структура, включающая все дизайны, от верхнего уровня, на котором находится один дизайн (который имеет ее имя), до самого нижнего, называется проектом. Естественно, части проекта, удовлетворяющие этому определению, также могут рассматриваться и как отдельные проекты. Кстати, этим мы будем пользоваться в описанных здесь лабораторных работах.

Система MAX+PLUS II разработана фирмой «Altera» и обеспечивает многоплатформенную архитектурно-независимую среду создания дизайна, легко приспосабливаемую для конкретных требований пользователя. Система MAX+PLUS II имеет средства удобного ввода дизайна, быстрого прогона и непосредственного программирования устройств.

1.1.2 Работа в MAX+PLUS II

Прежде чем создавать свой собственный проект, необходимо организовать под него отдельный каталог на жестком диске, также необходимо учесть, что в процессе работы может потребоваться создание новых каталогов, поэтому рекомендуется заранее продумать, где они будут расположены, дабы не занимать пространство жесткого диска.

Запустив систему, можно увидеть окно, представленное на рис. 1, в верхней строке после слов MAX+PLUS II Manager можно увидеть название текущего проекта, представляющее собой название головного файла без расширения. При первом запуске это пустая строка.

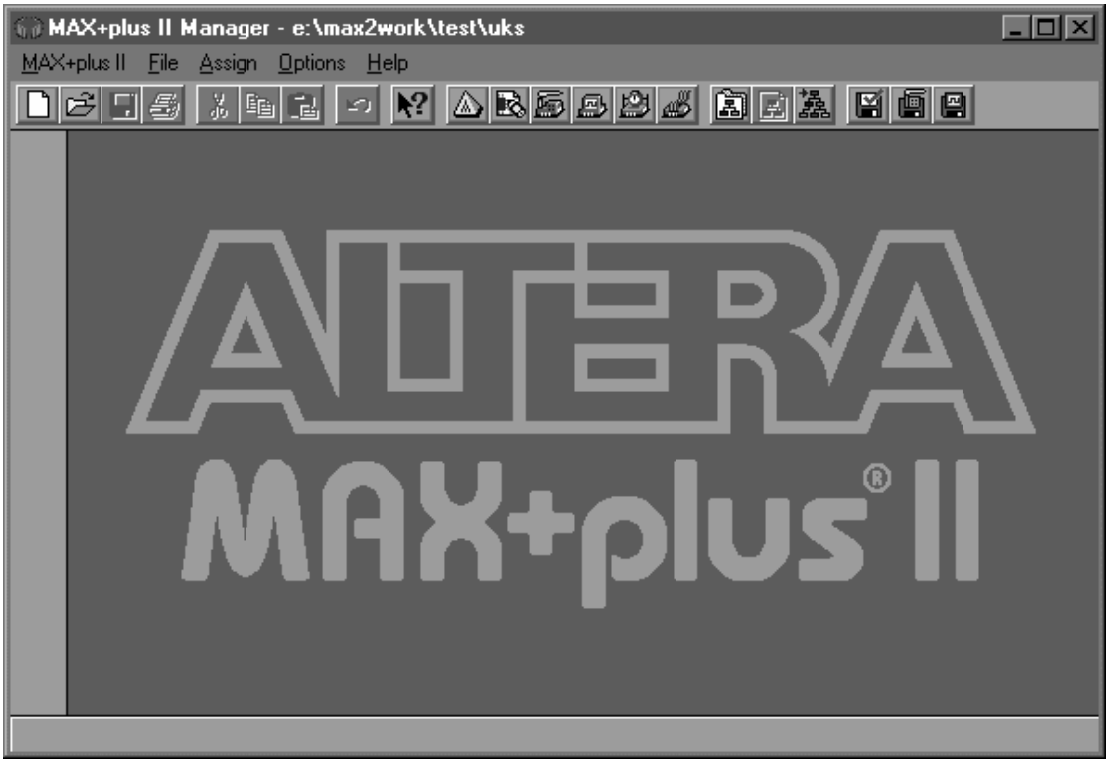


Рисунок 1 – Окно графического интерфейса MAX+PLUS II при запуске системы

Далее следует создать головной файл нового проекта, выбрав пункт NEW из меню FILE (см. рис. 2).

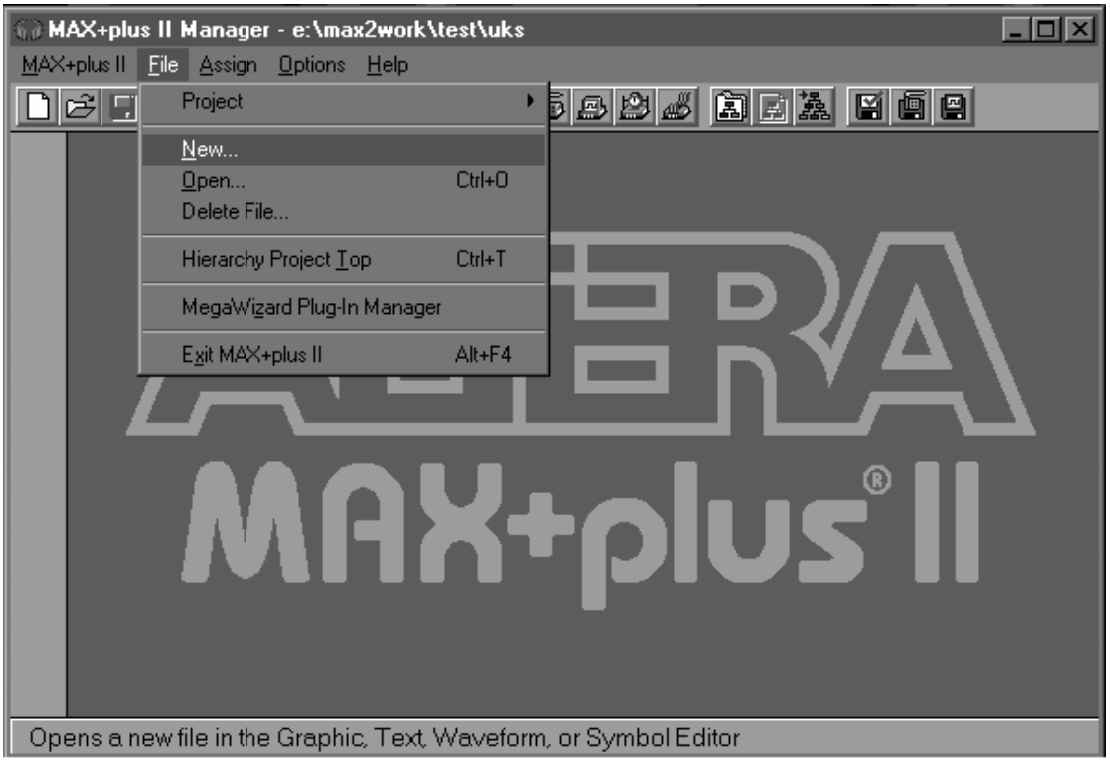


Рисунок 2 – Окно графического интерфейса MAX+PLUS II перед выбором пункта NEW

В появившемся окошке (рис. 3) следует выбрать тип файла, который будет являться главным файлом проекта. В нашем случае это файл графического редактора. Справа предлагается выбрать тип создаваемого файла, по умолчанию это

GDF-файл графического редактора MAX+PLUS II, но система может работать также с файлами пакета OrCAD типа SCH. Так как необходимости интеграции с OrCAD у нас нет, рекомендуется пользоваться «родным» форматом файлов.

После нажатия кнопки OK будет создан безымянный файл графического редактора (см. рис. 4), который нужно сохранить в выделенной под него директории под желаемым именем, выбрав команду SAVE из меню FILE. В нашем случае файл был сохранен под именем PRIMER.GDF в каталоге E:\max2work\primer. Затем следует сказать системе, что именно над этим файлом мы сейчас и работаем, так как, если взглянуть в поле названия проекта, можно увидеть, что текущим по-прежнему является предыдущий проект (при первом запуске – пустой проект).

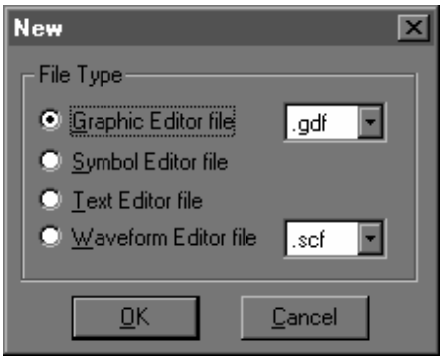


Рисунок 3 – Диалоговое окошко NEW

Это делается путем выбора из меню **File > Project** пункта **Set Project to Current File** (см. рис. 5). После выполнения указанной последовательности действий можно начинать проектирование, отладку, моделирование, временной анализ какой-либо цифровой схемы.

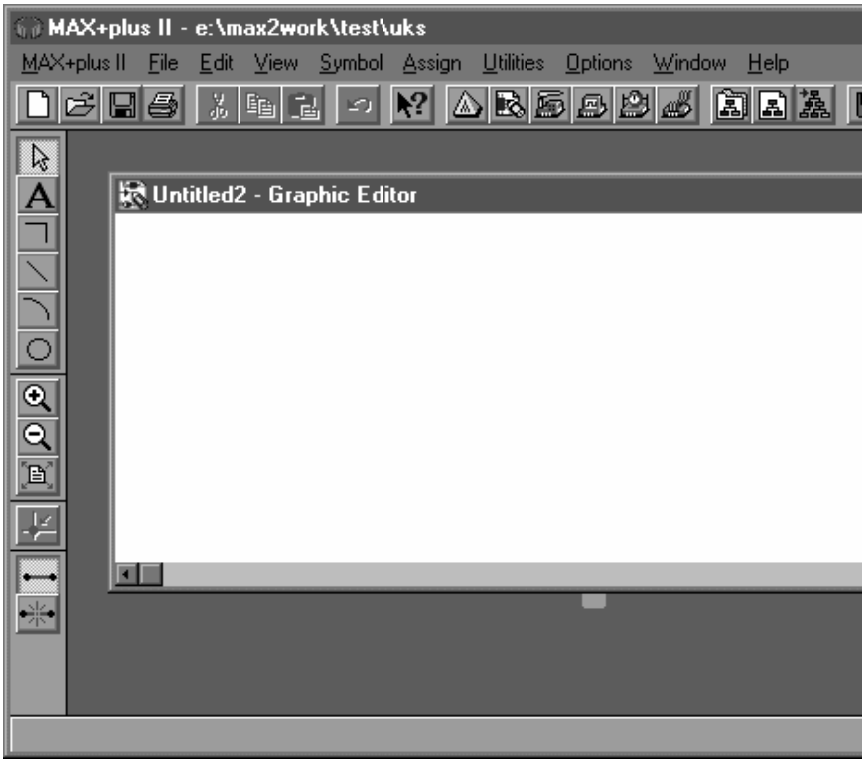


Рисунок 4 – Пока безымянный файл проекта в графическом редакторе

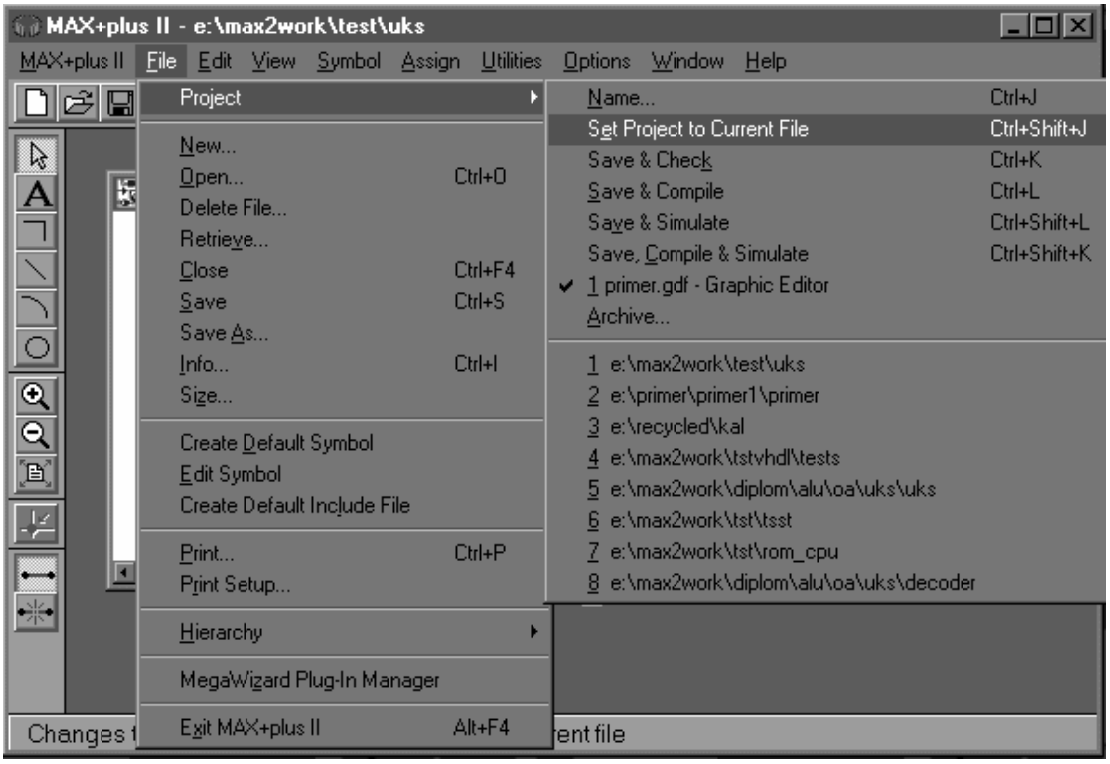


Рисунок 5 – Так системе указывается, что работа идет над новым проектом

На рис. 6 представлен внешний вид системы, готовой для ввода проекта PRIMER, с головным файлом PRIMER.GDF.

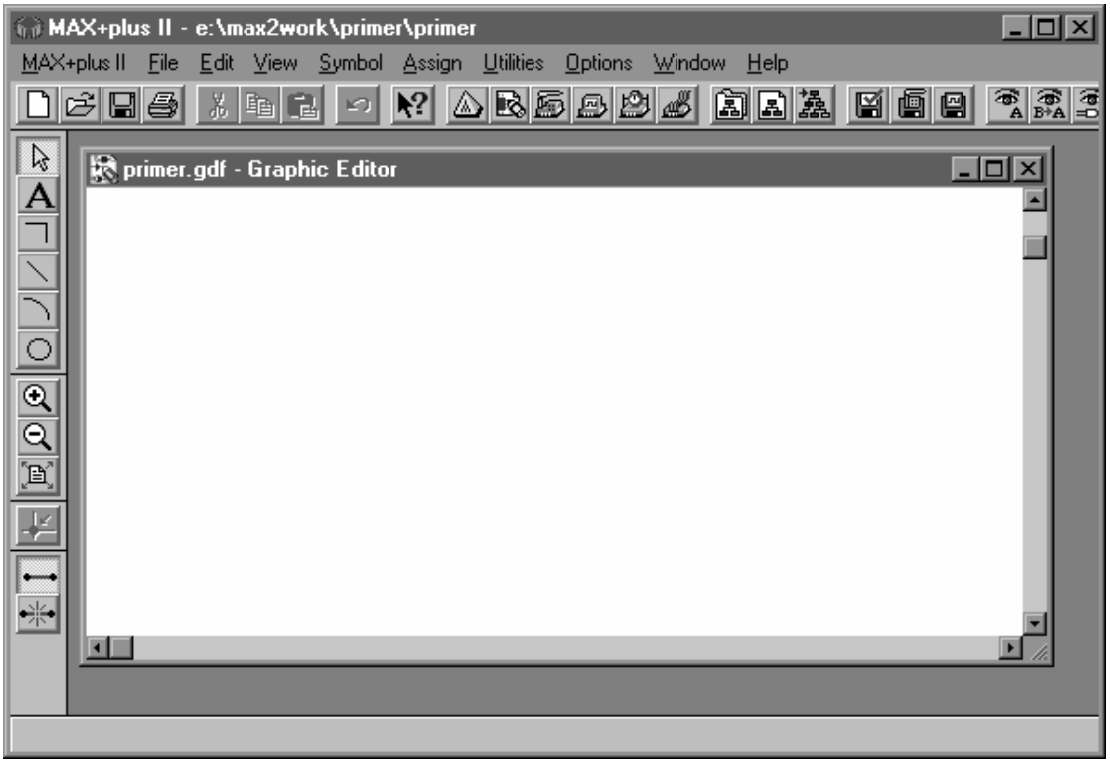


Рисунок 6 – Окно программы, готовой к работе с новым проектом

В качестве примера создадим простую схему (рис. 7), состоящую из сумматора и трех регистров, два из которых входные, а один выходной.

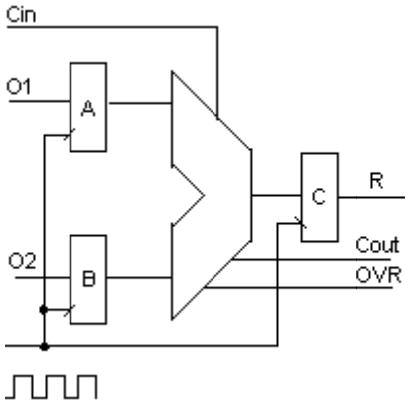


Рисунок 7 – Блок-схема сумматора

Функция сумматора заключается в следующем. По фронту синхросигнала операнды O1 и O2 защелкиваются в регистры A и B, а по его срезу результат защелкивается в регистр C. Сумматор также принимает сигнал переноса Cin и выдает сигналы переноса и переполнения Cout и OVR соответственно.

Для того чтобы вставить символ в поле графического редактора, нужно щелкнуть правой кнопкой мыши на свободном месте и в появившемся меню выбрать **Enter Symbol** (рис. 8). После этого появится окошко, изображенное на рис. 9, в котором представлен список доступных библиотечных директорий, среди которых можно заметить и текущую директорию проекта.

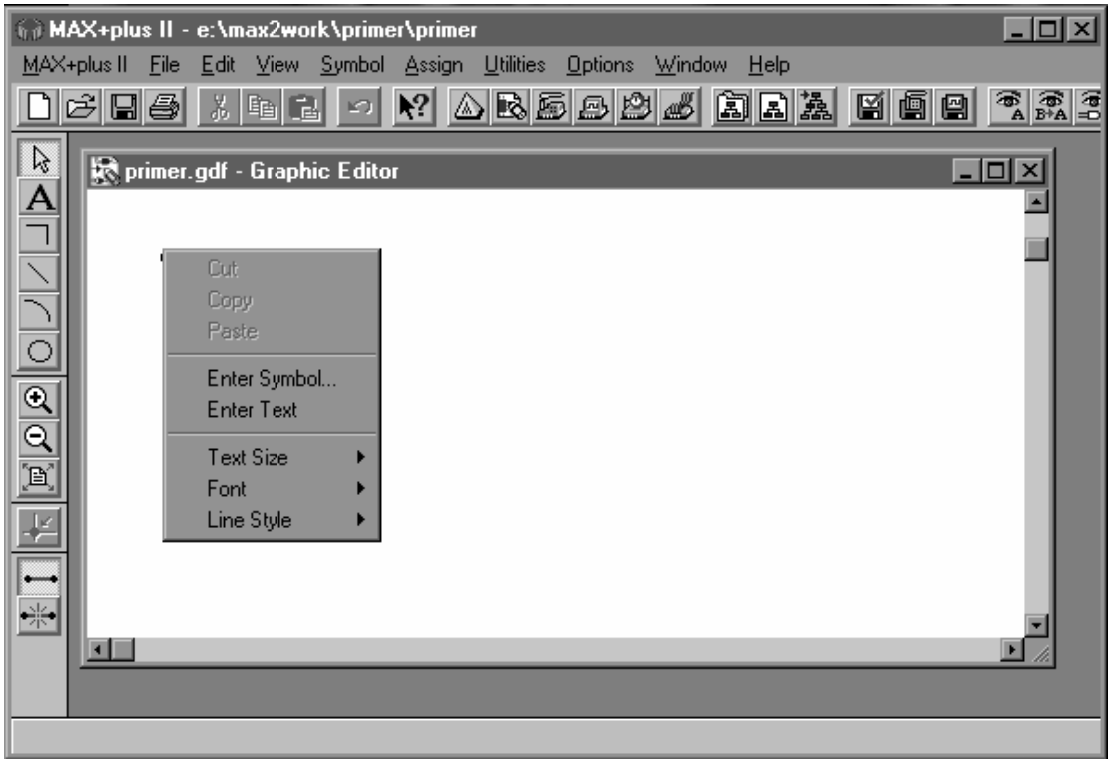


Рисунок 8 – Окно программы перед вставкой нового символа в поле графического редактора

Далее надо либо выбрать необходимый компонент из имеющихся в наличии, либо создать новый при помощи утилиты MegaWizard Plug-In Manager (для ее освоения достаточно несколько раз в разных комбинациях последовать ее эк-

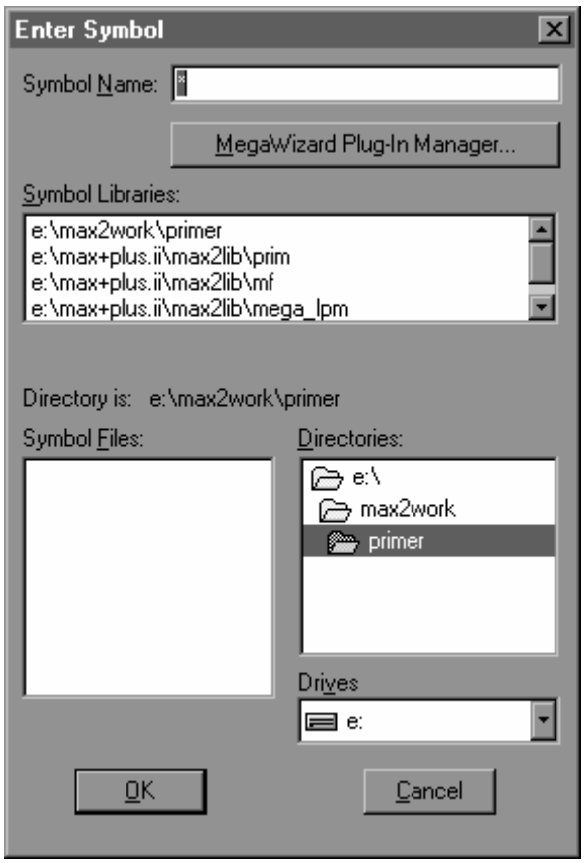


Рисунок 9 – Диалоговое окно программы для вставки нового символа в поле графического редактора

ранным инструкциям). Если компонент создается при помощи MegaWizard Plug-In Manager'а, то следует сохранить его в отдельной директории.

На рис. 10 представлен внешний вид уже введенного проекта, реализующего схему, представленную на рис. 7. Следует отметить, что входные и выходные ножки проектируемого устройства вводить обязательно, они лежат в директории **\prim** и называются **input** и **output** соответственно.

Компоненты проекта соединяются соединительными линиями, толщина их может быть изменена правым щелчком мыши по любой точке соответствующей линии (пункт всплывающего меню **Line Style**).

Следует отметить, что **толстая линия не означает шины**. Для того чтобы сделать линию шиной, необходимо ее поименовать, щелкнув правой кнопкой по любой ее точке выбрав **Enter Node/Bus name** и введя имя вида **<имя>[m..n].** Аналогично задав в качестве имени входа или выхода конструкцию вида **<имя>[m..n]**, получим не вывод, а сразу группу выводов с именами **<имя>m, <имя>m+1, <имя>m+2, ..., <имя>n.** Отделение от шины какой-либо линии производится путем явного указания имени этой линии.

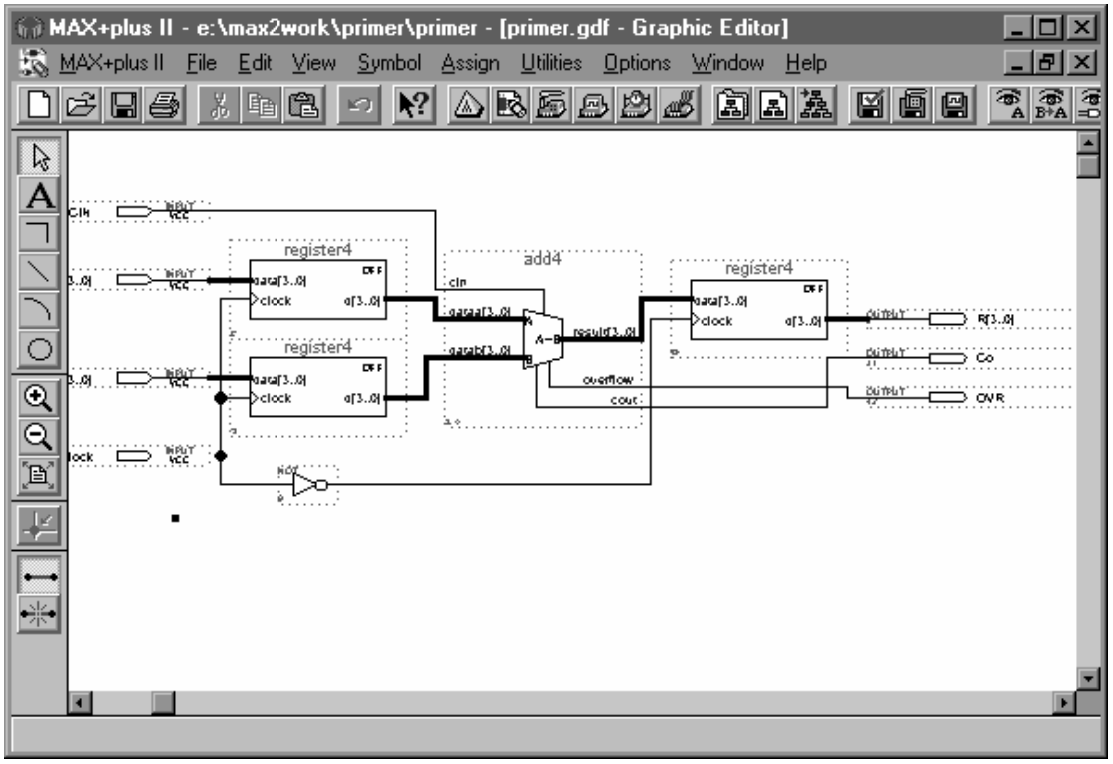


Рисунок 10 – Внешний вид готового проекта, реализующего сумматор, показанный на рис. 7

После успешного введения проекта следует его откомпилировать, выполнив команду **Compiler** из меню **Max+Plus II** (рис. 11). При этом появится окно компилятора с единственной активной кнопкой **Start**.

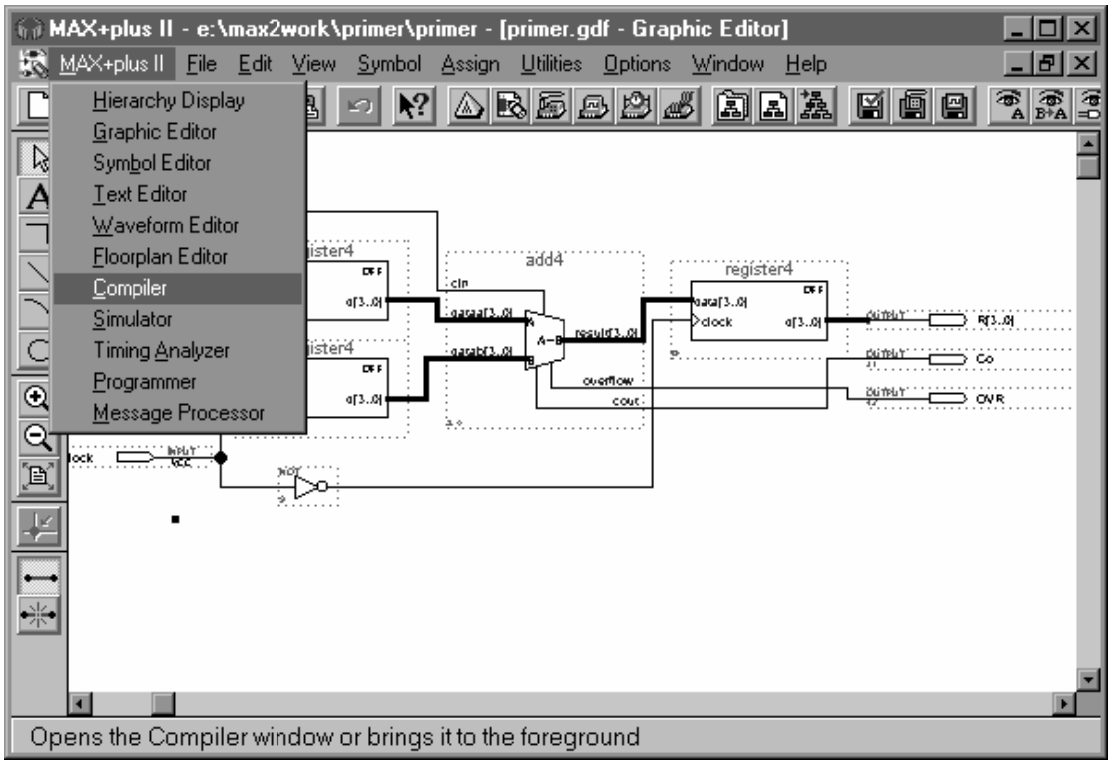


Рисунок 11 – Окно программы перед выбором команды **Compiler**

В процессе компиляции неизбежны сообщения об ошибках. Следует не полениться и тщательно перевести эти сообщения, в случае необходимости можно

воспользоваться справочной системой, наведя курсор на соответствующее сообщение и нажав на кнопку **Help on Message**.

После того как все ошибки будут устранены, можно приступать к отладке и моделированию введенной схемы. Для этого сначала нужно создать тестовый вектор при помощи утилиты Waveform Editor (рис. 12).

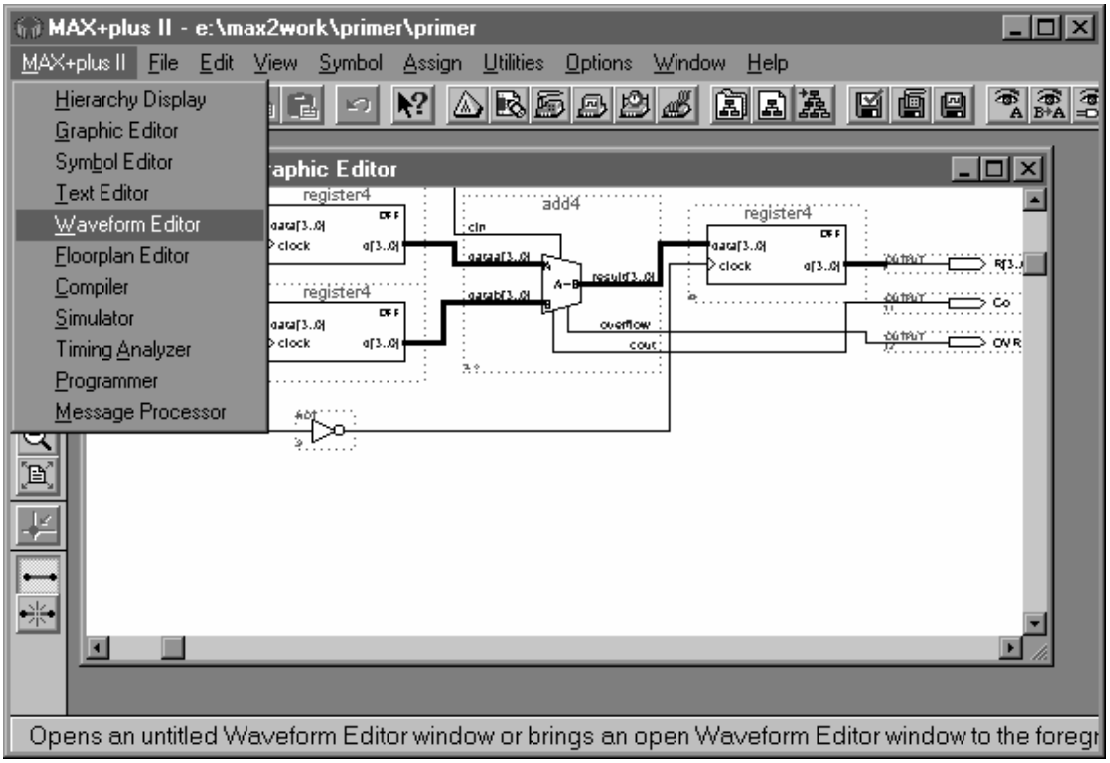


Рисунок 12 – Окно программы перед созданием вектора временных диаграмм

Waveform Editor позволяет вводить и выводить тестовые последовательности для любого узла (Node) проекта. Узлами в первую очередь считаются входные и выходные ножки схемы (они обязательно имеют имена, введенные пользователем), а также соединительные линии и шины, имена которых могут быть даны системой автоматически.

Входные линии имеют тип IN, выходные – OUT а внутренние – BURIED.

Для того чтобы добавить узел в список отображаемых узлов, надо щелкнуть правой кнопкой мыши по пустой строке и выбрать Insert Node (рис. 13).

Значения в соответствующих строках рисуются с помощью инструментов в левой части окна системы. Не следует забывать, что с помощью мышки можно нарисовать входные значения лишь с дискретностью решетки, которая по умолчанию равна 100 нс, чтобы сменить этот шаг надо воспользоваться меню **Options>GridSize**. Задавать какие-либо значения узлам типов OUT и BURIED нецелесообразно, так как эти значения будут затерты при моделировании.

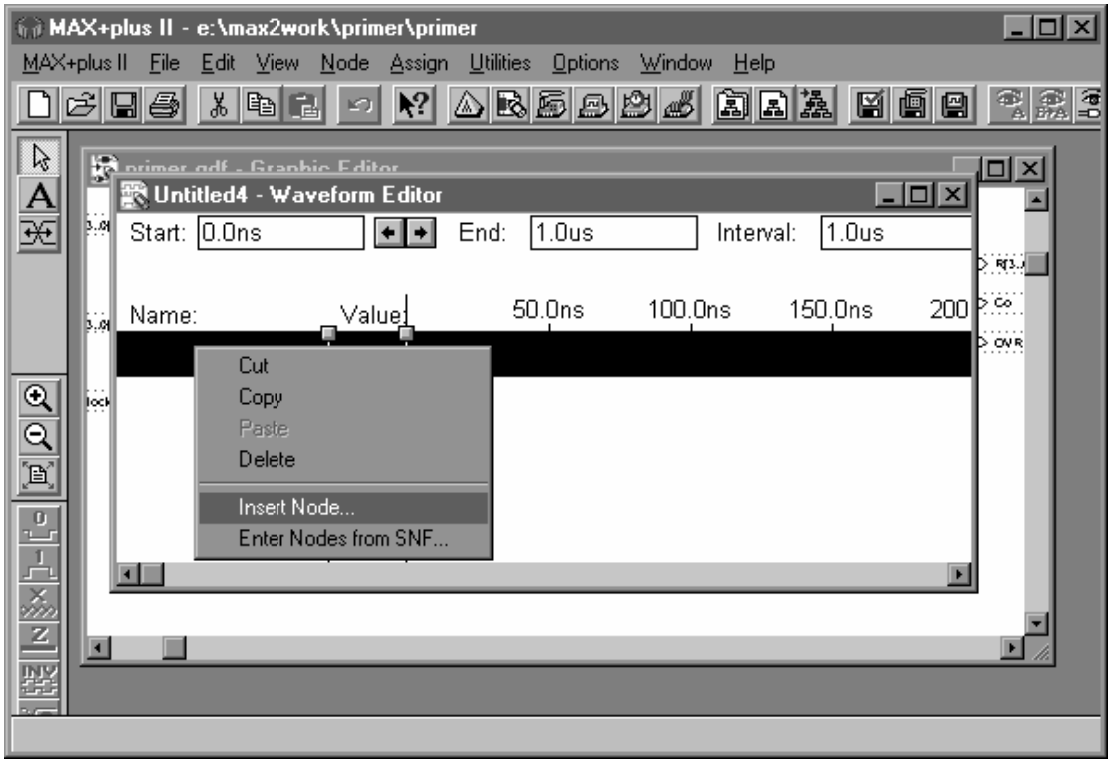


Рисунок 13 – Окно программы перед добавлением узла из списка всех узлов, имеющихся в проекте

После создания тестового файла его надо сохранить в директории проекта с именем, совпадающим с именем проекта.

Далее можно запускать Simulator (см. рис. 14). По завершении его работы в окошке нажать кнопку Open SNF и наблюдать результаты моделирования.

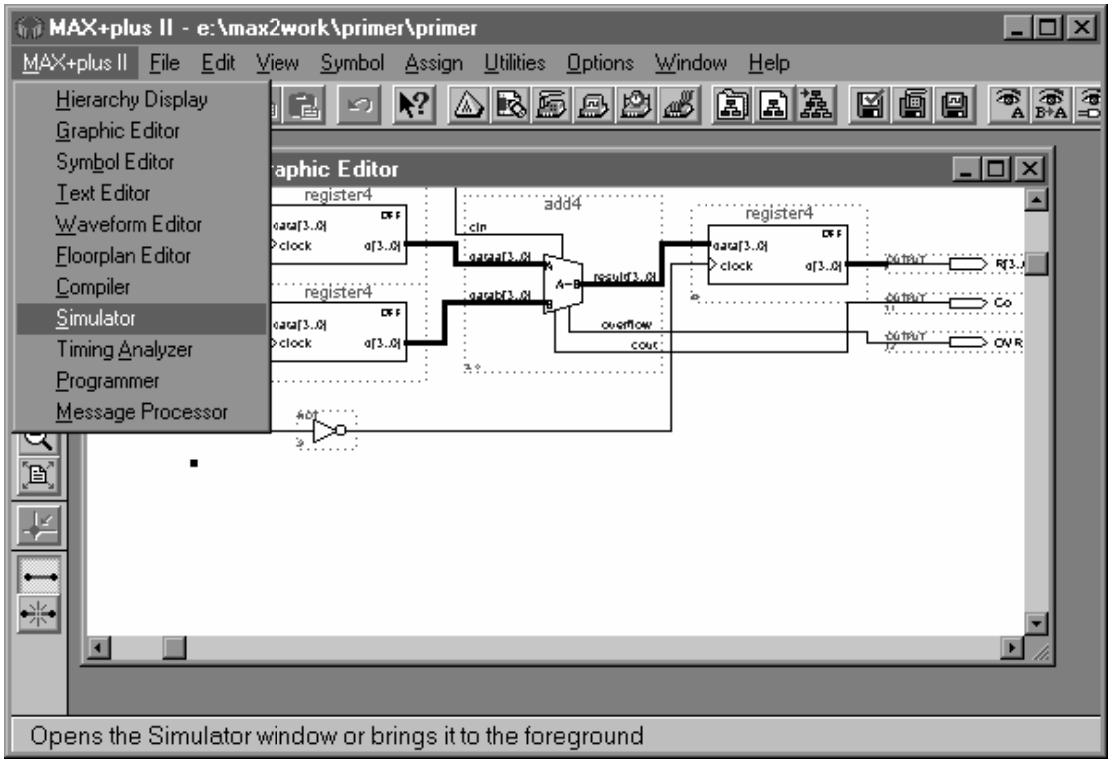


Рисунок 14 – Окно программы перед запуском симулятора

Последний штрих создания проекта File > Create Default Symbol, при этом создается символьный файл проекта (квадратик с ножками), который может быть использован в качестве строительного материала при создании других проектов. Еще одна полезная команда File>Project>Archieve, которая позволяет сгруппировать в одной директории все файлы (библиотечные тоже), использованные в проекте для безболезненного переноса проекта с одного рабочего места на другое.

1.2 Задание на самостоятельную работу

Создать проект 4-х разрядного вычитателя. Старший четвертый разряд – знаковый. Для этого в **Wizard** (Мастер) выбрать окно **Arithmetic**. Затем, следуя подсказкам Мастера, выбрать опцию **Subtruction Only**. Ограничиться только неотрицательными числами.

При создании вычитателя с помощью Мастера следует выбрать необходимые выводы для осведомительных сигналов – признаков нулевого результата и переполнения.

Входные и выходные регистры следует создавать также с помощью Мастера, выбрав при этом категорию **Storage**.

Входы, выходы и инвертор можно выбрать и без Мастера, используя папку **Primitives**. Соответственно, следует выбрать категории **Input**, **Output** и **Not**.

Следует тщательно, с перекрытием, соединять линиями выбранные и размещенные на экране значки элементов проекта. Для обозначения шины используются квадратные скобки, например: A[3 .. 0] – для четырехразрядной шины.

Для демонстрации работы в симуляторе из вертикального меню слева надо нажать на третью кнопку по порядку снизу (то есть навести курсор мыши на экране дисплея на изображение этой кнопки и нажать на левую клавишу мыши). Рекомендуется в качестве начальных значений для уменьшаемого выбрать четыре или пять, инкремент выбрать равным единице, а начальное значение для вычитаемого выбрать равным нулю, инкремент выбрать равный двум.

Период смены данных на входах следует выбрать вдвое большим, чем период тактовых импульсов. Для этого в том же диалоговом окне, которое появляется при нажатии на третью снизу кнопку, задать опцию **Multiple,** равную двум. При этом начальную фазу следует выбрать так, чтобы передний фронт тактовых импульсов находился посередине между фронтами импульсов на входах вычитателя. Это можно сделать, задав начальное значение тактовым импульсам (0 или 1).

Запустить **Simulator** при максимальной частоте, которую позволяет САПР. Затем повторить запуск при меньших частотах. Определить задержку времени при срабатывании вычитателя. Определить максимальную частоту, на которой проектируемое устройство работоспособно. Сравнить с тактовой частотой современных компьютеров и микроконтроллеров.

Проекты для выполнения лаб. работ с программой Max+PLUS II находятся в папке «Лаб. работа вводная по Max+plus II - Пример выполнения программы для лаб. работы».

Готовый проект, относящийся к лабораторной работе №1, находится в папке \firstz.dir. Однако рекомендуется сделать его вновь самостоятельно – он довольно простой. Перед компиляцией рекомендуется выбрать опцию Assign | Device | MAX7000.

В конце работы сохраните проект также в виде элемента. Попробуйте включить его в блок-схему нового проекта в качестве этого элемента.

1.3 Содержание отчета

Отчет должен содержать цель работы, постановку задачи, блок-схему проектируемой схемы, блок-схему проекта, созданного в ходе выполнения лабораторной работы, а также временные диаграммы с заданными параметрами сигналов. Следует записать длительность периода тактовых импульсов. Также отчет должен содержать значение максимальной рабочей частоты проекта.

Следует сделать выводы по результатам работы.

1.4 Контрольные вопросы

1. Расскажите об этапах проектирования и тестирования проекта в среде MAX+PLUS II.
2. Для чего нужна опция в MAX+PLUS II «Set Project to Current File»? Можно ли без нее обойтись в лабораторной работе?
3. Назовите основные достоинства и недостатки MAX+PLUS II.
4. Для чего нужна опция в MAX+PLUS II «File>Project>Archieve»? Можно ли без нее обойтись в лабораторной работе?
5. Объясните, для чего использован инвертор в блок-схеме вычитателя.
6. Объясните, для чего нужны входные и выходные регистры, если комбинационные схемы элемента add4 и так содержат триггерные ячейки.
7. Поясните логику осведомительных сигналов в проекте вычитателя.
8. Как связаны между собой разрядность и быстродействие вычитателя?

2. ЛАБОРАТОРНАЯ РАБОТА №2. ПРИНЦИПЫ ПОСТРОЕНИЯ АРИФМЕТИКО-ЛОГИЧЕСКОГО УСТРОЙСТВА

Цель лабораторной работы – изучение принципов построения арифметико-логических устройств (АЛУ), исследование АЛУ различных архитектур, получение навыков работы с ними.

2.1 Теоретические основы лабораторной работы

Назначением АЛУ является реализация арифметических, логических и других операций из списка операций $F=(f_1,\ldots,f_G)$ над словами информации, называемыми операндами. АЛУ является одним из основных устройств, входящих в состав центрального процессора ЭВМ. В состав современных процессоров обычно включаются АЛУ двух типов – целочисленные (с фиксированной запятой) и с плавающей запятой. Функционирование АЛУ осуществляется под управлением центрального устройства управления (ЦУУ) процессора. Для управления работой АЛУ обычно применяется один из двух вариантов: централизованное или децентрализованное (автономное) управление.

Централизованное управление используется в компьютерах класса RISC. При централизованном управлении работой АЛУ (как исполнительным устройством) управляет ЦУУ процессора путём выработки управляющих сигналов, обеспечивающих выборку операндов, выполнение операции, сохранение результата.

Децентрализованное управление используется в компьютерах класса CISC. При децентрализованном управлении ЦУУ осуществляет выборку операндов из памяти в регистры АЛУ. Выполнением операции управляет не ЦУУ, а встроенное в АЛУ автономное устройство управления, которое, приняв от ЦУУ код «g» операции $f_g \in F$, формирует сигналы, под управлением которых в АЛУ выполняется операция f_g. Запись результата операции f_g в память осуществляется опять под управлением сигналов ЦУУ процессора.

Известно, что АЛУ с автономным устройством управления, ЦУУ процессора, контроллеры различных периферийных устройств относятся к классу операционных устройств (ОУ) [1]. Операционным называется устройство, назначением которого является выполнение операций из заданного списка операций $F=(f_1,\ldots,f_G)$ над словами-операндами из списка $D=(d_1,\ldots,d_H)$ с целью получения слов-результатов $R=(r_1,\ldots,r_Q)$, причём в каждый момент времени в ОУ может выполняться одна операция $R=f_g(D^*)$ (рис. 15*а*).

Организация ОУ базируется на принципе микропрограммного управления [1], в соответствии с которым любая операция $f_g \in F$ рассматривается как сложное действие и разделяется (путём декомпозиции) на совокупность элементарных действий, называемых микрооперациями (МО). Совокупность элементарных действий задаётся списком МО: $СМО=(МО_1, \ldots, МО_M)$. Порядок выполнения МО (из списка СМО) задаётся алгоритмом A_g операции $f_g \in F$. Элементарность микрооперации $МО_m \in СМО$ означает, что для её выполнения достаточно операнды подать на входы специальной (уникальной) комбинационной схемы

(КС) и, после завершения переходных процессов в ней (после паузы), сохранить (принять) результат с выхода $КС_m$. Пример: двоичный комбинационный сумматор обеспечивает выполнение МО сложения кодов (операндов A и B), подаваемых на входы A, B сумматора: $C_{out}.C := A+B+C_{in}$, где C – сумма, C_{in} – значение на входе переноса в младший разряд сумматора, C_{out} – значение на выходе переноса из старшего разряда сумматора. Для сохранения (приема) ре-

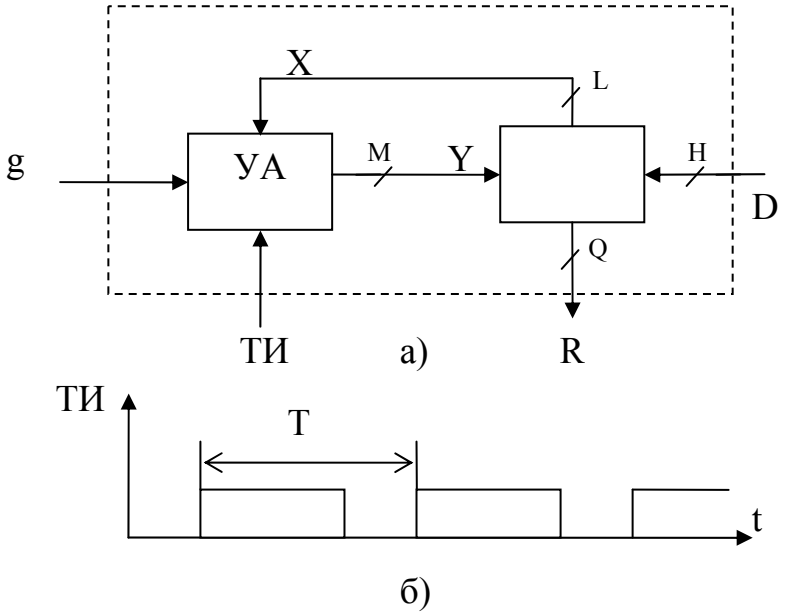


Рисунок 15 – Операционное устройство АЛУ:
а) – функциональная схема; б) – временная диаграмма работы

зультатов микроопераций с выходов комбинационных схем обычно используются регистры.

Порядок выполнения МО в процессе выполнения операции $f_g \in F$ задаётся алгоритмом A_g и зависит от значений логических условий (ЛУ). Логическое условие – булева переменная, которая принимает значение истина или ложь в зависимости от значений операндов и результатов МО, формируемых в процессе выполнения алгоритма A_g. Пример ЛУ: переменная Z (Zero) принимает значение 1 (истина), если результат МО равен 0, и значение 0 (ложь) – в противном случае. Логические условия используются в процессе выполнения алгоритма A_g в качестве условий (обычно) альтернативных переходов.

Алгоритм A_g, представленный в терминах МО и ЛУ, называется микропрограммой $МП_g$. Микропрограмма $МП_g$ определяет (задаёт) порядок выполнений микроопераций и проверки логических условий во времени. Совокупность микропрограмм $МП_1,\ldots,МП_G$ определяет (задаёт) функцию ОУ. Так, функцию АЛУ задают МП всех операций из списка операций $F_{АЛУ}$, в который включаются обычно арифметические операции сложения, вычитания, умножения, деления, логические операции и другие.

Итак, для выполнения микроопераций $МО_1,\ldots,МО_M$ используются соответствующие комбинационные схемы $КС_1,\ldots,КС_M$. Для сохранения результатов МО используются регистры. Для управления подачей операндов на входы КС (из раз-

личных регистров) используются специальные КС – мультиплексоры. Для управления записью результатов МО (с выходов различных КС) в регистры – демультиплексоры.

Часть ОУ, содержащая перечисленные выше схемы, образует исполнительную часть ОУ, которую принято называть операционным автоматом (ОА) устройства, в частности ОА АЛУ (рис. 15*а*). *Назначение ОА* – хранение слов информации, выполнение микроопераций из списка СМО=($МО_1$, …$МО_M$) под воздействием управляющих сигналов Y=(y_1, …, y_M) из УА, формирование значений логических условий (осведомительных сигналов) X=(x_1, …, x_L) для УА.

Выполнение операции $f_g \in F$, задаваемой кодом операции «g» на управляющем входе ОУ, осуществляется в последовательности, заданной алгоритмом A_g, представленным в форме $МП_g$ (другими словами, в ОУ по коду g запускается процесс выполнения микропрограммы $МП_g$). Управление процессом выполнения $МП_g$ осуществляется частью ОУ, которая относится к классу конечных автоматов. Ее принято называть управляющим автоматом (УА) АЛУ (рис. 15*а*). *Назначение УА* – управление работой ОА. Управление осуществляется путём выработки управляющих сигналов $y_m \in Y$ в той последовательности, которая задаётся микропрограммой $МП_g$ операции $f_g \in F$ и значениями осведомительных сигналов X=(x_1, …, x_L) из ОА. УА является управляющей (активной) частью ОУ.

Самый простой способ организации работы ОУ во времени – *синхронный*, при котором ОУ работает тактами. Такт – это фиксированный отрезок времени Т определённой длительности, задаваемый обычно как интервал времени между двумя соседними фронтами тактовых импульсов ТИ, вырабатываемых генератором тактовых импульсов с периодом Т (с частотой F=1/T) (рис. 15*б*). Этот отрезок времени (такт Т) отводится на выполнение (одной или нескольких) МО в ОУ и состоит из следующей последовательности этапов:

1) этап выработки управляющих сигналов $y_m \in Y$ в УА,
2) этап выборки операндов и выполнения $МО_m$ в $КС_m$ в ОА,
3) этап формирования ЛУ $x_L \in X$ по результатам $МО_m$,
4) этап занесения результатов $МО_m$ в регистры.

Далее эта последовательность этапов периодически повторяется с периодом Т до завершения процесса выполнения операции $f_g \in F$, т.е. до конца $МП_g$.

Такая организация работы ОУ называется синхронной. Положительный фронт тактового импульса ТИ (рис. 15*б*) используется обычно как начало этапа 2, а отрицательный фронт (срез) ТИ – как начало этапа 4. То есть по фронту запускается процесс выбора и подачи операндов на входы КС ОА, выполнения МО в КС и формирования значений ЛУ по результатам МО. А по срезу – процесс сохранения результатов МО и ЛУ в регистрах ОА с последующей выработкой управляющих сигналов в УА.

Продолжительность такта Т при синхронной организации работы ОУ определяется суммой продолжительностей этапов 1, 2, 3, 4 – $\tau_1 + \tau_2 + \tau_3 + \tau_4$, причём для наихудшего случая (максимальных по длительности МО и ЛУ):

$$\mathbf{T = \tau_{УА} + max(\tau_1, \ldots, \tau_m) + max(\tau_{X1}, \ldots, \tau_{XL}) + \tau_{сохр}}. \qquad (1)$$

Здесь: $\tau_1=\tau_{УА}$=const – время формирования управляющих сигналов $\{y_m\}$ в УА обычно постоянно, не зависит от того, какие управляющие сигналы $y_m \in Y$ вырабатываются в данном такте; $\tau_4=\tau_{сохр}$=const – время занесения результатов в регистры ОА. Время выполнения $МО_1$, …$МО_M$ (по сигналам y_1, …, y_M), а также время формирования ЛУ x_1, …, x_L в общем случае разное (поскольку задержка сигналов в соответствующих КС разная), поэтому продолжительность этапов τ_2, τ_3 определяется как максимум от всех возможных значений.

Основные *технические характеристики* АЛУ – быстродействие и затраты оборудования [4]. Быстродействие определяется количеством операций $f_g \in F$, выполняемых АЛУ в единицу времени: $V_g=1/t_g$, и зависит от времени выполнения операции $t_g=n_g \cdot T$, где n_g – количество тактов на выполнение операции $f_g \in F$. Продолжительность такта T=const, количество тактов n_g – величина переменная, зависит от сложности микропрограммы $МП_g$ операции f_g и для сложных операций (умножение, деление) значений операндов.

Затраты оборудования определяются сложностью схем ОА и УА, растут с увеличением количества G и сложности операций в списке операций F.

Описание схемы лабораторного макета АЛУ и порядок выполнения работы приведены ниже.

2.2 Задание на самостоятельную работу

Ознакомьтесь с тем, как выглядит проект АЛУ.

Проект по лабораторной работе №2 находится здесь: «Лаб. работы `Организация АЛУ` - Примеры выполнения программ для лаб. работы\2011\alu_full0.dir» (рекомендуется использовать этот проект). В нем для компиляции рекомендуется использовать файл ..\alu_full.gdf. Микропрограмму следует сначала записать в память. Для этого перед компиляцией нужно выбрать опцию Assign | Device | FLEX10K. Далее, после компиляции, рекомендуется выбрать файл ..\\alu_full.scf. После выбора опции «Simulator», но до нажатия его клавиши «Start», следует инициализировать память путем выбора опции «Initialise | Initialise memory». Рекомендуется выбрать опцию «Memory name | ua2:|pzu:37:lpm_rom:LPM_ROM_component|altrom:srom|content», и в открывшемся окне выбрать «Import file | alu_full(6311).mif». Для правильной работы микропрограммы (состоящей в этой работе из двух микрокоманд) этой микропрограмме должны предшествовать несколько других микрокоманд. Они записаны в загруженном файле alu_full(6311).mif.

Установите соответствие имен блоков АЛУ в этом проекте и имен этих блоков в данной методичке. По результатам работы составьте таблицу.

По тестовому примеру из лабораторной работы №5 проверьте правильность работы проекта. Для этого воспользуйтесь симулятором в MAX+PLUS II.

Определите максимальное быстродействие спроектированного АЛУ. Для этого запустите указанный симулятор в данном проекте несколько раз при разных длительностях периода ГТИ. Сравните результаты с тестовым примером из лабораторной работы №5. По результатам определите максимальную рабочую частоту ГТИ для данного спроектированного устройства. Сделайте выводы.

2.3 Содержание отчета

Отчет должен содержать цель работы, постановку задачи, блок-схему проектируемой схемы, блок-схему проекта, созданного в ходе выполнения лабораторной работы, а также временные диаграммы с заданными параметрами сигналов. Следует записать длительность периода тактовых импульсов. Также отчет должен содержать значение максимальной рабочей частоты проекта.

Следует сделать выводы по результатам работы.

2.4 Контрольные вопросы

1. Из каких структурных элементов состоит изучаемое АЛУ?
2. В чем состоит принцип микропрограммного управления, который впервые предложил Морис Уилкс?
3. Какие сигналы определяют данные на входе D ОА АЛУ?
4. Какие сигналы определяют данные на выходах АЛУ?
5. Для чего предназначены входы START, КОП, RES схемы АЛУ?
6. Каково назначение схем MUX1, MUX2, MUX3?

3 ЛАБОРАТОРНАЯ РАБОТА №3. ОРГАНИЗАЦИЯ ОПЕРАЦИОННОГО АВТОМАТА АРИФМЕТИКО-ЛОГИЧЕСКОГО УСТРОЙСТВА

Цель лабораторной работы – изучение принципов построения операционных автоматов арифметико-логических устройств (ОА АЛУ).

3.1 Теоретические основы лабораторной работы

3.1.1 Принципы построения операционного автомата арифметико-логического устройства

При построении ОА АЛУ можно использовать различные способы организации, отличающиеся по техническим характеристикам – производительности, быстродействию и затратам оборудования [4].

Под *производительностью* ОА понимается количество МО, выполняемых за один такт. В одном такте (одновременно) могут выполняться только совместимые МО, которые указываются в одном операторе микропрограммы МПg. Их число может меняться от такта к такту (в зависимости от микропрограммы МПg и значений операндов). Следовательно, производительность – это случайная величина, которую можно оценивать либо максимальным значением (равным максимальному числу МО за один такт), либо средним значением. Средняя производительность зависит от числа совместимых МО в операторах МПg, частоты использования операторов и структуры ОА, которая может накладывать ограничения на функциональную совместимость МО.

Быстродействие ОА характеризуется длительностью такта ОА – $T_{ОА}$ (чем меньше длительность такта, тем выше быстродействие). Такт операционного автомата – это отрезок времени, необходимый для выполнения микроопераций и формирования значений логических условий, то есть промежуток времени от момента поступления управляющих сигналов на вход ОА до момента выработки значений осведомительных сигналов, соответствующих состоянию ОА. Такт ОА зависит от внутренней структуры комбинационных схем и характеристик логических и запоминающих элементов, используемых в комбинационных схемах и памяти (регистрах) ОА.

Затраты оборудования определяются суммарным числом элементов в памяти (регистрах) ОА, комбинационных схемах, реализующих МО и формирующих значения ЛУ, и зависят от сложности и количества КС и регистров.

Важными характеристиками структур являются *регулярность и универсальность*. Регулярной называется структура, состоящая из однотипных частей, одинаковым образом связанных между собой. *Регулярность* структуры может проявляться в использовании одинаковых элементов или в использовании одинаковых схем для обработки значений в каждом разряде слова. Структура ОА является максимально регулярной, если все слова обрабатываются одинаковым образом (одним набором МО) и одновременно с этим все разряды слова обрабатываются одинаково. Чем более регулярна структура, тем проще процесс её производства, что в итоге приводит к уменьшению стоимости изделия.

Универсальность (многофункциональность) структуры проявляется в возможности выполнения этой структурой широкого класса функций. Если структура ОА универсальна, то реализация любого алгоритма сводится к перестройке структуры внешними средствами путём микропрограммирования. Определение степени универсальности вызывает серьёзные трудности, однако сравнительную оценку универсальности двух структур можно выполнить достаточно легко, то есть всегда можно определить, какая структура из двух является более универсальной. Чем более универсальна структура, чем шире область её применения. Увеличение степени универсальности структур позволяет сохранить номенклатуру выпускаемых изделий и увеличить объём выпуска каждого изделия, что приводит к снижению стоимости производства. Универсальность и регулярность структур особенно важна для БИС, поскольку их проектирование требует значительных затрат, которые окупаются при большом объёме производства. Следовательно, увеличение универсальности структуры приводит к снижению стоимости даже в тех случаях, когда универсальность достигается за счёт введения в структуру дополнительного оборудования.

Практика (опыт проектирования) показывает, что регулярность и универсальность – взаимосвязанные свойства. Регулярные структуры обычно более универсальны, чем нерегулярные, и увеличение степени универсальности можно достичь за счёт увеличения степени регулярности структуры.

В зависимости от производительности и затрат оборудования различают ОА с максимальными характеристиками (с максимальной производительностью и максимальными затратами оборудования) – *I-автоматы*, ОА с минимальной производительностью (одна МО за один такт) – *M-автоматы*, ОА с промежуточными (между min и max) значениями характеристик – *IM-автоматы*.

Структура ОА предопределяется его функцией. *Функция* ОА задаётся в виде трёх множеств:

- множество (список) СМО=($МО_1$, …, $МО_M$) или Y=(y_1,…,y_M),
- множество (список) ЛУ=СЛУ=($ЛУ_1$, …, $ЛУ_L$) или X=(x_1,…,x_L),
- множество слов S_1, …, S_K (операндов, результатов, внутренних слов).

Микрооперация описывается *оператором присваивания*:

$$\mathbf{S_i:=\varphi_m(S_j,S_n)}, \qquad (2)$$

где φ_m – некоторая вычислимая функция (например, сложение);

S_j, S_n – её аргументы (слова-операнды);

S_i – значение функции φ_m, вычисленное при заданных значениях аргументов $S_j=S^*_j$, $S_n=S^*_n$ (например, $S^*_j=5$, $S^*_n=10$, $S_i=5+10=15$) и присвоенное слову S_i (S_i – слово-результат).

Выполнение $МО_m \in$ СМО в ОА осуществляется под управлением сигнала $y_m \in$ Y, поступающего из УА (рис. 15). Поэтому микрооперация $МО_m$ обычно отождествляется с управляющим сигналом y_m, который возбуждает её выполнение в ОА – y_m:$S_i:=\varphi_m(S_j,S_n)$. Поэтому набор Y=(y_1,…,y_M) рассматривается и как список микроопераций СМО, и как список сигналов Y.

Логическое условие описывается выражением:

$$\mathbf{x_l := \psi_l(S_i)}, \qquad (3)$$

где ψ_l (пси эль) – булева функция;

$\mathbf{S_i}$ – ее аргумент;

$\mathbf{x_l}$ – значение функции ψ_l, вычисленное при $\mathbf{S_i = S_i^*}$.

Например, сигнал $\mathbf{x_l}=1$, если значение слова $\mathbf{S_i^*=0}$, или $\mathbf{x_l}=0$, если $\mathbf{S_i^* \neq 0}$. Набор значений $X=(x_1,\ldots,x_L)$ отображает состояние ОА.

Для хранения значений слов $S_1, \ldots, S_K$ используются одноименные регистры ОА, количество которых равно К. В этом случае МО вида (2) будет выполняться за один такт. Однако в целях экономии оборудования в ОА можно использовать всего один регистр-аккумулятор (как в ЭВМ с аккумуляторной архитектурой). В этом случае для хранения остальных слов используются ячейки памяти, поэтому в алгоритме операции $f_g \in F$ появляются дополнительные (вспомогательные) действия: загрузить (операнд в регистр), записать (результат в ячейку), увеличивающие время выполнения операции $f_g \in F$ (количество тактов).

Максимальная производительность ОА первого типа (I-автомата) обеспечивается тем, что его структура организуется в виде, который позволяет одновременно (в одном такте) выполнять все функционально совместимые МО. Для I–автомата характерно, что каждый из регистров $S_1, \ldots, S_K$ обслуживается своей комбинационной схемой $\Phi_1, \ldots, \Phi_K$, средствами которых реализуются $МО_!, \ldots, МО_K$, вычисляющие значения соответствующих слов. Отсюда максимальная производительность, которая при наличии К регистров и К комбинационных схем $\Phi_1, \ldots, \Phi_K$ в принципе может достигать К микроопераций за такт.

В структуре I-автомата могут содержаться эквивалентные по функциям КС, используемые для обслуживания различных регистров. Следовательно, затраты оборудования в комбинационной части ОА можно минимизировать, если каждую такую комбинационную схему обобщить по отношению ко всем регистрам, т.е. если использовать одну схему для выполнения всех эквивалентных МО. ОА, построенные на основе *принципа обобщения* комбинационных схем, используемых для выполнения МО, называются М-автоматами. В них для вычисления любого двоичного выражения $\varphi_m(S_{\alpha 1}, \ldots, S_{\alpha k})$ используется одна (универсальная) комбинационная схема Ф, равнодоступная по отношению к регистрам $S_1, \ldots, S_K$ ОА. Структура М-автомата представлена на рис. 16.

Операнды, участвующие в $МО_m$, подаются на входы А и В универсальной КС Ф по шинам А, В с помощью мультиплексоров MSA, MSB. Для выборки слов на шину А (для управления схемой MSA) используются управляющие сигналы $a_1, \ldots, a_K$, а для выборки слов на шину В (для управления MSB) – сигналы $b_1, \ldots, b_K$. Сигнал a_i инициирует микрооперацию передачи $A:=S_i$, а сигнал b_j – микрооперацию $B:=S_j$. Многофункциональная схема Ф на выполнение микрооперации $C:=\varphi_m(A,B)$ настраивается сигналом y_m, m=1, … , M, управляющим мультиплексором MSC. Результат $МО_m$ с выхода С схемы Ф заносится в регистр S_n сигналом c_n, управляющим демультиплексором DMC.

Чтобы выполнить микрооперацию $S_n:=\varphi_m(S_i,S_j)$, необходимо подать набор управляющих сигналов (a_i, b_j, y_m, c_n), под воздействием которых на входы КС φ_m подаются слова S_i, S_j. Над ними выполняется преобразование φ_m, и результат загружается в регистр S_n. Для выполнения унарной МО (например, микрооперации счета $S_n:=S_n+1$) ни один из сигналов b_j не вырабатывается (операнд B=0) и схема Ф реализует МО счета (увеличение операнда S_n на 1).

Загрузка операндов в регистры ОА извне (с шины D) осуществляется путем их подачи на вход B схемы Ф под управлением сигнала b_D B:=D. Одновременно на вход A подается константа ноль (по сигналу a_0), а в схеме Ф возбуждается МО «ИЛИ» C:=AvB (0vD), результат которой по сигналу c_n записывается в регистр S_n. Выдача результата операции $f_g \in F$ на выходную шину R осуществляется по сигналу c_R.

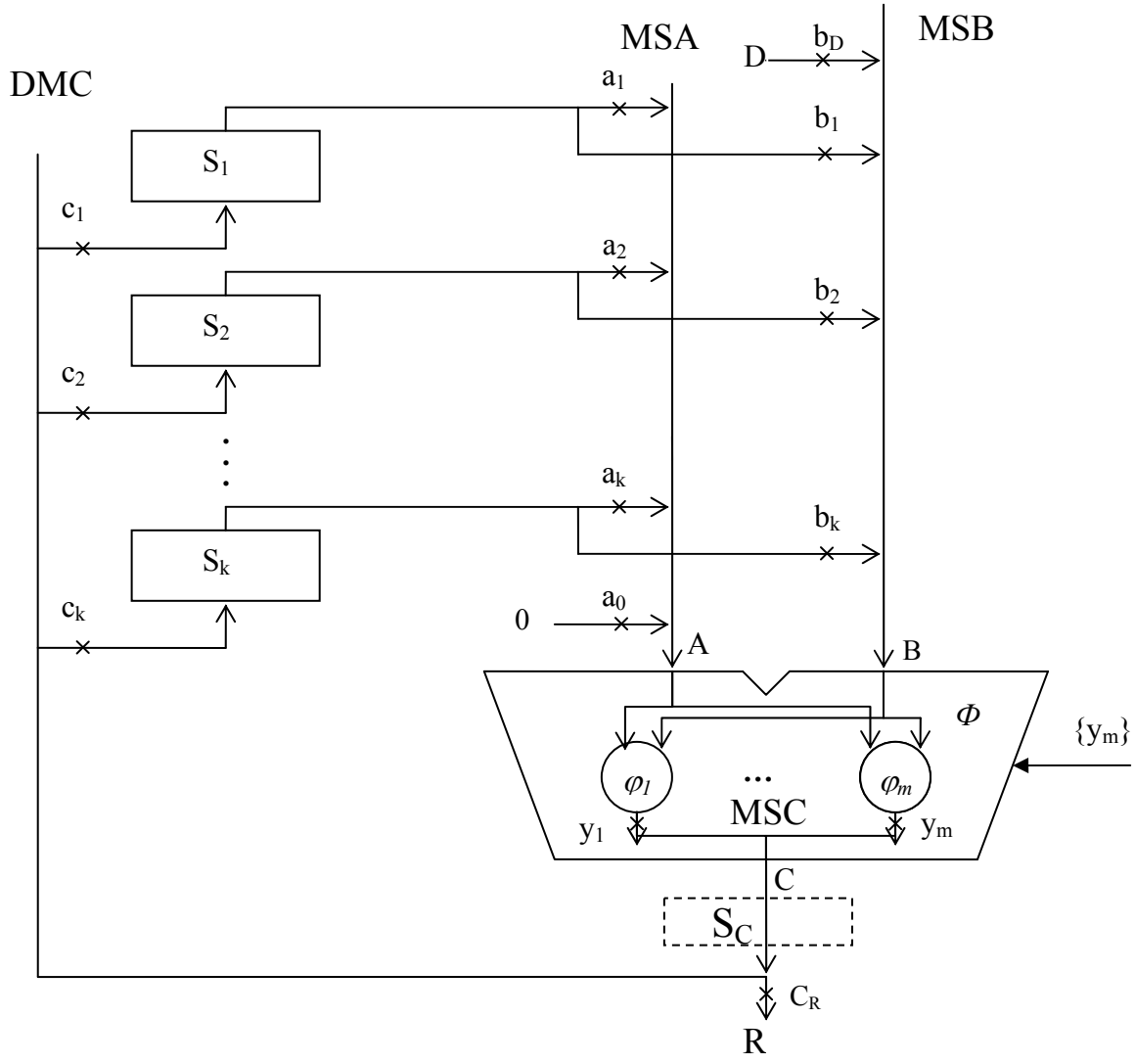


Рисунок 16 – Структурная организация М-автомата

Как видно из рис. 16, в каждом такте М-автомат может выполнить только одну МО вида $S_n:=\varphi_m(S_i,S_j)$, т.е. производительность М-автомата минимальная (одна МО за один такт). Быстродействие М-автомата отличается от быстродействия I-автомата незначительно, поскольку длительность такта М-автомата увеличивает-

ся не более чем на 2τ за счет введения в схему мультиплексоров MSA, MSB. Затраты оборудования в М-автомате минимальны, поскольку каждая схема $\varphi_i \in \Phi$ используется для выполнения всех эквивалентных МО.

Для обеспечения устойчивости (*устранения «гонок*») в М-автомате регистры $S_1, \ldots, S_K$ можно выполнить на основе двойных триггеров типа MS (Master Slave). Количество оборудования в каждом регистре при этом, по крайней мере, удваивается. Более экономичной будет схема, в которой в выходную шину С (на входе демультиплексора DMC) устанавливается вспомогательный регистр S_c (одинарный, типа D). Запись информации в него осуществляется в середине такта (по инверсному сигналу ТИ). Регистры $S_1,\ldots, S_K$ при этом также организуются на более простых одинарных триггерах типа D. Запись информации (обновление) в них осуществляется в начале такта (по прямому сигналу ТИ).

Итак, структурная организация I-автоматов базируется на принципе закрепления комбинационных схем, используемых для выполнения МО, за каждым из регистров $S_1,\ldots, S_K$. За счёт этого все функционально совместимые МО могут выполняться параллельно в одном такте, и их результаты записываются в разные регистры. Структурная организация М-автоматов базируется на обобщении КС по отношению ко всем регистрам, за счёт чего уменьшаются затраты оборудования.

Эти два класса автоматов обладают диаметрально противоположными свойствами. Для I-автоматов характерна максимальная производительность и наибольшие затраты оборудования. М-автоматам присуща минимальная производительность при наименьших затратах оборудования. Следует ожидать, что между этими двумя классами структур ОА лежат варианты структур ОА (IM-автоматы), обладающие промежуточными свойствами, а именно: достаточно высокой (не минимальной) производительностью при умеренных (не максимальных) затратах оборудования. ОА, структурная организация которых вносит некоторые ограничения на совместимость МО и, одновременно с этим, обеспечивает выполнение более чем одной МО за такт, называют IM-автоматами. Организация IM-автоматов также базируется на принципе обобщения эквивалентных КС, однако степень обобщения не столь высока, как в М-автоматах, и позволяет в одном такте выполнять более сложные действия, чем $S_i :=^* S_j$, $S_i := S_j * S_n$, («*» – знак операции), типичные для М-автоматов.

Структурная организация IM-автоматов базируется на принципе использования (для выполнения МО) последовательного и параллельного соединения многофункциональных КС $\Phi_1, \Phi_2, \ldots$. Первый принцип приводит к структурам IM-автоматов с *последовательной комбинационной частью* (рис. 17), а второй – к IM-автоматам с *параллельной комбинационной частью* (рис. 18).

Принцип последовательной организации приводит к структурам, в которых комбинационная часть состоит из нескольких (обычно трёх) схем Φ_1, Φ_2, Φ_3, реализующих микрооперации из трёх множеств $\{f_k\}$, $\{g_l\}$, $\{h_m\}$ соответственно (рис. 17). Схема Φ_1 обычно служит для формирования констант, полей, инверсий кодов, возбуждаемых одним из сигналов множества $\{f_k\}$, и называется формирователем кодов (для схемы Φ_2). Схема Φ_2 обычно служит для выполнения бинарных операций (основной из которых является сложение), возбуждаемых одним из сигналов множества $\{g_l\}$. Схема Φ_3 используется для выполнения унарных МО сдви-

га, возбуждаемых одним из сигналов множества $\{h_m\}$, и называется сдвигателем. В одном такте такой ОА может выполнить преобразование $S_n:=h_m(g_l(S_i,f_k(S_j)))$, эквивалентное трем последовательно выполняемым МО f_k, g_l, h_m. Выбор операндов из регистров S_i, S_j и запись результата в регистр S_n обеспечивается мультиплексорами MSA, MSB и демультиплексором DMC под управлением сигналов их наборов $\{y_A\}$, $\{y_B\}$, $\{y_C\}$. В результате производительность IM-автомата в три раза превышает производительность M-автомата. Однако продолжительность такта больше вследствие последовательного соединения схем Φ_1, Φ_2, Φ_3.

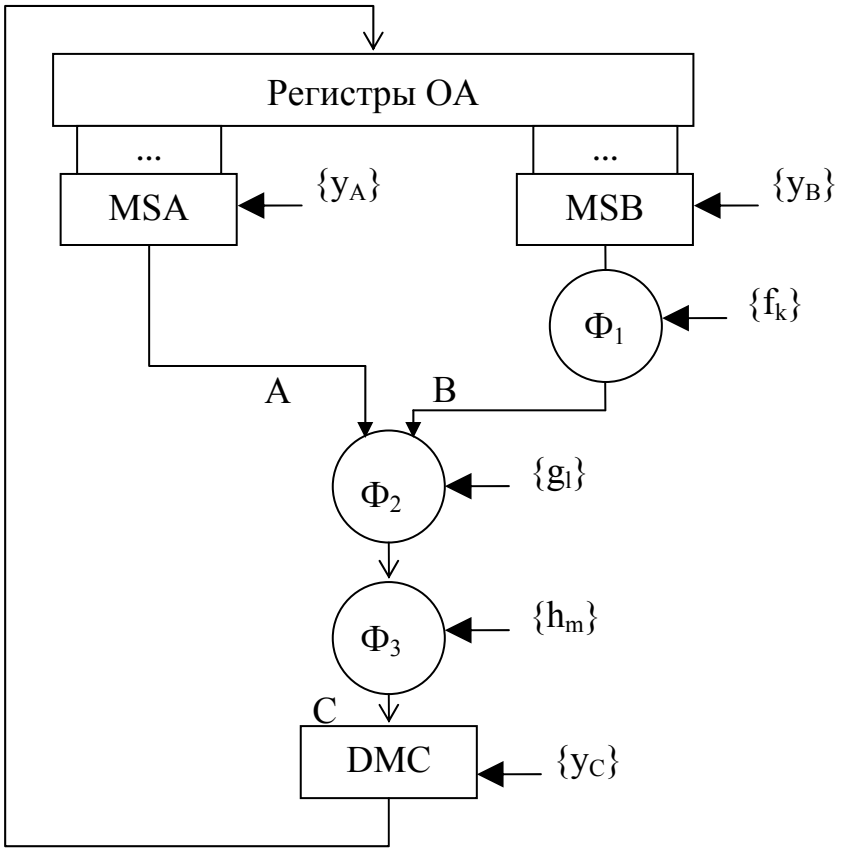


Рисунок 17 – Структура IM-автомата с последовательной комбинационной частью

Принцип параллельной организации приводит к структурам, в которых комбинационная часть состоит из нескольких (например, двух) схем Φ_1, Φ_2, реализующих микрооперации из множеств $\{f_n\}$, $\{g_h\}$ (рис. 18). Схема Φ_1 обычно служит для выполнения бинарных операций, возбуждаемых одним из сигналов множества $\{f_n\}$. Схема Φ_2 используется для выполнения унарных МО, возбуждаемых одним из сигналов множества $\{g_h\}$. В одном такте такой ОА может выполнить две МО $S_n:=f_n(S_i,S_j)$, $S_r:=g_h(S_k)$. Выбор операндов из регистров S_i, S_j и запись результата в регистр S_n обеспечивается мультиплексорами MSA, MSB, MSC и демультиплексорами DM1, DM2, управляемых сигналами из наборов $\{y_A\}$, $\{y_B\}$, $\{y_C\}$, $\{y_{D1}\}$, $\{y_{D2}\}$. В результате производительность такого IM-автомата в два раза превышает производительность M-автомата. IM-автомат с параллельной

организацией можно рассматривать как композицию из нескольких (двух и более) М-автоматов, имеющих общую память $S_1, \ldots, S_K$.

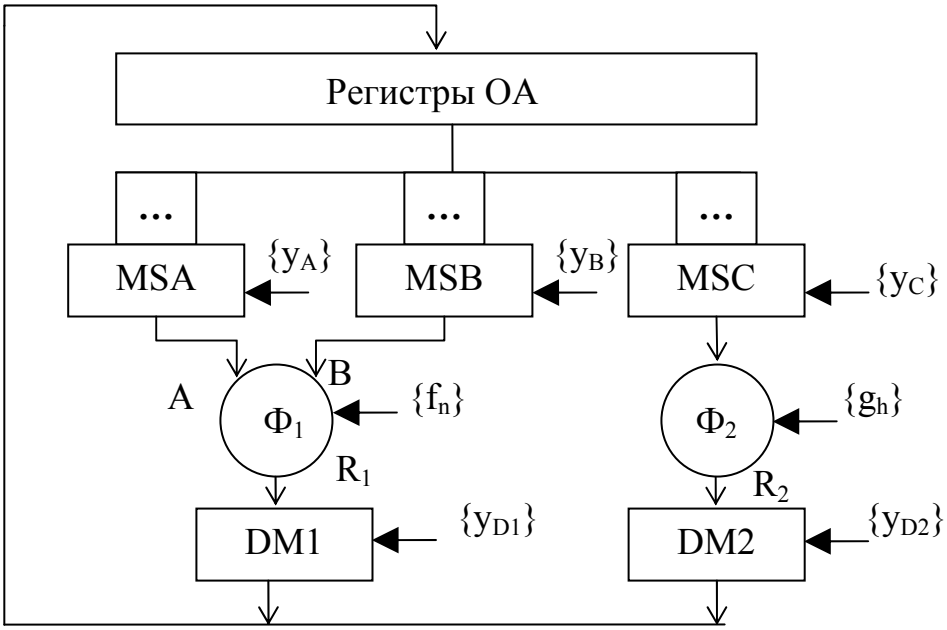


Рисунок 18 – Структура IM-автомата с параллельной комбинационной частью

3.1.2 Схема лабораторного макета операционного автомата арифметико-логического устройства

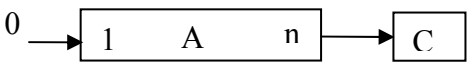
В качестве примера рассмотрим IM-автомат с последовательной комбинационной частью, функции которого заданы списками МО, ЛУ, количеством слов К и их разрядностью n в памяти ОА.

Список арифметико-логических микроопераций $Y_1=\{y_m\}$, $m\in(1, \ldots, 8)$, A(1:n), B(1:n) – слова-операнды, F(1:n) – слово-результат:

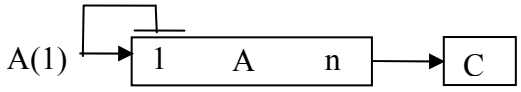
1) F:=A+B+C_{in} – сложение, C_{in} – вход переноса в младший разряд сумматора;
2) F:=A-B-1+C_{in} – вычитание (сложение с дополнительным кодом B);
3) F:=B-A-1+C_{in} – вычитание;
4) F:=A v B – логическое сложение (дизъюнкция);
5) F:=A&B – логическое умножение (конъюнкция);
6) F:= $\overline{A}\vee B$ – инверсия, если B=0;
7) F:=A$\oplus$B – исключающее ИЛИ;
8) F:=A $\overline{\oplus}$B – сравнение на равенство.

Список МО сдвига $Y_2=\{y_k\}$, $k\in(9, \ldots, 16)$, (A(1:n) – операнд, B(1:n) – результат, C – бит переноса, сформированный ранее и занесенный в триггер C регистра признаков результата):

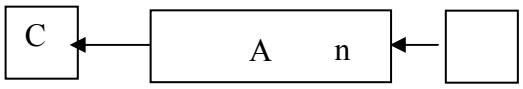
1) C.B:=C.A – нет сдвига;
2) B.C:= R1(0.A) – сдвиг вправо на 1 разряд с обнулением старшего разряда и с сохранением младшего разряда в C (логический сдвиг вправо);



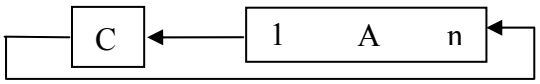
3) B.C:= R1(A(1).A) – сдвиг вправо на 1 разряд с сохранением старшего разряда (знака) A(1) и младшего разряда в C (арифметический сдвиг вправо);



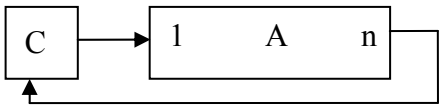
4) C.B:= L1(A.0) – сдвиг влево на 1 разряд с сохранением старшего разряда в C и обнулением младшего разряда;



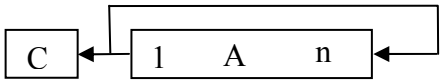
5) C.B:= L1(A.C) – циклический сдвиг влево на 1 разряд через C;



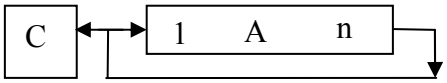
6) B.C:= R1(A.C) – циклический сдвиг вправо на 1 разряд через C;



7) C.B:= L1(A.A(1)) – циклический сдвиг влево на 1 разряд с сохранением старшего разряда A(1) в C;



8) B.C:= R1(A(0).A) – циклический сдвиг вправо на 1 разряд с сохранением младшего разряда A(n) в C;



Список (типичных) логических условий:

1) Z(zero) – признак (флаг) нулевого результата. Значение Z:=1, если результат F арифметико-логической МО равен 0, иначе (т. е. если F≠0) Z:=0;

2) S:=F(1) – значение старшего разряда результата F (при обработке чисел со знаками трактуется как знак (sign) результата);

3) OVR – признак переполнения. Значение OVR:=1, если результат F арифметической МО переполняет n-разрядную сетку, OVR:=0 – если не переполняет;

4) C:=C_{OUT} – значение бита переноса из старшего разряда сумматора при выполнении операций сложения и вычитания (бит Carry).

При выполнении арифметических операций формируются все четыре ЛУ, при выполнении логических операций – только два – Z, S, сигналы C, OVR не изменяются. При выполнении операций сдвига могут измениться все четыре ЛУ.

Типичная структура IM-автомата с последовательной комбинационной частью представлена на рис. 3*а*. Память S аналогична памяти M-автомата (на рис. 16). Комбинационная часть ОА АЛУ лабораторного макета (рис. 19) состоит из универсальной комбинационной схемы UKS и сдвигателя SH, соединенных последовательно, а также формирователя логических условий FLU.

Схема UKS (рис. 20) обеспечивает выполнение МО из списка арифметико-логических операций под воздействием управляющих сигналов y_1, …, y_8, поступающих из УА АЛУ на входы $\{y_m\}$.

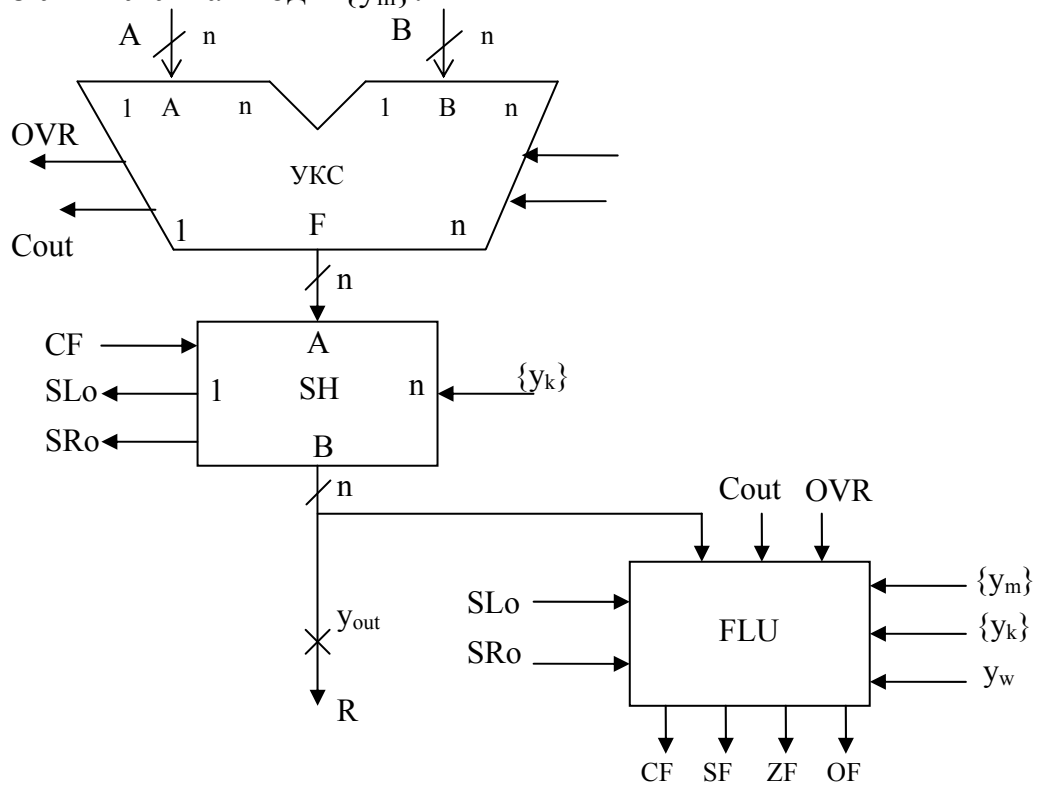


Рисунок 19 – Схема комбинационной части ОА

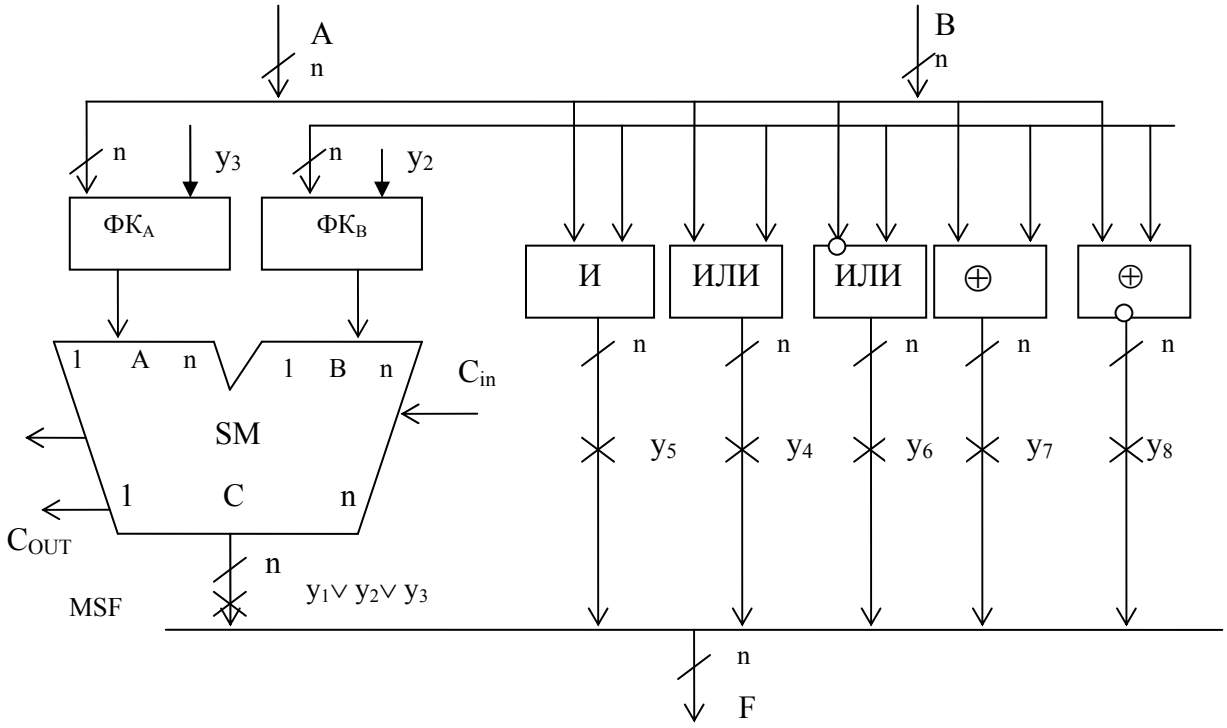


Рисунок 20 – Схема UKS

Сигнал y_1 возбуждает МО сложения (первую МО в списке) y_1:F:=A+B+C_{in}, сигнал y_2 – вторую и т.д. по списку Y_1= (y_1, …, y_8). Сигналы y_1,…,y_8 – это импульсы, продолжительность которых равна длительности такта ОА (остается неизмен-

ной, пока идет выборка операндов, выполнение МО, сохранение результата – см. временную диаграмму, рис. 21).

Схемы $ФК_A$ и $ФК_B$ (формирователи кодов на входах A и B сумматора SM) обеспечивают инвертирование операнда при выполнении операции вычитания по сигналам y_2, y_3 соответственно. Выполнены они на основе двухвходовых элементов «исключающее или» в количестве n штук, на вторые входы которых подаётся сигнал y_2 (или y_3). Крестик на соответствующих шинах, возле которого указан символ y_m (например, y_5), – это условное обозначение управляемой (этим сигналом) шины.

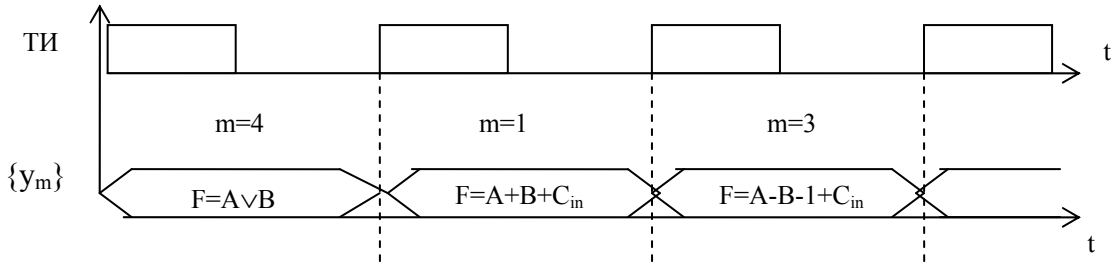


Рисунок 21 – Временная диаграмма работы UKS

Схема сдвигателя SH (рис. 22) обеспечивает выполнение МО сдвига из списка Y2 под воздействием управляющих сигналов $y_9,\ldots,y_{16}$, поступающих из УА АЛУ на входы $\{y_k\}$ сдвигателя. Сигнал y_9 возбуждает первую МО в этом списке (нет сдвига), сигнал y_{10} – вторую (сдвиг вправо) и т.д. по списку. Входы SL_{in}, SR_{in} представляют собой входы сдвигателя, используемые при левом (SL_{in}) и правом (SR_{in}) сдвигах. Выходы SL_0, SR_0 используются соответственно при левом и правом сдвигах.

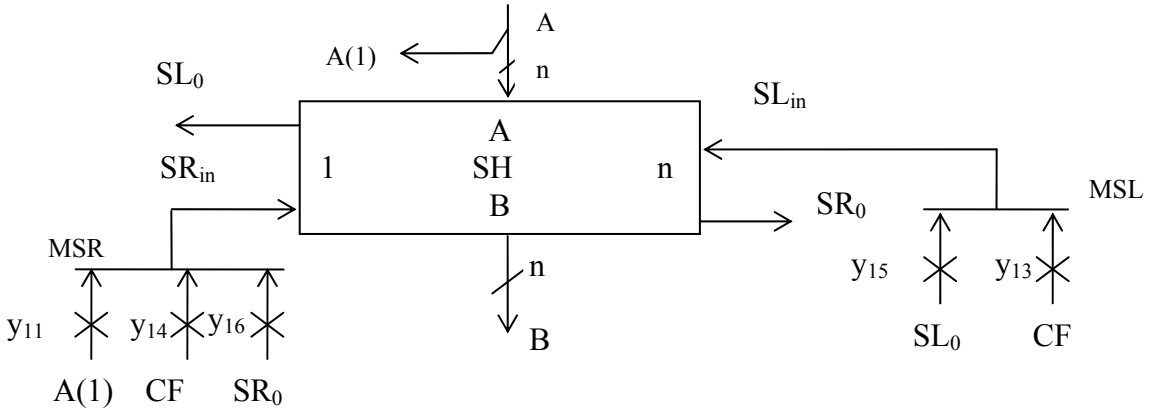


Рисунок 22 – Схема сдвигателя SH

Нужная МО сдвига обеспечивается мультиплексорами правого (MSR), левого (MSL) сдвига и мультиплексором MSC, установленным на входе триггера TC, в котором сохраняется значение бита C. Мультиплексор – это совокупность управляемых шин с общим выходом. Микрооперации, выполняемые мультиплексором (например, y_{11}, y_{14}, y_{16} для MSR), являются несовместимыми и призваны обеспечивать коммутацию одного из входов на единственный выход. Например, сигнал $y_{11}=1$ подает старший бит A(1) сдвигаемого слова A на вход SR_{in} при вы-

полнении МО y_{11}:B.C:=R1(A(1).A)) – сдвиг правый арифметический.

Формирователь логических условий FLU (рис. 23) обеспечивает формирование сигналов Z, C, S, OVR и сохранение их в регистре флагов RF.

Формирование ЛУ осуществляется по следующим правилам. Если выполняется только АЛО микрооперация (без последующего сдвига ее результата), то формируются и заносятся в RF все четыре ЛУ. Если выполняется логическая микрооперация (без последующего сдвига ее результата), то формируются и заносятся в RF только Z, S (флаги CF, OF не изменяются). Если выполнение МО в УКС совмещается с микрооперацией сдвига, то формирование ЛУ осуществляется на основе результата МО сдвига на выходе сдвигателя. Флаг OF устанавливается в единицу, если в результате сдвига изменился знаковый (старший) разряд результата.

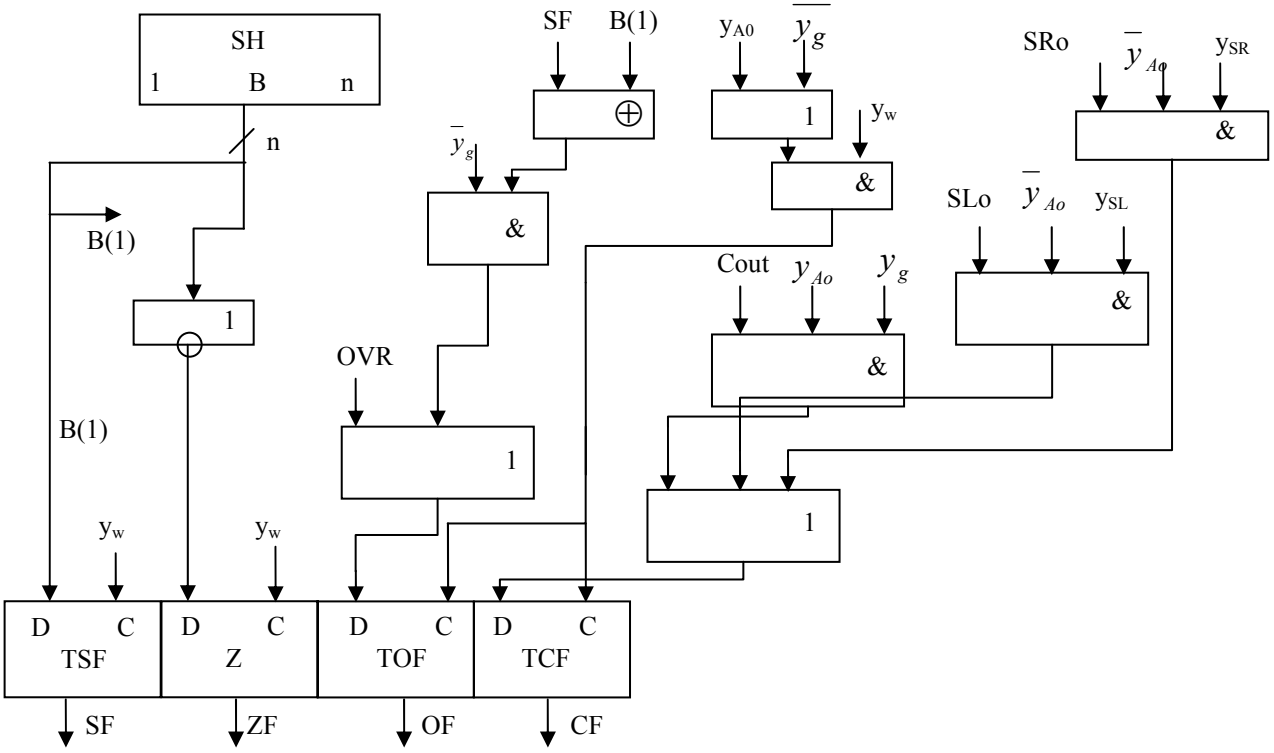


Рисунок 23 – Схема FLU

При выполнении арифметических МО формируется сигнал $y_{AO}=y_1 \vee y_2 \vee y_3$. Сигналы $y_{SL}=y_{12} \vee y_{13} \vee y_{15}$, $y_{SR}=y_{10} \vee y_{11} \vee y_{14} \vee y_{16}$ формируются при выполнении соответствующих МО левого и правого сдвига. Для формирования сигнала Z используется схема ИЛИ-НЕ. Сигнал S (тривиально) определяется значением B(1), т. е. S= B(1). Сигналы OVR и C_{OUT} формируются в сумматоре SM.

Память S ОА (регистры $S_1,\ldots,S_K$) выполнена в виде блока RON (регистров общего назначения (РОН)) по схеме запоминающего устройства (рис. 24). Доступ к ячейкам (регистрам) осуществляется по адресам (номерам) регистров 1, ..., К, подаваемым на адресные входы A, B.

Такая организация эквивалентна схеме, представленной на рис. 16. Однако в ней, с целью экономии оборудования, регистры $S_1,\ldots,S_K$ выполнены на основе одинарных (синхронизируемых) D-триггеров, а для обеспечения устойчивости

ОА (на входе IN блока РОН) поставлен буферный регистр BR, также построенный из одинарных D-триггеров.

Запись слова, подаваемого на вход IN (по адресу на входе B), осуществляется по сигналу записи W демультиплексором, функции которого возбуждаются сигналами B_1, …, B_K с выходов дешифратора DCB, подключенными к входам

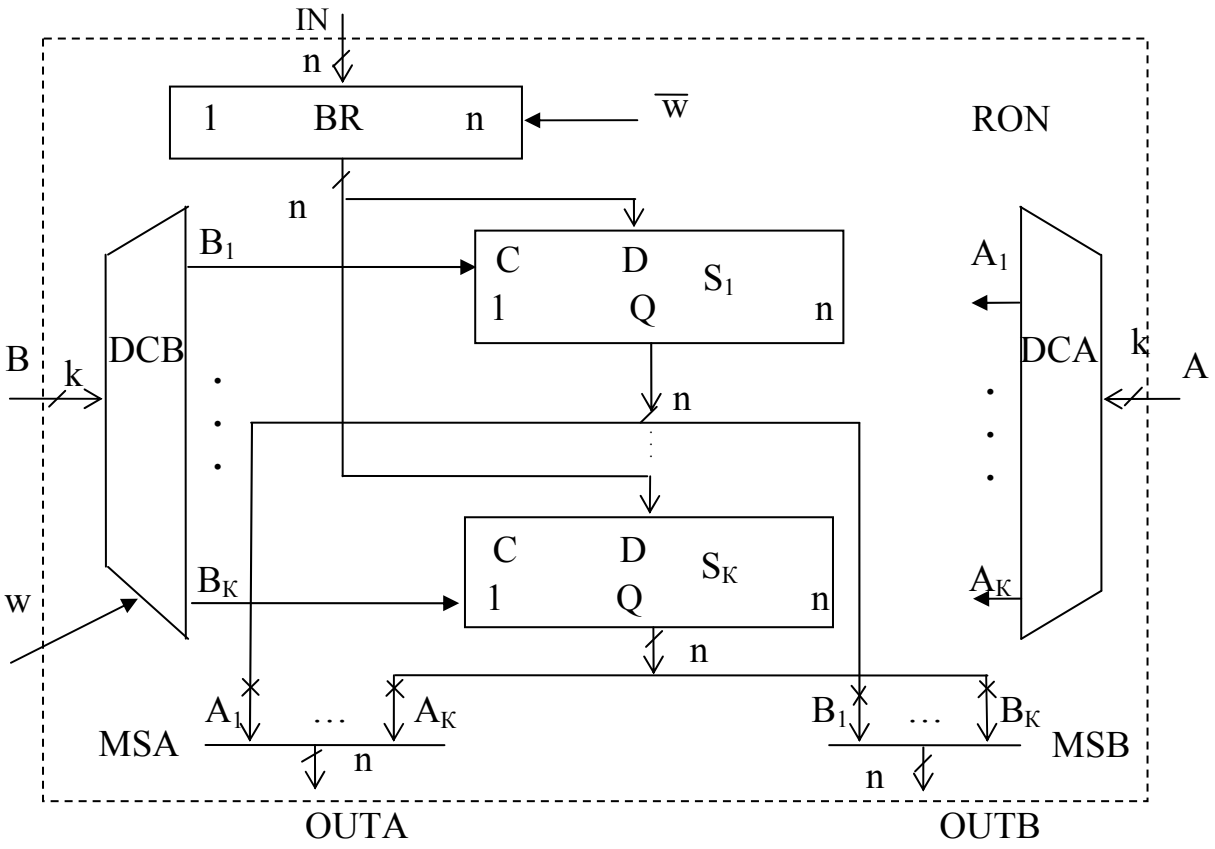


Рисунок 24 – Структура блока РОН ОА АЛУ

синхронизации C регистров S_1, …, S_K соответственно. Сигнал записи (импульс) W передаётся на вход C адресуемого регистра в виде сигнала (импульса) B_k с соответствующего выхода DCB, то есть сигнал записи W возбуждает МО записи: $W:S_k:=IN$ (или W:[B]:=IN).

Таким образом, список МО записи результатов в регистры (S_1,…, S_K) $c_1:S_1:=IN$, …, $c_K:S_K:=IN$, возбуждаемых сигналами c_1,…,c_K в структуре ОА, представленной на рис. 16, в блоке РОН заменяется одной МО записи вида «W:[B]:=IN», возбуждаемой сигналом W. Время выполнения МО записи $\tau_{зап}=\tau_{DC}+\tau_{TP}=\tau+3\tau=4\tau$. Чтение слов из регистров (выдача их на выходы OUTA, OUTB) осуществляется с помощью мультиплексоров MSA, MSB, управляемых сигналами A_1, …, A_K, B_1, …, B_K с выходов дешифраторов DCA, DCB, на входы которых подаются адреса A, B регистров.

Микрооперации передачи слов на шины A, B $a_1:A:=S_1$, …, $a_K:A:=S_K$, $b_1:B:=S_1$, …, $b_K:B:=S_K$, возбуждаемые сигналами a_1, …,a_K, b_1, …,b_K в структуре ОА (рис. 16), в блоке РОН заменяются двумя МО вида – OUTA:=[A], OUTB:=[B]. Изменение адресов на входах A, B изменяет значения слов на выходах OUTA, OUTB с задержкой $\tau_1+\tau_2$ на выходе OUTB (по отношению к сигналу W) и с задержкой τ_1 по отношению к моменту изменения адреса на адресном входе (временная диа-

грамма, рис. 25). Это означает, что на выход OUTA (OUTB) выдается значение слова S_k, k=A, (k=B) до тех пор, пока не изменится адрес на входе A (входе B), либо пока не изменится содержимое регистра S_k в момент записи по адресу B. Задержка $\tau_2=3\tau$ – это задержка в дешифраторе адреса и мультиплексоре

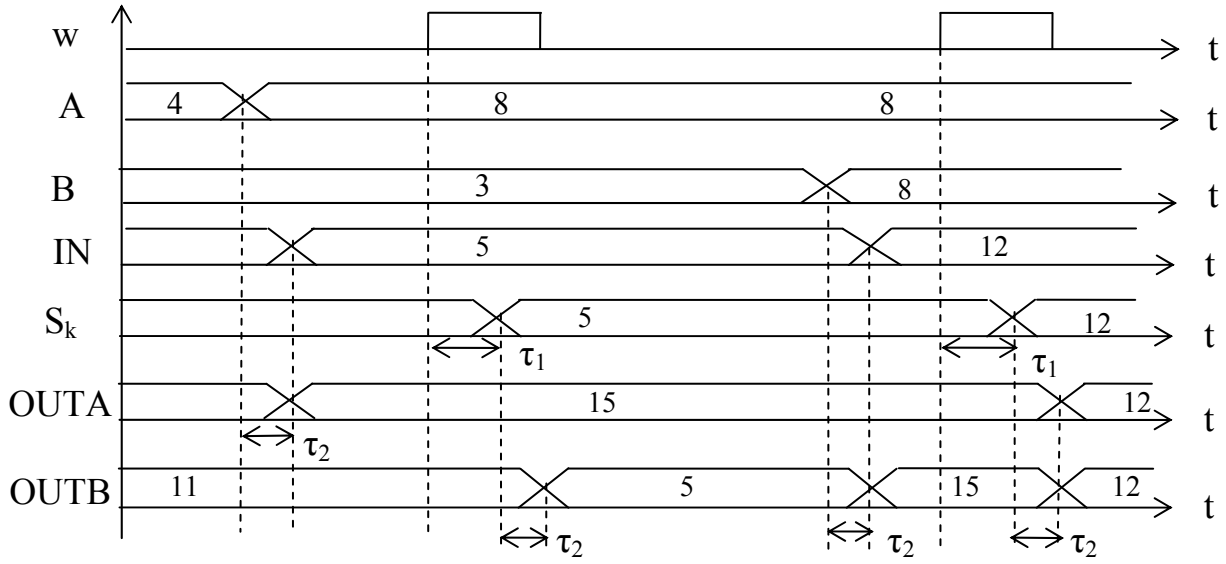


Рисунок 25 – Временная диаграмма работы РОН ОА АЛУ

MSA (MSB), задержка $\tau_1=\tau_{зап}=4\tau$ – это время переключения триггеров регистра при записи, где τ – время переключения логического элемента.

Полная укрупненная схема (лабораторного макета) ОА АЛУ представлена на рис. 26.

ОА АЛУ способен выполнять МО двух типов. Микрооперация первого типа эквивалентна микрооперации $S_n=\varphi_m(S_i,S_j)$, возбуждаемой (в схеме ОА на рис. 2) сигналами (a_i, b_j, y_m, c_n). Здесь она возбуждается набором сигналов y_0, y_D, обеспечивающих подачу операндов A, B на входы A, B схемы UKS, сигналом $y_m \in Y_1$ из набора Y_1, обеспечивающим выполнение АЛО МО, сигналом $y_k \in Y_2$, обеспечивающим МО сдвига, и сигналом $y_W=1$, обеспечивающим запись результата МО в регистр с номером A2. Если сигнал $y_W=0$, то результат МО (второго типа) не записывается в регистр, что равносильно отсутствию МО (нет операции – NOP). Таким образом, запись результата в регистр осуществляется в каждом такте работы ОА, за исключением тех тактов (случаев), в которых, например, выполняются микрокоманды перехода.

Регистр RF выполнен на основе двойных триггеров типа MS. Запись признаков результатов S, Z, OVR, C в регистр флагов RF осуществляется (как в РОН) по сигналу y_W (за исключением пустых тактов – NOP в ОА, когда $y_W=0$). Следует отметить, что запись флагов Z, OVR осуществляется только при выполнении арифметических МО. Запись флага S осуществляется при выполнении арифметических и логических МО, а запись флага C – при выполнении арифметических МО и соответствующих МО сдвига.

По завершении операции $f_g \in F$ в АЛУ ее результат выдается на выходную шину R под управлением сигнала y_{OUT}: R:=B (B – выход схемы SH).

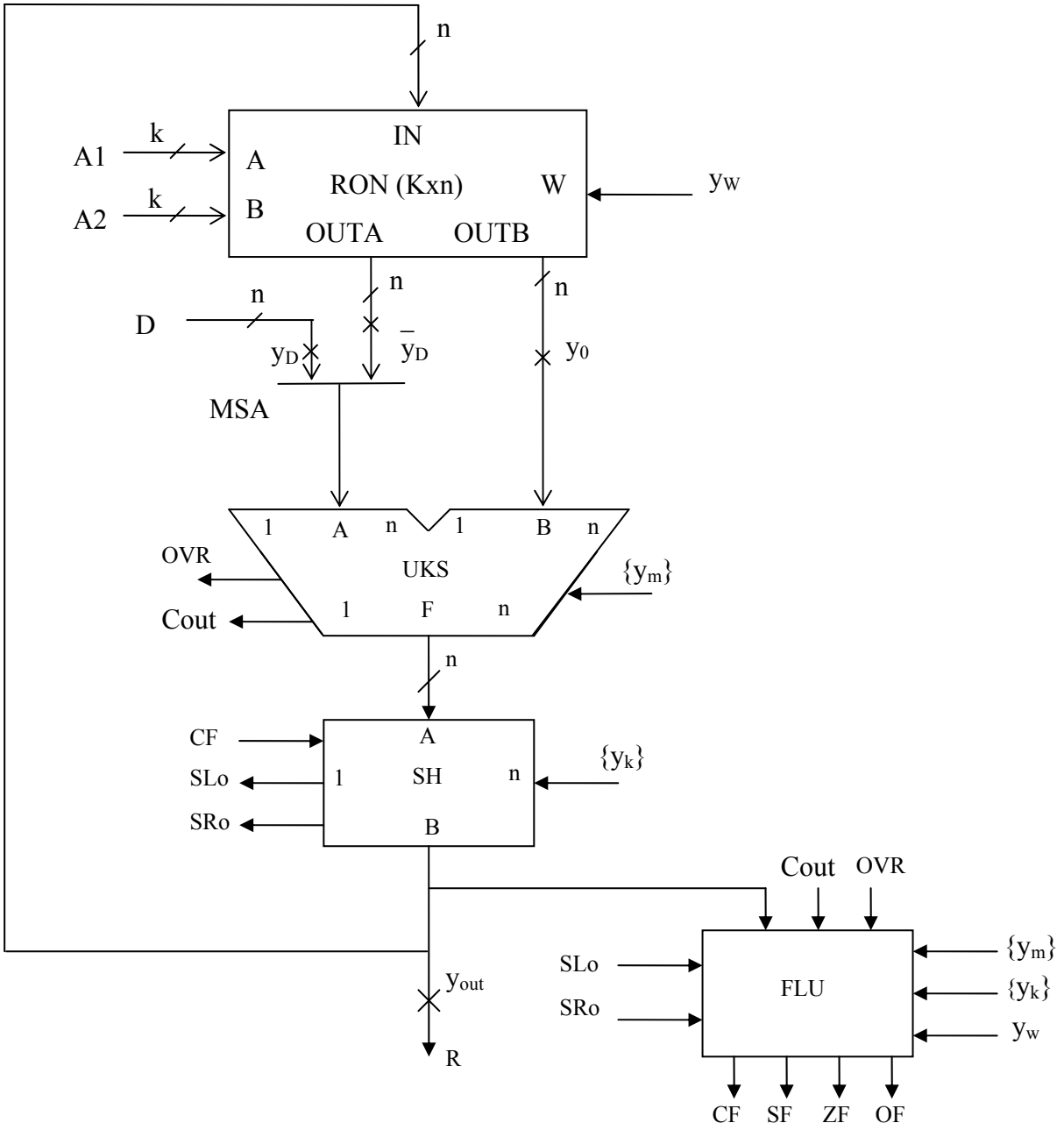


Рисунок 26 – Схема лабораторного макета ОА АЛУ

Таким образом, для настройки ОА (рис. 26) на выполнение нужных действий на его управляющие входы (перечисленные выше) необходимо подавать соответствующие сигналы (из УА АЛУ). Полный набор сигналов (формат микрокоманды для ОА) представлен на рис. 27.

C_{in}	y_0	y_D	1 ALO 3	1 SH 3	y_W	y_{OUT}	1 A1 k	1 A2 k

Рисунок 27 – Формат микрокоманды ОА АЛУ

Поля (биты) y_0 ,y_D обеспечивают подачу операндов на входы A, B UKS в четырех комбинациях (табл. 1).

Таблица 1 – Источники операндов

y_0	y_D	A	B
0	0	OUT A	0
0	1	D	0
1	0	OUT A	OUT B
1	1	D	OUT B

Поле ALO указывает (задает) АЛО МО $y_m \in Y_1$, список которых приведен в табл. 2. Поле SH задает МО сдвига $y_k \in Y_2$, список которых приведен в табл. 3. В поле C_{in} размещается значение бита переноса (на входе C_{in} сумматора), которое используется при выполнении операций сложения и вычитания.

Таблица 2 – АЛО микрооперации

Код ALO	Микрооперация
000	$F=A+B+C_{in}$
001	$F=A-B-1+C_{in}$
010	$F=B-A-1+C_{in}$
011	$F=A \vee B$
100	$F=A\&B$
101	$F=\overline{A}+B$
110	$F=A \oplus B$
111	$F=A \overline{+} B$

Таблица 3 – МО сдвига

Код SH	Микрооперация
000	C.B=C.A (NOP)
001	B.C=R1(0.A)
010	B.C=R1(A(1).A)
011	C.B=L1(A.0)
100	C.B=L1(A.C)
101	B.C=R1(A.C)
110	C.B=L1(A.A(1))
111	B.C=R1(A(0).A)

В поле y_W помещается 1, если результат нужно записать в РОН по адресу A2, указанному в поле A2. Поля A1, A2 используются для указания местоположения операндов, извлекаемых из памяти ОА. Поле A2, кроме того, используется и для записи результата (формат МК – двухадресный). Разрядность k полей A1, A2 зависит от количества K регистров в памяти ОА, остальные поля (слева от A, B) имеют фиксированную длину.

Таблица 4 –
Приемники результата

y_W	y_{OUT}	Микро-операция
0	0	NOP
0	1	R=B
1	0	[A2]=B
1	1	[A2]=B, R=B

Поля (биты) y_W ,y_{OUT} обеспечивают выдачу результатов (с выхода схемы SH) на выход R в четырех комбинациях (табл. 4).

Максимальная производительность ОА – две МО (АЛО и сдвига) за 1 такт, продолжительность такта ОА:

$$T_{OA}=\tau_{BO}+\tau_{UKS}+\tau_{SH}+\tau_{POH}, \quad (4)$$

где $\tau_{BO}= \tau_{OUT}+\tau_{MS}$ – время выборки (чтения) операндов из РОН и подачи их на входы UKS;

τ_{UKS} – максимальная задержка в схеме UKS;

τ_{SH} – максимальная задержка в схеме SH;

τ_{POH} – время записи результата в РОН.

3.2 Задание на самостоятельную работу

1. Изучить схему и операционные возможности ОА АЛУ.

2. Разработать микропрограмму в соответствии с вариантом задания, указанным преподавателем. Варианты заданий представлены в табл. 5. Выполнить микропрограмму в шаговом режиме. Результаты выполнения представить в виде временной диаграммы и трассы в виде таблицы.

3. Определить задержки сигналов от входов до выходов ОА и продолжительность такта Т (по максимальной задержке).

4. Написать отчет, который должен содержать задание, микропрограмму, временную диаграмму и трассу выполнения микропрограммы.

Таблица 5 – Микропрограммы для вариантов задания

Номер варианта	Задание	Источники операндов	Знаки и разрядность операндов			Особенности результата
			N1	N2	n	
1	*2*	*3*	*4*	*5*	*6*	*7*
1	R=N1-2N2	OUT A, OUT B	-	+	4	Переполнение
2	R=2N1+N2	OUT B, D	+	+	8	Переполнение
3	R=2N1-N2/2	OUT A, OUT B	+	+	12	Больше 0
4	R=2N1+N2/2	OUT B, D	-	+	16	Меньше 0
5	R=4N1-N2	OUT A, OUT B	+	+	4	Равен 0
6	R=N1-N2/2	OUT A, OUT B	+	-	8	Больше 0
7	R=N1/2-N2/4	OUT B, D	-	-	12	Меньше 0
8	R=2N1-N2/4	OUT B, D	+	-	16	Больше 0
9	R=2N1-4N2	OUT A, OUT B	-	+	4	Переполнение
10	R=2N1⊕N2/2	OUT A, OUT B			8	R=11111111
11	R=4N1&N2	OUT B, D			12	Не равен 0
12	R=N1/2v4N2	OUT B, D			16	Не равен 0
13	R=3N1-N2/2	OUT A, OUT B	-	+	4	Переполнение
14	R=(5N1)&N2	OUT A, OUT B			8	Не равен 0
15	R=2N1-N2/2	OUT B, D	+	-	12	Переполнение
16	R=2N1⊕N2/4	OUT B, D			16	Равен 0
17	R=3N1+N2/2	OUT B, D	-	+	4	Меньше 0
18	R=5N1-N2	OUT A, D	-	-	8	Больше 0
19	R=3N1-5N2	OUT A, OUT B	+	+	12	Больше 0
20	R=2N1-3N2	OUT A, OUT B	-	+	16	Переполнение
21	R=N1/2-3N2	OUT B, D	+	+	4	Равен 0
22	R=8N1 $\overline{\oplus}$N2	OUT B, D			8	R=11111111
23	R=2N1+3N2	OUT A, OUT B	-	-	12	Переполнение
24	R=5N2-N1/2	OUT A, OUT B	+	-	16	Переполнение

Проект по лабораторной работе №3 находится в папке «Лаб. работы `Организация АЛУ - Примеры выполнения программ для лаб. работы\2011\oa_full.dir». В нем для компиляции рекомендуется использовать файл ..\oa_full.gdf. Перед компиляцией рекомендуется выбрать опцию Assign | Device | FLEX10KA. Микропрограмму рекомендуется задавать в «полуавтоматическом режиме» в редакторе волновых фронтов, выбрав предварительно файл ..\oa_full.scf.

3.3 Содержание отчета

Отчет должен содержать цель работы, постановку задачи, блок-схему проектируемой схемы, блок-схему проекта, созданного в ходе выполнения лабораторной работы, а также временные диаграммы с заданными параметрами сигналов. Следует записать длительность периода тактовых импульсов. Также отчет должен содержать значение максимальной рабочей частоты проекта.

Следует сделать выводы по результатам работы.

3.4 Контрольные вопросы

1. Какой принцип лежит в основе построения операционных устройств (в частности, АЛУ)? Поясните основные положения этого принципа и организацию ОУ.
2. Назначение и основные характеристики АЛУ.
3. Организация работы ОУ во времени. Поясните понятия: такт работы, синхронная работа с постоянной длительностью такта.
4. От чего зависит и как определяется продолжительность такта?
5. Назначение и основные характеристики ОА.
6. Поясните понятия МО и ЛУ. Описание и реализация МО и ЛУ.
7. Функция и структура операционного автомата. Способы построения ОА.
8. Краткая характеристика I-автоматов, M-автоматов, IM-автоматов.
9. Поясните организацию и порядок работы лабораторного макета ОА АЛУ.
10. Поясните организацию блока РОН ОА АЛУ. Почему у него два информационных выхода, а вход – один?
11. Какие сигналы (входы) предназначены для управления работой блока РОН?
12. Объясните смысл и назначение сигналов S, Z, OVR, C схемы ОА.
13. Какие сигналы (входы) предназначены для настройки комбинационной части ОА на выполнение требуемых действий?
14. Какие сигналы (входы) предназначены для настройки ОА АЛУ на выполнение требуемых действий?
15. Поясните формат и назначение полей микрокоманды, предназначенной для управления ОА АЛУ.
16. Как оценить продолжительность такта T ОА АЛУ?
17. Поясните порядок выполнения работы на примере первого варианта задания.

4 ЛАБОРАТОРНАЯ РАБОТА №4. ОРГАНИЗАЦИЯ УПРАВЛЯЮЩЕГО АВТОМАТА АРИФМЕТИКО-ЛОГИЧЕСКОГО УСТРОЙСТВА

Цель лабораторной работы – изучение принципов построения управляющего автомата арифметико-логического устройства (УА АЛУ).

4.1 Теоретические основы лабораторной работы

4.1.1 Принципы построения управляющих автоматов

Функция УА определяется совокупностью закодированных графов микропрограмм Γ_1, …, Γ_G. *Закодированный граф* Γ_g получается из содержательного графа $МП_g$ путем замены микроопераций $МО_1$, …, $МО_M$, указанных в операторных вершинах $МП_g$, символами y_1, …, y_M и замены логических условий $ЛУ_1$, …, $ЛУ_L$, содержащихся в условных вершинах графа, символами x_1, …, x_L (кодированием микропрограммы $МП_g$). На основе графов Γ_1, …, Γ_G можно синтезировать соответственно G управляющих автоматов $УА_1$, …, $УА_G$, которые обеспечат управление ОА. Однако такое решение получается неэффективным. Графы различных МП обычно содержат некоторое число одинаковых операторных и условных вершин, которые сольются, если построить граф Г, объединяющий в себе графы Γ_1, …, Γ_G. Уменьшение числа вершин в МП влечет уменьшение затрат оборудования в УА. Поэтому функцию УА принято представлять объединенным графом Г.

Метод объединения закодированных графов Γ_1, …, Γ_G в единый граф Г, содержащий минимальное число вершин, изложен в работе [5].

В объединенной микропрограмме пути развития процессов выполнения операций f_1, …, f_G задаются кодом $g=g_1 \dots g_k$ операции f_g (где $g \in (1, \dots, G)$ – номер операции f_g), длина которого $k \geq \log_2 G$. Двоичные переменные g_1, …, g_k кода g в объединенной МП играют роль логических условий (как и условия x_1, …, x_L).

Для формирования управляющих сигналов Y, вырабатываемых в зависимости от значений g_1, …, g_k и x_1, …, x_L, можно использовать последовательностную логическую схему – (конечный) автомат с памятью. При этом множество входных сигналов X автомата определяется множеством осведомительных сигналов, поступающих из ОА, и битов g_1, …, g_k кода операции f_g: $X=x_1, \dots, x_L, g_1, \dots, g_k = (x_1, \dots, x_l, x_{l+1}, \dots, x_L)$, $L=l+k$. Множество выходных сигналов определяется множеством управляющих сигналов $Y=y_1, \dots, y_M$, возбуждающих микрооперации $МО_1$, …, $МО_M$ в ОА.

Закон функционирования конечного автомата задается объединенным графом Г и определяет порядок преобразования входной последовательности X(0), X(1), …, X(t) в выходную последовательность Y(0), Y(1), …, Y(t), где t – автоматное время. Способ реализации функции УА на основе принципа интерпретации микропрограммы автоматом с памятью приводит к построению *УА с жесткой логикой* управления (УА с ЖЛ) [4].

УА можно построить и иначе – на основе *принципа управления по хранимой* (в памяти) *микропрограмме* (принципа Уилкса). Такой способ приводит к по-

строению *УА с программируемой логикой* управления [4]. В соответствии с этим принципом для формирования управляющих сигналов используется (новое понятие) *микрокоманда* – это управляющее слово (разделенное на поля определенного назначения и фиксированной длины), которое определяет порядок функционирования (операционного) устройства в течение одного такта.

Микрокоманда (МК) содержит информацию о микрооперациях, которые должны выполняться в данном такте, а также информацию об адресе следующей МК. Совокупность МК, описывающих объединенную МП, образует массив микрокоманд МК[1:P], хранимый в памяти (в постоянном ЗУ – в ПЗУ).

Простая структура управляющего слова, достаточная для представления (описания) МК, имеет вид:

1 Y m	1 X h	1 A1 p	1 A2 p

Поле Y определяет номера микроопераций, возбуждаемых микрокомандой в ОА (в текущем такте). Если поле Y=0 (пустое), то УА не возбуждает ни одной МО. Номера МО в списке $y_0, y_1, \ldots, y_M, y_{M+1}$ можно кодировать m-разрядным позиционным кодом, $m \geq \log_2(M+2)$, где y_0 – пустая МО, y_{M+1} – сигнал об окончании операции f_g (конец микропрограммы МПg).

Для определения адреса следующей МК можно использовать принудительную адресацию МК (основанную на принципе связного списка). В этом случае в адресной части МК (в полях A1, A2) указываются адреса следующих МК. Адрес следующей МК можно задавать *безусловно* (независимо от значений ЛУ) или *условно* (в зависимости от условия, определяемого значениями осведомительных сигналов). В первом случае адрес следующей МК указывается в поле A1. Во втором случае адрес следующей МК выбирается из поля A1 или из поля A2 адресной части МК, в зависимости от заданного условия.

Примем, что в каждой условной МК можно проверять значения только одного ЛУ из множества X. В этом случае адресная часть МК состоит из полей X, A1, A2. В поле X указывается номер $l \in (1, \ldots, L)$ осведомительного сигнала x_l, значение которого анализируется МК условного перехода. Если поле $X=l \neq 0$, то адрес следующей МК определяется в зависимости от значения x_l. Если значение $x_l=0$, то адрес следующей МК $A_{МК}$ определяется полем A1, если $x_l=1$ – полем A2. Если поле X=0 (пустое), то адрес МК равен A1.

В итоге получаем:

$$A_{МК} := \begin{cases} \text{A1, если } X=0, \\ \text{A1, если } X \neq 0 \text{ и } x_X=0 \ (X=l) \\ \text{A2, если } X \neq 0 \text{ и } x_X=1. \end{cases} \qquad (5)$$

Длина p полей A1, A2 зависит от P – количества МК (емкости ПЗУ): $p \geq \log_2 P$. Длина поля X: $h \geq \log_2(L+1)$.

УА, построенный на основе принципа Уилкса, называется *УА с принудительной адресацией МК* (рис. 28). Для хранения микрокоманд используется ПЗУ

емкостью P ячеек, в каждой из них размещается МК длиной n=m+h+2p разрядов. Работа УА иллюстрируется временной диаграммой (рис. 29). Перед началом работы ЦУУ посылает код операции g в УА АЛУ (на вход g преобразователя начального адреса ПНА микропрограммы $МП_g$).

Запуск автомата на выполнение операции f_g производится сигналом «start» (из ЦУУ процессора). По нему триггер T устанавливается в 1, тем самым разрешается выборка слова из ПЗУ по адресу, сформированному на выходе ПНА, и открывается путь для тактовых сигналов с выхода ГТИ в УА (на выход C). ПНА обеспечивает формирование адреса первой МК микропрограммы $МП_g$. По нему она извлекается из ПЗУ и по первому сигналу C загружается в регистр микрокоманд РМК.

Адрес очередной МК формируется в соответствии с выражением (5). В последнем такте микропрограммы $МП_g$ триггер T сбрасывается сигналом y_{M+1} – конец операции f_g.

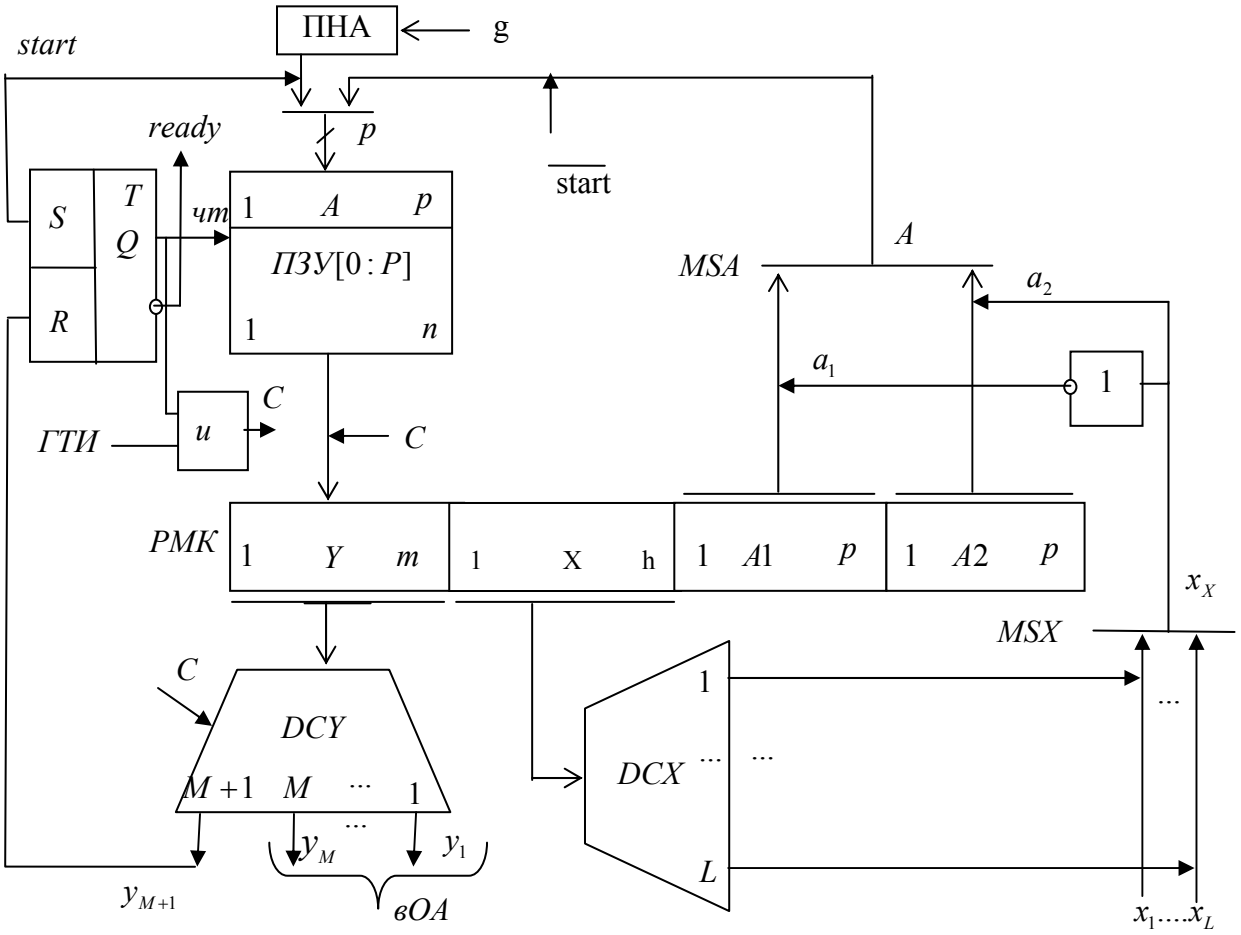


Рисунок 28 – Структура УА с принудительной адресацией

Микрокоманда, выбранная из ПЗУ и загруженная в РМК, выполняется следующим образом. Поле Y расшифровывается (дешифратором DCY) и выходной сигнал y_m с выхода DCY (импульс, сформированный по C, по $\overline{C}$ (не C) или не стробируемый совсем) поступает в ОА и возбуждает в нем соответствующую МО. Поскольку в каждом такте вырабатывается один сигнал y_m, то в ОА будет выполняться одна МО.

Переход к следующей МК осуществляется в зависимости от значения поля X и значений ЛУ x_1, …, x_L в соответствии с выражением (1). Если выполняется (в РМК загружена) МК условного перехода, то адрес МК выбирается (мультиплексором адреса MSA) из полей A1 или A2 в зависимости от значения условия x_X одним из сигналов a_1 (A:=A1) или a_2 (A:=A2). Здесь условно считается, что

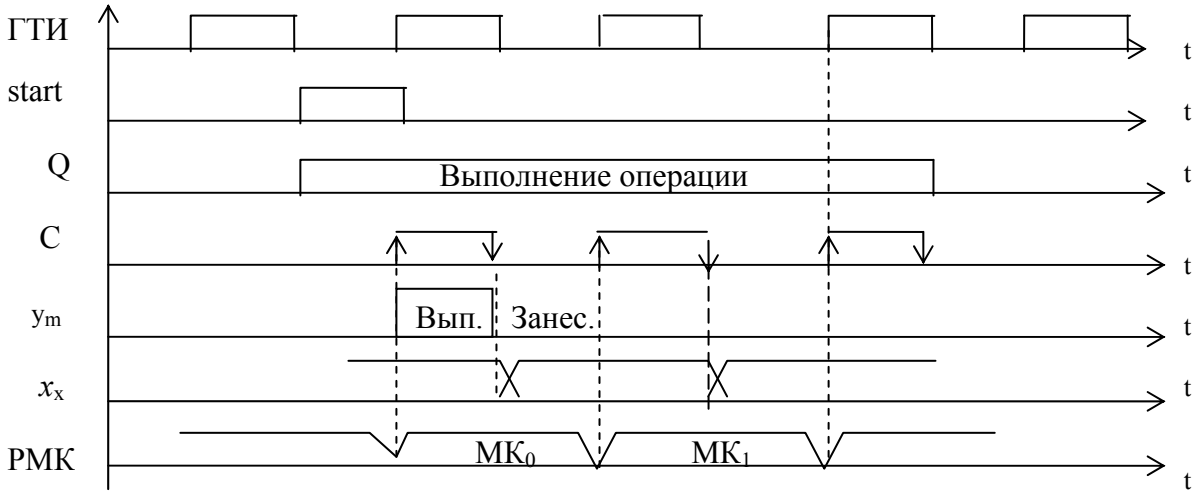


Рисунок 29 – Временная диаграмма работы УА с принудительной адресацией

условие x_0 тождественно равно 0. В этом случае адрес МК равен A1 (берется из поля A1). Выбор одного из условий x_1, …, x_L осуществляется мультиплексором MSX, которым управляет дешифратор DCX. Таким образом, к середине такта на адресном входе A ПЗУ будет сформирован адрес следующей МК, а к концу такта микрокоманда будет выбрана из ПЗУ (появится на выходе ПЗУ) и по очередному сигналу C будет загружена в РМК.

Кодирование МО. Микрооперации из списка $Y=y_1$, …, y_M возбуждаются МК, в которой указываются наименования (коды) микроопераций, выполняемых совместно в ОА. Простой (очевидный) способ кодирования МО состоит в следующем: набору сигналов y_1, y_2, ..., y_M ставится в соответствие слово $Y= y_1, y_2, ..., y_M$, в котором каждой МО, если она должна выполнятся в данном такте, ставится в соответствие единица, если не должна – то ноль, т.е. производится **унитарное кодирование МО**. Пример: M=8, такт i+1, $y_2 = y_5 = y_6 =1$, остальные биты равны нулю. Это слово используется для управления (элементами И): 1 – элемент И открыт для прохождения импульса с выхода ГТИ, 0 – закрыт (рисунок 30). Чтобы в следующем такте можно было вырабатывать другие сигналы, достаточно сменить слово Y на входе этой схемы.

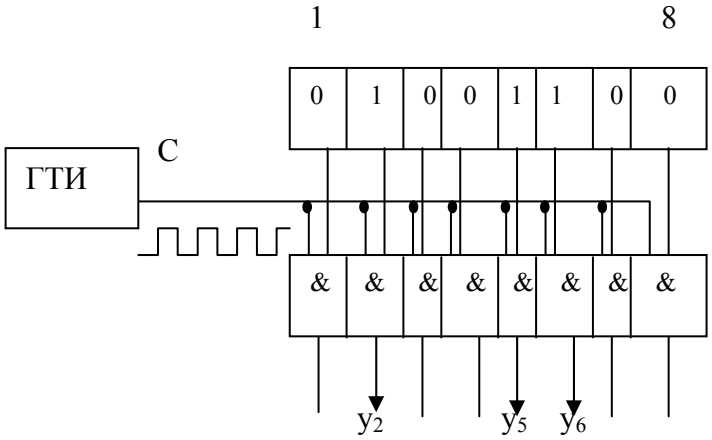


Рисунок 30 – Унитарное кодирование микроопераций

Недостаток унитарного кодирования – большая длина МК и, следовательно, большая ёмкость

ПЗУ для хранения МП. В связи с этим встаёт естественная задача сокращения длины МК.

Простой способ сокращения длины МК – **позиционное кодирование** МО: каждой МО $y_m \in Y$, которая должна выполняться в данном такте, ставится в соответствие m-разрядный позиционный код ($m \geq \log_2 M$). Пример: M=8, m=3, сигнал y_5=1, остальные сигналы равны нулю, y_5 кодируется кодом 5 Dec=101 Bin. Сигнал y_5=1 на основе этого кода вырабатывается схемой, представленной на рис. 31. Дешифратор DCY стробируется сигналами С с выхода ГТИ, один из которых появляется на выходе y_5. Недостаток позиционного кодирования – в каждом такте можно вырабатывать только один сигнал. Это нежелательно, так как вносит ограничения на совместимость МО и снижает производительность ОА.

Чтобы не вносить ограничений на совместимость МО и сократить длину МК (по сравнению с унитарным кодированием), используется **смешанное кодирование** МО. Для кодирования совместно выполняемых МО в МК выделяются поля Y_1, Y_2, …, Y_I, количество которых определяет предельное количество совместно выполняемых МО. В общем случае поле Y_i может возбуждать некоторое подмножество (несовместимых) МО $Y_i=(y_\alpha, y_\beta, \ldots, y_\omega)$ множества Y, для кодирования которых используется $m_i \geq l\log_2(M_i+1)$ разрядов, причем значение Y_i=0 является признаком пустого поля (сигнал МО в этом такте не вырабатывается). Сумма $M_O=m_1+m_2+\ldots+m_I$ определяет длину операционной части МК. Она влияет на затраты оборудования в УА и на быстродействие ОУ. В частности, уменьшение длины M_O уменьшает разрядность ячеек ПЗУ (т.е. сокращает затраты оборудования в нем). С целью экономии оборудования следует (стремиться) уменьшить длину операционной части МК. Существуют различные методы, которые позволяют это сделать.

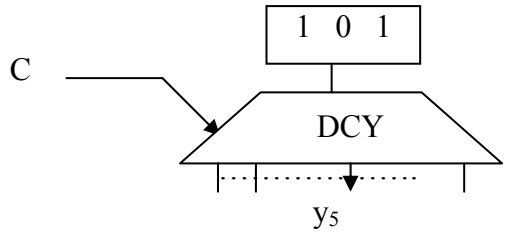


Рисунок 31 – Позиционное кодирование

Пример смешанного кодирования МО (рис. 32): три группы несовместимых МО – первая состоит из семи МО, вторая – из 9, третья – из 18. Разрядность полей: k=3 ($\log_2 7$), h=4 ($\log_2 9$), n=5 ($\log_2 18$). Всего: 3+4+5=12 разрядов вместо M=36 (7+9+18) при унитарном кодировании.

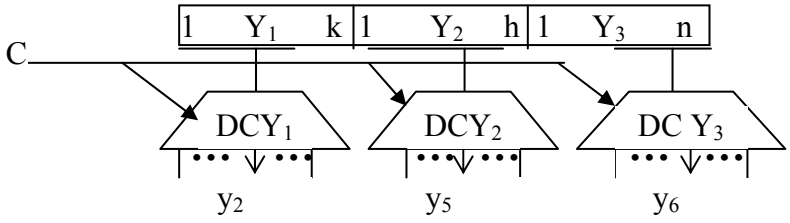


Рисунок 32 – Смешанное кодирование

Количество операционных полей. Если ОА строится на основе принципа обобщения МО, то количество операционных полей равно числу групп несовместимых МО. Так, для управления М-автоматом (рис. 16) нужна МК с четырьмя операционными полями A, B, Y, C. В них указываются двоичные номера МО,

принадлежащих наборам $\{a_i\}$, $\{b_j\}$, $\{y_m\}$, $\{c_k\}$, возбуждаемых в одном такте (совместно выполняемых).

Для управления IM-автоматом с последовательной комбинационной частью (рис. 17) необходима МК с шестью операционными полями A, B, F, G, H, C. В них указываются номера из наборов $\{a_i\},\{b_j\},\{f_k\},\{g_l\},\{h_m\},\{c_n\}$.

Для управления I-автоматом (построенным на основе принципа закрепления МО), который допускает совместное выполнение до H микроопераций, число полей в МК можно выбирать равным K=1, 2, …, H. В случае K=1 операционная часть МК имеет минимальную длину, но в каждом такте реализуется только одна МО. В результате чего функциональный оператор микропрограммы, состоящий из K совместимых МО, будет выполняться за K тактов. При K=H любой функциональный оператор (состоящий из K МО) будет выполняться за 1 такт, но операционное поле МК будет иметь суммарную длину, что увеличивает емкость ПЗУ. Таким образом, уменьшение количества операционных полей позволяет экономить оборудование в УА, но одновременно уменьшает быстродействие ОУ (увеличивает количество тактов).

Адресация микрокоманд. Для адресации МК используются два основных способа – *принудительная* и *естественная* адресация.

Принудительная адресация МК. Этот способ реализован в УА, структура которого изображена на рис. 28. Адрес следующей МК определяется либо полем A1, либо полем A2 в зависимости от кода в поле X и значения условия x_X.

Недостаток принудительной адресации – длинная адресная часть МК (и большие затраты памяти). Сократить длину адресной части МК можно, если оставить одно адресное поля A. Сделать это можно следующим образом. Если поле X=0, то значение поля A (безусловно) определяет адрес следующей МК – $A_{МК}$:=A. Если же поле X≠0, то $A_{МК}$:=A+ x_X, где x_X – значение ЛУ с номером X, т.е. в этом случае реализуется условный переход: если x_X=0, то $A_{МК}$:=A, если x_X=1, то $A_{МК}$:=A+1 – на единицу больше A. Для увеличения адреса на 1 можно использовать комбинационный счетчик КСЧ (рис. 33).

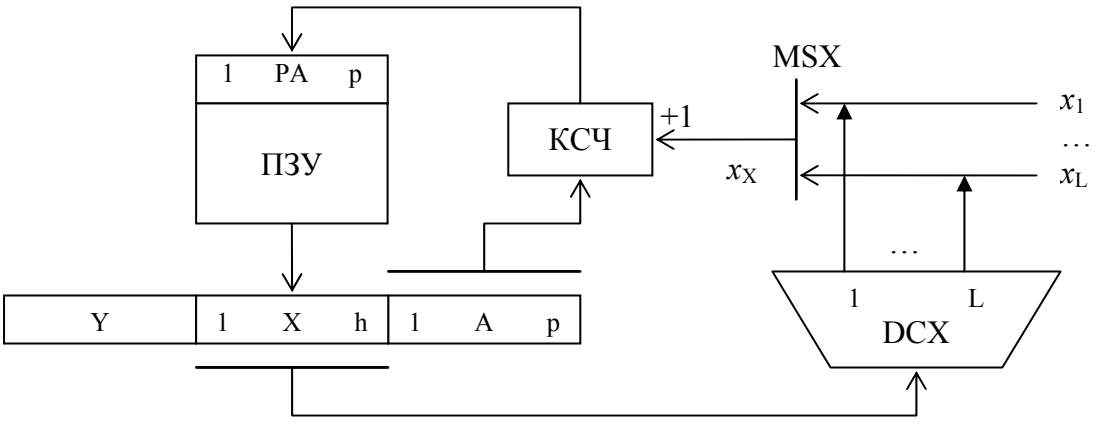


Рисунок 33 – Комбинационный счетчик

Однако введение счетчика КСЧ в схему УА снижает быстродействие автомата, поскольку длительность его такта в этом случае увеличивается на время выполнения операции счета в p-разрядном счетчике.

Естественная адресация МК. При естественной адресации адрес следующей МК формируется путем увеличения на 1 адреса предыдущей МК. В этом случае отпадает необходимость вводить адресное поле А в каждую МК. Если микрокоманды следуют в естественном порядке, то для формирования адреса МК можно использовать (накопительный) счетчик микрокоманд – СМК, состояние которого увеличивается на 1 после чтения очередной МК и дешифрации поля X:

$$\text{СМК}:= \begin{cases} \text{СМК+1, если } X\neq 0,\ x_X=0 & \text{(ЕП)} \\ \text{А, если } X=0 & \text{(БП)} \\ \text{А, если } X\neq 0,\ x_X=1 & \text{(УП)} \end{cases} \qquad (6)$$

Как видно из (6), микрокоманды естественного перехода ЕП (по СМК+1) не используют поле А для формирования адреса МК и поэтому могут содержать только операционную часть, представленную полями $Y_1,\ldots,Y_H$.

При выполнении МК переходов (безусловного – БП, условного – УП) используются поля X, A. Однако их выполнение трудно (не всегда удается) совместить с выполнением МО в ОА (так как по результатам выполненных в текущем такте МО сформируются сигналы $x_1,\ldots,x_L$, которые используются в качестве ЛУ, обычно, в следующем такте). Поэтому операционная часть в этих микрокомандах переходов не используется (поля Y – пустые, в ОА нет МО – NOP). В результате, при выполнении последовательности из двух функциональных операторов, не используется адресная часть по крайней мере в одной МК (т.е., проще говоря, не встречается последовательность из двух идущих подряд команд перехода), чтобы по ее результатам сформировались сигналы $x_1,\ldots,x_L$ (вторую МК можно совместить, например, с БП). При выполнении последовательности операторов перехода не используется (пустует) операционная часть большинства МК.

В этих условиях рационально использовать МК двух типов – *операционные* и *управляющие*.

Операционная МК содержит только поля $Y_1,\ldots,Y_H$ и неявно полагает адрес следующей МК равным СМК+1.

Управляющая МК используется только для изменения естественного порядка выполнения МК (путем выполнения МК безусловного и условного переходов), поэтому содержат только поля X, A.

Для определения (указания) типа МК вводится поле P типа МК:

P	1	Y_1	m_1	…	1	Y_H	m_H

P	1	X	h	1	A	p

Например, если P=0, то МК является операционной, если P=1, то – управляющей (или наоборот).

При использовании МК двух типов сокращается длина МК (экономится память), однако увеличивается количество тактов на реализацию операций (микропрограмм), поскольку такты отводятся либо только на выполнение МО в ОА, либо только на выполнение переходов в микропрограмме (пустые для ОА такты). Для уменьшения количества тактов можно не делить МК на два типа, а использовать один формат (как на рисунке 33). Экономии памяти в этом случае не будет.

Итак, стремление повысить производительность УА приводит к неэффективному использованию емкости ПЗУ, а стремление к экономии оборудования – к увеличению количества тактов и снижению производительности.

Быстродействие УА. Быстродействие автомата характеризуется продолжительностью такта УА, т.е. временем, затрачиваемым на формирование одного набора управляющих сигналов. Оно складывается из 3 составляющих: 1) времени формирования адреса МК; 2) времени обращения к ПЗУ; 3) времени дешифрации МК.

Основная доля времени приходится на чтение из ПЗУ, поэтому ощутимого увеличения быстродействия УА с программируемой логикой можно достичь путем уменьшения времени обращения к ПЗУ (применяя ПЗУ с более высоким на данный период времени быстродействием) или за счет сокращения количества обращений к ПЗУ.

При фиксированном быстродействии ПЗУ быстродействие УА можно повысить за счет параллельной выборки (нескольких) МК. В этом случае за одно обращение из ПЗУ выбирается K микрокоманд, и время обращения к ПЗУ, приходящееся на одну МК, в пределе сокращается в K раз. Ясно, что не все из K выбранных МК будут исполнены (в виду возможных переходов). Таким образом, из K выбранных МК в среднем реализуется N микрокоманд (где 1<N<K), т.е. эффективное быстродействие увеличивается в N раз (N<K).

Укрупненная структура УА с ПЛ (рис. 34) состоит из трех частей: схемы формирования адреса следующей МК СФАМ (которая формирует адрес, например, в соответствии с выражением (6)), ПЗУ и схемы формирования управляющих сигналов $y_1,\ldots, y_M$, состоящей из дешифраторов $DCY_1,\ldots,DCY_M$.

Временная диаграмма работу УА с ПЛ приведена на рис. 35.

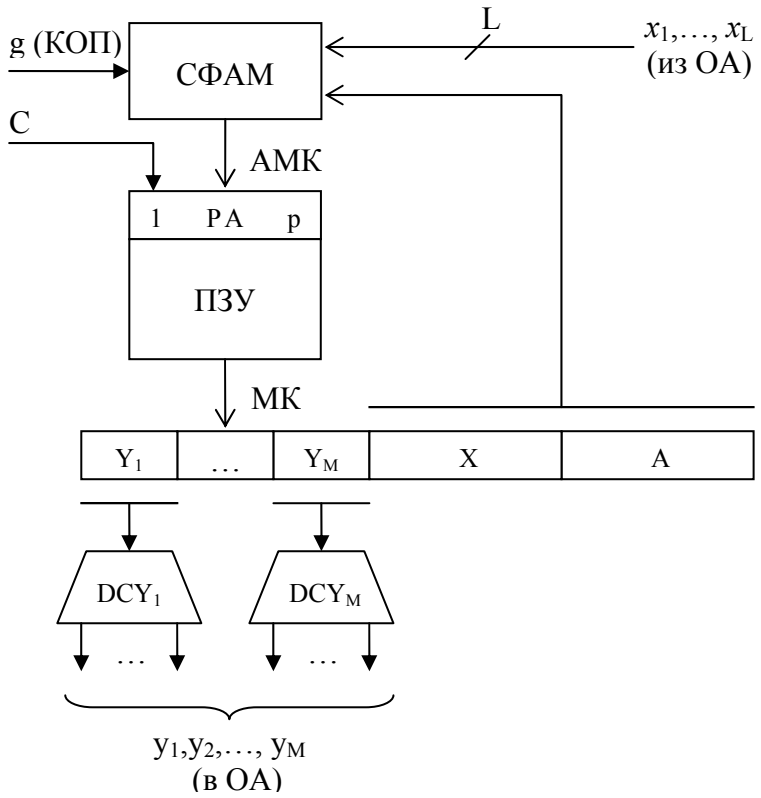


Рисунок 34 – Схема УА с ПЛ

Продолжительность такта $T_{ОУ}=T_{УА}+T_{ОА}$ работы операционного устройства определяется суммарным временем $T_{УА}$, затрачиваемым в УА на формирование адреса – $\tau_{ФА}$, на выборку МК из ПЗУ – $\tau_{ПЗУ}$ и на дешифрацию МК – $\tau_{ДШ}$: $T_{УА}=\tau_{ФА}+\tau_{ПЗУ}+\tau_{ДШ}$, и суммарным временем $T_{ОА}$, затрачиваемым в ОА на выполнение МО – $\tau_{ОА}$, на формирование логических условий $\tau_{ЛУ}$ и на запись результатов в регистры ОА – $\tau_{РЕГ}$: $T_{ОА}=\tau_{ОА}+\tau_{ЛУ}+\tau_{РЕГ}$. При последовательной работе УА и ОА продолжительность такта $T_{ОУ}$ оказывается большой.

Для уменьшения длительности такта $T_{ОУ}$ можно использовать *конвейерную организацию* работы ОУ, которая предполагает совмещение во времени работы УА и ОА. С целью совмещения работа ОУ разделяется на этапы, например: 1) выборка МК из ПЗУ; 2) ее реализация. В этом случае УА, выбрав i-ю МК, может (не дожидаясь завершения процесса ее выполнения в ОА) начать выборку следующей (i+1)-й МК.

Чтобы обеспечить возможность *опережающей выборки* МК, очевидно, в состав УА необходимо ввести специальный регистр МК (РМК), в который заносится выбранная МК. Надобность в РА (как на рисунке 34) при конвейерной обработке отпадает. К моменту времени, когда выбранная МК будет реализована, в РМК заносится (i+1)-я МК, выбранная с опережением (рисунок 36). Как видно из временной диаграммы на рис. 36, такт $T_{ОУ}$ равен такту работы конвейера $T_К$, продолжительность которого определяется:

$$T_К = \max(T_Р, T_В), \qquad (7)$$

где $T_Р$ – время реализации (выполнения) i-й МК, $T_В$ – время выборки (i+1)-й МК.

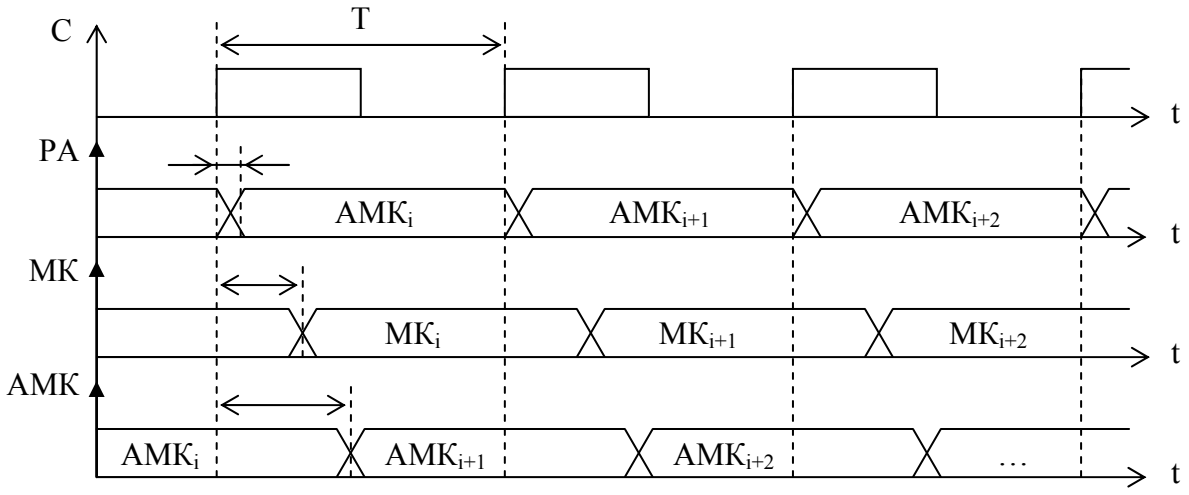


Рисунок 35 – Временная диаграмма работы УА с ПЛ

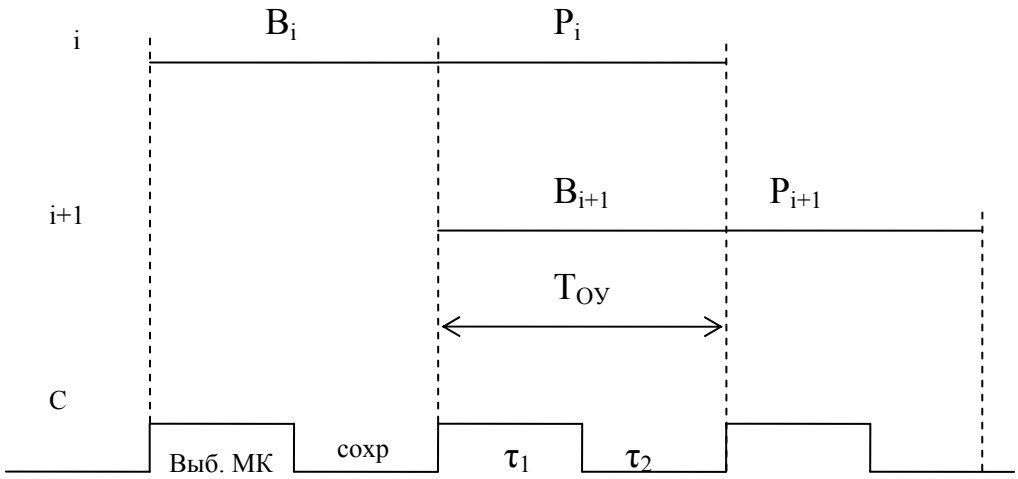


Рисунок 36 – Опережающая выборка микрокоманд

Для реализации конвейерного режима работы процессы, протекающие в УА и в ОА, следует разделить на два этапа. В УА к первому этапу относятся: формирование адреса МК и обращение по этому адресу к ПЗУ (выборка МК), а ко второму – занесение МК в РМК и ее дешифрация. В ОА к первому этапу относятся: выборка операндов из памяти ОА и выполнение МО - $\tau_{МО}$, формирование осведомительных сигналов $\tau_{ЛУ}$; ко второму – запись результатов в регистры ОА. Отсюда следует, что продолжительность такта конвейера $T_К$ определяется:

$$T_К = \max[\underbrace{(\tau_{ФА} + \tau_{ПЗУ})}_{УА}, \underbrace{(\tau_{МО} + \tau_{ЛУ})}_{ОА}] + \max[\underbrace{(\tau_{ЗАП} + \tau_{ДЕШ})}_{УА}, \underbrace{(\tau_{РЕГ})}_{ОА}] . \quad (8)$$

Первое слагаемое определяет продолжительность первого полутакта τ_1, а второе – второго полутакта τ_2: $T_К = \tau_1 + \tau_2$, задаваемого сигналами С с выхода ГТИ. В первом полутакте УА формирует адрес МК и выбирает ее из ПЗУ, а ОА в это время выбирает операнды, выполняет МО и формирует ЛУ. Во втором полутакте УА заносит выбранную МК в РМК и декодирует ее, а ОА – заносит результаты МО в регистры.

Итак, по фронту сигнала С запускаются первые этапы в УА и ОА, а по срезу – вторые этапы, а сигнал С заводится на РМК.

Следует отметить, что конвейерная организация увеличивает быстродействие ОУ почти вдвое, при условии, что продолжительности этапов в ОА и УА приблизительно одинаковые.

Время выполнения операций в ОУ, которое определяется величиной $t_{опер} = nT_{ОУ}$, где n – среднее количество тактов, при конвейерной организации зависит от количества условных переходов в микропрограмме. Точнее, от количества «промахов» при выборке МК, условия перехода для которой еще не «созрели» в ОА. Если при условном переходе выбрана не та МК, то результаты ее выполнения аннулируются (пустой такт в ОА) и потребуется дополнительный такт для выборки МК по альтернативному адресу.

Уменьшить количество «промахов» при выполнении МК условных переходов можно путем предсказания переходов. Простейший способ предсказания –

привязать выбор (одного из двух) адреса к типу МК перехода, например, если это обычный альтернативный переход, то вероятность перехода – пятьдесят на пятьдесят. В этом случае можно выбирать следующую МК по условию «нет». Если условный переход используется для организации цикла (МК стоит в конце цикла), то можно его выделить в специальный тип перехода – «конец цикла». В этом случае вероятность перехода в начало цикла выше, чем завершения цикла, поэтому опережающую выборку МК следует выполнять по адресу в начало цикла.

Следует отметить, что микропрограммы различных операций ОУ могут содержать общие по функциям участки, которые целесообразно оформлять как подпрограммы, вызываемые по (микро) команде Call, возврат из которых обеспечивается МК возврата - Ret. В этом случае в СФАМ встраивается регистр, в который по команде Call заносится адрес возврата из подпрограммы (т.е. i+1, где i – адрес МК Call), а по команде Ret адрес i+1 извлекается из регистра и выдается на выход СФАМ в качестве $A_{МК}$. Если вызов подпрограммы осуществляется из цикла, при организации которого используется команда типа «конец цикла», и если условие истинно, то в этом случае одного регистра для сохранения адреса возврата недостаточно (нужно как минимум два). Их (регистры) в этом случае рационально организовать как стековое ЗУ.

4.1.2 Схема лабораторного макета управляющего автомата арифметико-логического устройства

При разработке схемы макета выбран ориентированный на конвейерную работу вариант построения УА с естественной адресацией и единым форматом МК, состоящим из полей X, A, $Y_1 \ldots Y_H$. Структура УА представлена на рисунке 37. Для реализации конвейерной работы в УА включен регистр МК РМК. Поле X используется для указания типа МК. Всего УА реализует 13 МК, список которых представлен ниже (табл. 6).

Регистр микрокоманд РМК выполнен на триггерах типа MS, т. е. состоит из РМК′ (1-й уровень) и РМК (2-й уровень). В РМК′ микрокоманда заносится по срезу сигнала С, т. е. по $\overline{C}$ (по не С), а в РМК она переписывается по фронту С.

Блок декодирования полей $Y_1 \ldots Y_H$ (пунктирные линии на рис. 37) в лабораторном макете не реализован (так как ОА отсутствует).

Блок (схема) формирования адреса МК (рисунок 38) содержит накопительный СМК, стек, преобразователь начального адреса ПНА.

Адрес МК формируется на выходе СМК. Счетчик микрокоманд выполнен на основе трех блоков: СМК, КСЧ, СМК′. Комбинационный счетчик КСЧ реализует микрооперацию счета ВЫХ:=ВХ+1, т.е. СМК′:=СМК+1. Блоки СМК′, СМК выполнены как регистры на синхронных D-триггерах. Адрес МК на вход СМК мультиплексор адреса МА может выдавать из 4 мест:

– с входа D (по сигналу y_1),

– с выхода ПНА (по сигналу Start),

– из (вершины) стека (по сигналу y_2),

– из СМК′ (по сигналу y_3).

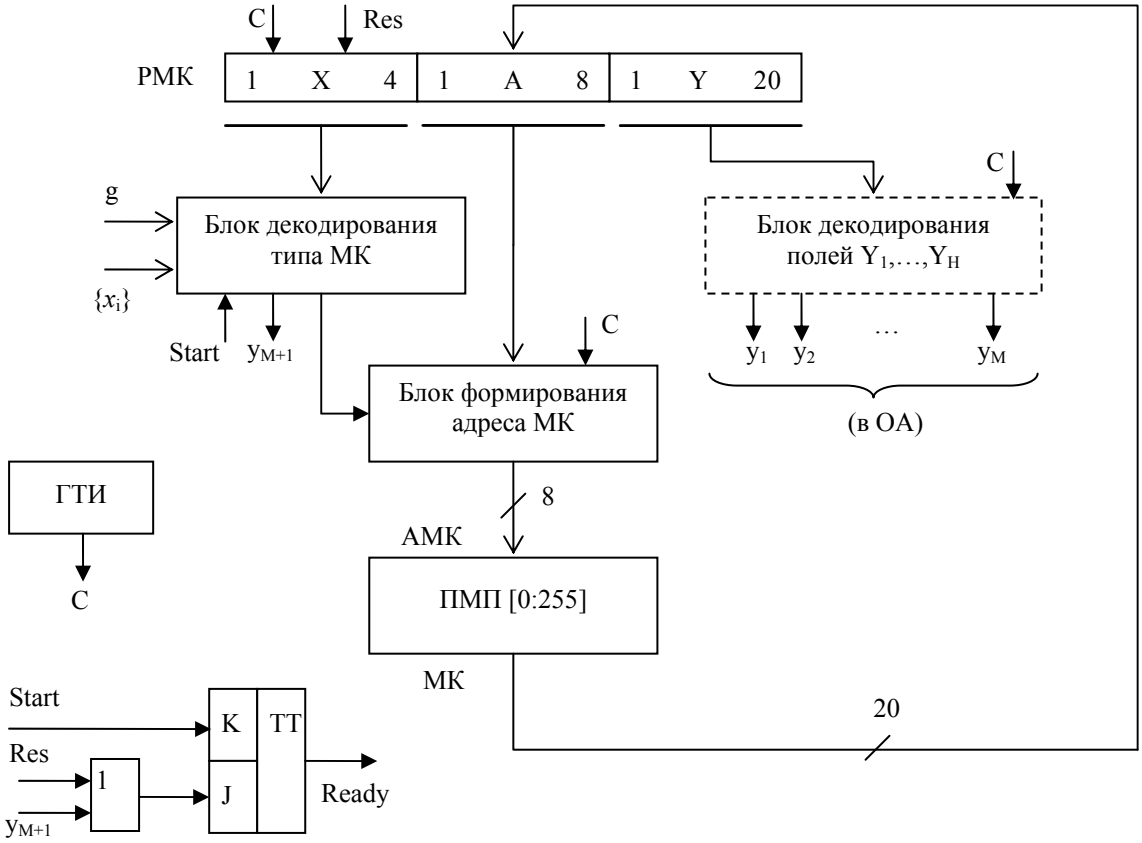


Рисунок 37 – Схема лабораторного макета УА

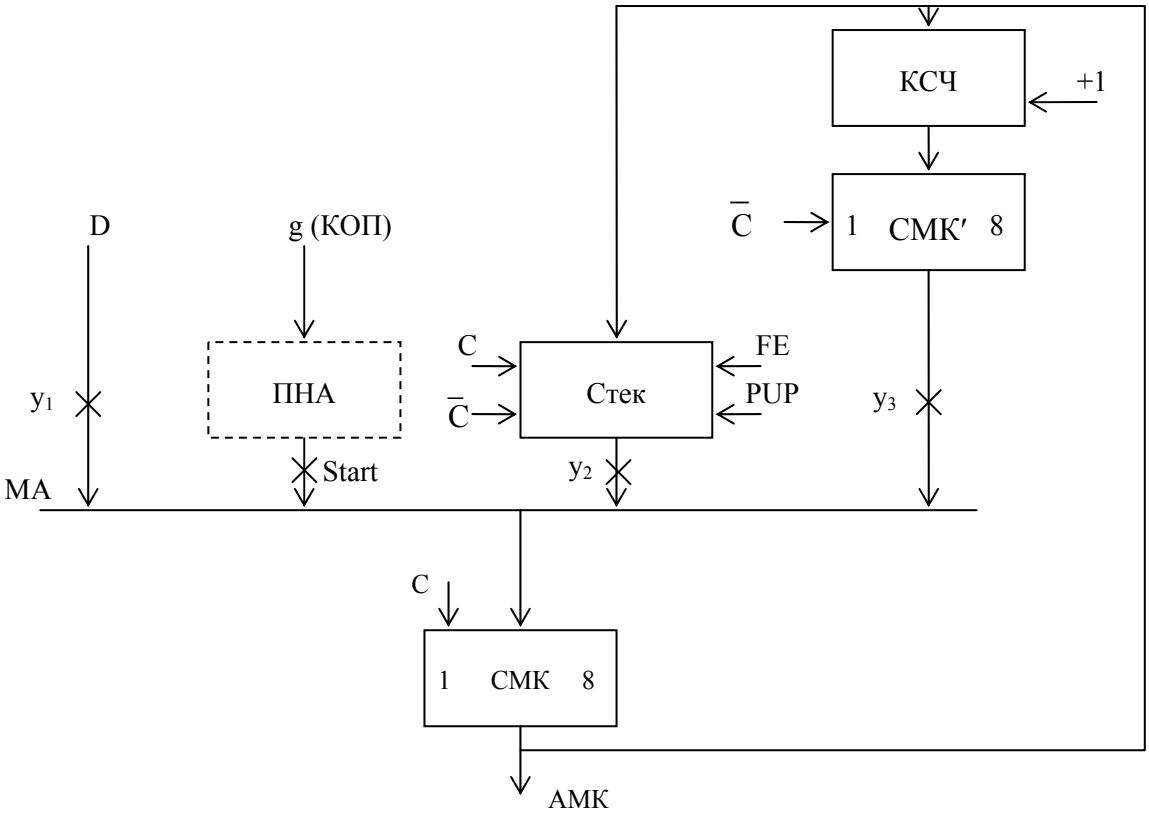


Рисунок 38 – Схема блока формирования адреса МК

Таблица 6 – Список МК

Тип МК	Описание МК
1. cnt	Переход по СМК (АМК=СМК)
2. Jmp	БП по адресу А из РМК (АМК=А)
3. Резерв	Не используется
4. Call	БП на подпрограмму (АМК=А, стек:=СМК)
5. Ret	Возврат из подпрограммы (АМК = стек)
6. Push	Запись в стек (СМК → стек) и переход по СМК
7. Endmp	Конец микропрограммы, формирование Ready
8. Резерв	Не используется
9. JZ	УП по нулю (по Z=1 СМК:=А)
10. JS	УП по знаку (по S=1 СМК:=А)
11. JC	УП по переносу (по С=1 СМК:=А)
12. JOVR	УП по переполнению (по OVR=1 СМК:=А)
13. EndZ	Конец цикла по нулю (по Z=1 – выход из цикла и выталкивание стека, по Z=0 АМК = стек)
14. EndS	Конец цикла по знаку (по S=1 – выход из цикла)
15. EndC	Конец цикла по переносу (по С=1 – выход из цикла)
16. Резерв	Не используется

На вход D адрес А подается из поля А РМК. Сигнал y_2 вырабатывается при выполнении МК условного и безусловного переходов. Стек используется для обслуживания МК Call, Ret, Push, EndZ, EndC, EndS (табл. 6). При их выполнении может вырабатываться сигнал y_2 (см. табл. 7). Сигнал y_3 вырабатывается при выполнении МК естественного перехода по СМК.

ПНА обычно реализуется на основе ПЗУ, в ячейках (с адресами $g \in (1, 2, 3, \ldots, G)$) которого размещены пусковые адреса микропрограмм МПg. Однако введение дополнительного ПЗУ в состав схемы ФАМ увеличивает продолжительность такта T_{yA}. Чтобы этого не произошло можно код g использовать как пусковой адрес. Для этого в ячейке ПМП с номером $g \in (1, 2, 3, \ldots, G)$ следует поместить МК безусловного перехода, в адресной части которой указан адрес первой микрокоманды $МКg_1$ микропрограммы МПg. Поэтому в схеме лабораторного макета ПНА отсутствует.

Порядок функционирования УА представлен временной диаграммой (рис. 39). По сигналу сброса Res УА приводится в исходное состояние (готовности выполнять операции) – сбрасывается РМК, устанавливается сигнал (флаг) готовности Ready (на выходе триггера ТТ). Сброшенный РМК содержит МК «Jmp 0».

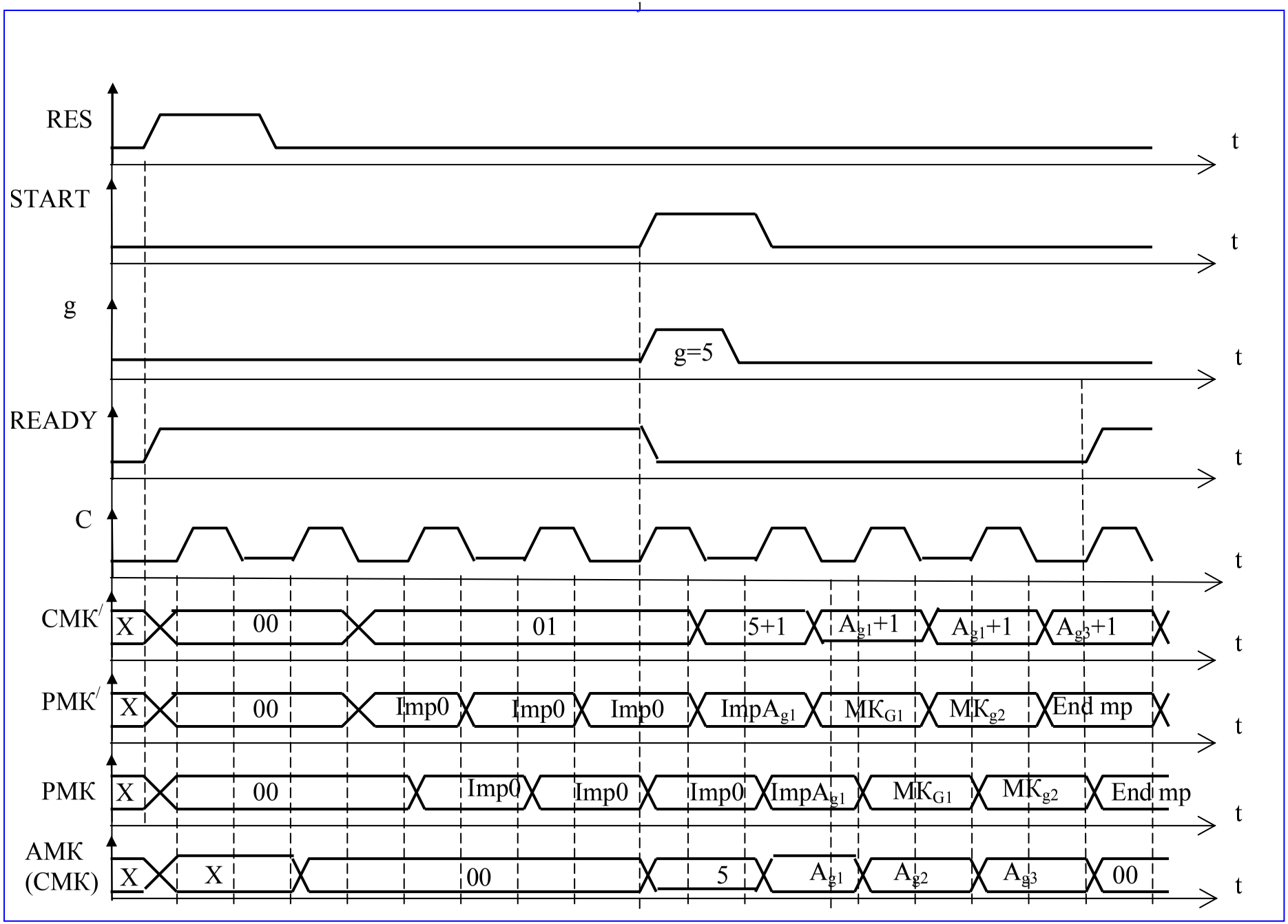


Рисунок 39 – Временная диаграмма работы УА

По первому (после RES) тактовому сигналу С в регистр СМК заносится адрес МК, равный нулю (поскольку РМК сброшен и на входе D адрес А=0). По нулевому адресу в ПМП записана МК Jmp с адресом А=0 - переход на нулевую ячейку. Она извлекается и загружается в РМК. Тем самым начинается (бесконечный) цикл передачи управления на себя. Он прерывается по сигналу Start и запускается процесс выполнения микропрограммы МПg.

По сигналу Start и коду операции g (поступающим из ЦУУ процессора) начинается процесс выполнения операции f_g (микропрограммы МПg). По сигналу Start сбрасывается триггер ТТ, т.е. сигнал готовности READY=0, и формируется начальный (пусковой) адрес МПg, по которому из памяти микропрограммы выбирается первая МК микропрограммы МПg и заносится в РМК. По сигналу Start адрес первой микрокоманды $МК_{g1}$ МПg заносится в СМК (по фронту сигнала С) и появляется на выходе АМК. По нему из ПМП выбирается первая $МК_{g1}$ и заносится в РМК′ (по $\overline{C}$), а затем - в РМК (по С). Кроме того, по сигналу $\overline{C}$ в регистр СМК′ заносится АМК (из СМК), увеличенный на 1 в КСЧ.

Во втором (следующем) такте адрес МК формируется в зависимости от типа первой МК, записанной в РМК. Если эта МК генерирует переход, то в СМК заноситься адрес либо со входа D (из поля А РМК), либо из (вершины) стека. Если эта МК является МК естественного перехода, то в СМК′ заносится СМК+1 с выхода

КСЧ. В конце МПg ставится МК Endmp (конец МП), по которой вырабатываются сигналы y_{M+1} (конец алгоритма). Он устанавливает триггер Т и тем самым формирует сигнал READY=1, обнаружив который, ЦУУ сохраняет результат операции f_g в памяти (в соответствии с адресной частью команды).

В макете реализован стек емкостью две ячейки (регистры R0, R1, выполненные на основе триггеров типа MS). Двух ячеек достаточно для обслуживания команд Call, Ret, EndZ и др. Однако недостаточно для написания вложенных подпрограмм, организованных с использованием команд конец цикла. Схема стека и временная диаграмма представлены на рисунках 40 и 41 соответственно.

Стек работает следующим образом. После сброса указателя стека УС (по сигналу Res) стек пуст и готов выполнять действия в соответствии с типом МК, находящейся в РМК. Блок дешифрации типа МК (поля Х) вырабатывает сигналы, управляющие схемой ФАМ: y_1, y_2, y_3, FE, PUP. Для управления работой стека предназначены сигналы FE, PUP. Сигнал FE=1 разрешает работу стека (FE=0 – запрещает). Если FE=1, то по сигналу PUP=1 выполняется запись в стек, а по сигналу PUP=0 - чтение стека. Запись в стек осуществляется в два этапа:

1) сначала модификация УС:=УС+1 (в начале такта – по С). Увеличение УС на 1 означает проталкивание стека;

2) затем запись в вершину стека [УС]:=IN (сигналами W_0, W_1 в середине такта – по $\overline{C}$).

Чтение (выталкивание) стека:

1) чтение из вершины стека OUT:= [УС] (постоянно);

2) модификация УС:=УС–1 (по сигналу $\overline{C}$) – выталкивание стека.

Чтение из стека (выдача на выход OUT) осуществляется с помощью мультиплексора MS, управляемого сигналами «0», «1» с выхода УС, выполненного на основе триггера со счетным входом (типа ТТ).

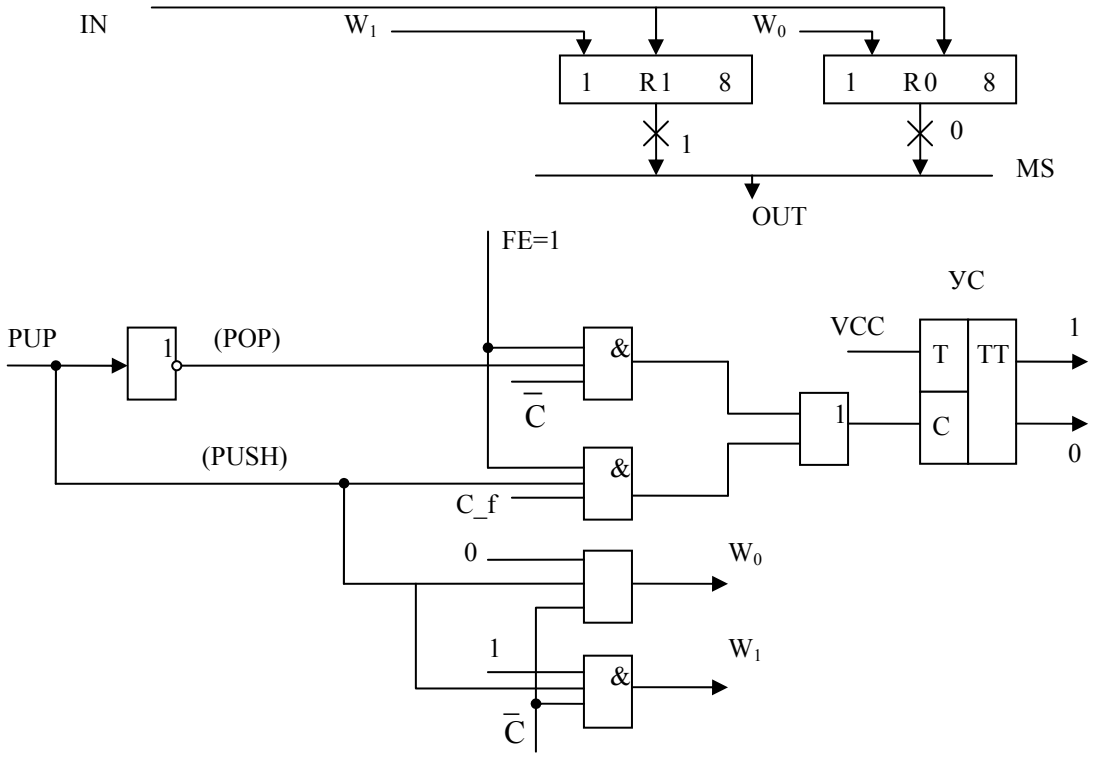


Рисунок 40 – Схема стека

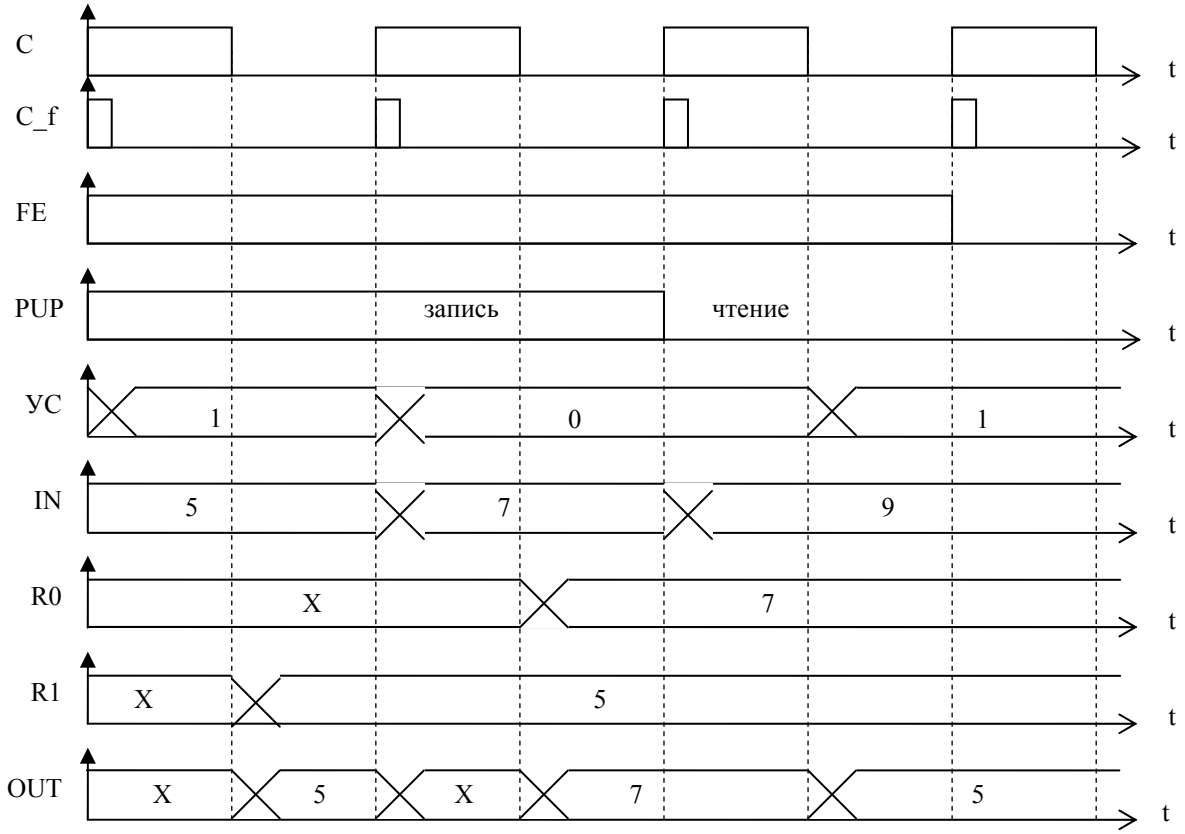


Рисунок 41 – Временная диаграмма работы стека

Блок декодирования типа МК (поля X) (рисунок 42) обеспечивает управление схемой ФАМ: коду из поля X и логическим условиям Z, S, C, OVR из ОА ставит в соответствие набор управляющих сигналов y_1, y_2, y_3, FE, PUP, y_{M+1}.

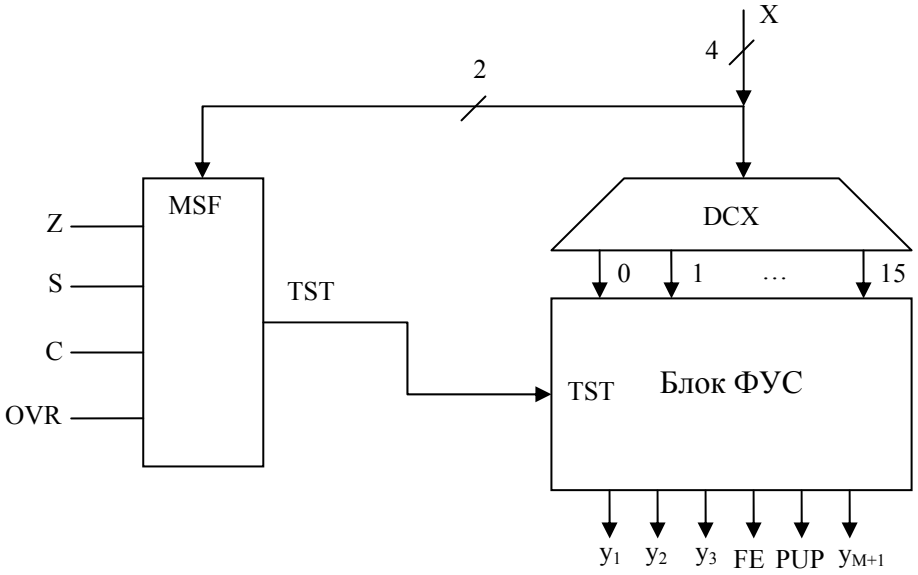


Рисунок 42 – Схема блока декодирования поля X

Таблица 7 – Формирование управляющих сигналов

Тип перехода	Код X		TST	Выходные сигналы (активные)					
		НЛУ		y_1	y_2	y_3	FE	PUP	y_{M+1}
Jmp	00	00	–	1					
Cnt	00	01	–			1			
Резерв	00	10	–						
Call	00	11	–	1			1	1	
Ret	01	00	–		1		1	0	
Push	01	01	–			1	1	1	
Endmp	01	10	–						1
Резерв	01	11	–						
JZ	10	00 (Z)	0			1			
			1 (да)	1					
JS	10	01 (S)	0			1			
			1	1					
JC	10	10 (C)	0			1			
			1	1					
JOVR	10	11 (OVR)	0			1			
			1	1					
EndZ	11	00	0		1				
			1			1	1	0	
EndS	11	01	0		1				
			1			1	1	0	
EndC	11	10	0		1				
			1			1	1	0	
Резерв	11	11	–						

Формирование управляющих сигналов производится в зависимости от типа МК (кода в поле X) и признаков (флагов) Z, S, C, OVR, вырабатываемых в ОА. Мультиплексор флагов MSF выбирает одно из условий и подает его значение на вход TST блока формирования управляющих сигналов (ФУС). Номер логического условия НЛУ представлен младшими битами поля X. Блок ФУС вырабатывает сигналы в соответствии с таблицей 7.

4.2 Задание на самостоятельную работу

1. Изучить схему и операционные возможности УА АЛУ.

2. В соответствии с указанными преподавателем вариантом задания и пусковым адресом микропрограммы разработать микропрограмму, обязательно содержащую (использующую) указанные в задании типы микрокоманд переходов. Варианты заданий представлены в табл. 8. Выполнить микропрограмму в шаговом режиме. Перед выполнением загрузить (инициализировать) память микропрограмм. Для этого после этапа компилирования проекта запустить симулятор, войти в меню инициализации, выбрать режим инициализации памяти. Результаты выполнения представить в виде временной диаграммы и трассы в виде таблицы.

3. Определить задержки сигналов от входов до выходов УА и продолжительность такта T УА.

4. Написать отчет, содержание отчета см. в подразделе 4.3.

Таблица 8 – Варианты для самостоятельных заданий

Номер варианта	Условный переход	Безусловный переход	Цикл
1	jz	jmp	endz
2	jz	jmp	ends
3	js	jmp	endc
4	jovr	jmp	endz
5	jc	call, jmp	ends
6	jz	jmp	endc
7	jz	call, jmp	endz
8	js	jmp	ends
9	jovr	jmp	endc
10	jc	jmp	endz
11	jz	jmp	ends
12	jz	call, jmp	endc
13	js	jmp	endz
14	jovr	jmp	ends
15	jc	jmp	endc
16	jz	jmp	endz
17	jz	jmp	ends
18	js	jmp	endc
19	jovr	call, jmp	endz
20	jc	jmp	ends
21	jz	jmp	endc
22	jz	call, jmp	endz
23	js	jmp	ends
24	jovr	jmp	endc
25	jc	call,jmp	endz

Проект для лабораторной работы 5 находится в папке «Лаб. работы `Организация АЛУ` - Примеры выполнения программ для лаб. работы\2011\alu_full6310s.dir» (рекомендуется использовать этот проект). В нем для компиляции следует использовать файл ..\ua.gdf. Микропрограмму следует сначала записать в память. Для этого перед компиляцией рекомендуется выбрать опцию «Assign | Device | FLEX10KA». Далее, после компиляции рекомендуется выбрать файл ..\\ua.scf. После выбора опции «Simulator», но до нажатия его клавиши «Start», следует инициализировать память путем выбора опции «Initialise | Initialise memory». Рекомендуется выбрать опцию «Memory name | pzu:37:lpm_rom:LPM_ROM_component|altrom:srom|content», и в открывшемся окне выбрать «Import file | ua(1)й.mif».

4.3 Содержание отчета

Отчет должен содержать цель работы, задание, микропрограмму, схему алгоритма программы, а также временные диаграммы с заданными параметрами сигналов и трассу выполнения микропрограммы. Следует записать длительность периода тактовых импульсов. Также отчет должен содержать значение максимальной рабочей частоты проекта.

Следует сделать выводы по результатам работы.

4.4 Контрольные вопросы

1. Назначение и организация работы УА (рис. 37).
2. Назначение и организация работы блока ФАМ (рис. 38).
3. Какие сигналы определяют адрес на выходе БФАМ?
4. Назначение стека и порядок его работы (рис. 40).
5. Назначение СМК и порядок его работы.
6. Назначение полей X и A микрокоманды.
7. Назначение и организация работы блока декодирования типа МК (поля X).
8. Назначение сигналов start, ready, res, y_{M+1}.
9. Поясните порядок выполнения работы на примере 1 варианта задания.
10. Поясните микрокоманды условного перехода.
11. Поясните микрокоманды безусловного перехода.
12. Поясните микрокоманды push, endmp.

5 ЛАБОРАТОРНАЯ РАБОТА №5. ОРГАНИЗАЦИЯ АРИФМЕТИКО-ЛОГИЧЕСКОГО УСТРОЙСТВА

Цель лабораторной работы – изучение принципов построения и операционных возможностей арифметико-логических устройств (АЛУ).

5.1 Теоретические основы лабораторной работы

5.1.1 Назначение и принцип действия арифметико-логического устройства

Назначением АЛУ является реализация арифметических и логических операций из списка операций. Для выполнения операции в АЛУ надо подать значения(е) операндов(а) и указать номер операции из списка операций.

Принципы работы и структура АЛУ подробно изложены в разделе 2.

5.1.2 Описание схемы лабораторного макета арифметико-логического устройства

Схема лабораторного макета АЛУ представлена на рисунке 43. АЛУ состоит из операционного автомата ОА, имеющего вход D и выход Y результата операции, управляющего автомата УА, мультиплексоров MUX1, MUX2, MUX3, реализующих логику приема данных с двух входов АЛУ (ВХ1, ВХ2) в регистры ACC, D и выдачу результата из них на выходы ВЫХ1, ВЫХ2, а также регистр признаков результата РПР.

Функционирование АЛУ осуществляется следующим образом. Процессор (ЦУУ ЦП) выбирает операнд(ы) (в соответствии с адресной частью команды) и подает их на входы АЛУ (загружает в регистры ACC, D). После загрузки операнда(ов) процессор посылает код операции на вход КОП управляющего автомата, а также сигнал пуска – START, по которому запускается процесс выполнения микропрограммы, заданной кодом операции. АЛУ выполняет операцию, формирует признаки (флаги) результата и загружает их в регистр РПР, а результат операции помещает в регистр ACC (и D – если нужно). После этого формируется сигнал (признак) готовности результата READY (для ЦУУ).

Управление подачей операндов на вход D ОА осуществляется мультиплексором MUX3, на управляющий вход OS которого подаются управляющие сигналы из УА в соответствии с табл. 9.

Таблица 9 – Управляющие сигналы, подаваемые на мультиплексор MUX3

OS	ВХОД D
0X	Поле R
10	ACC
11	D

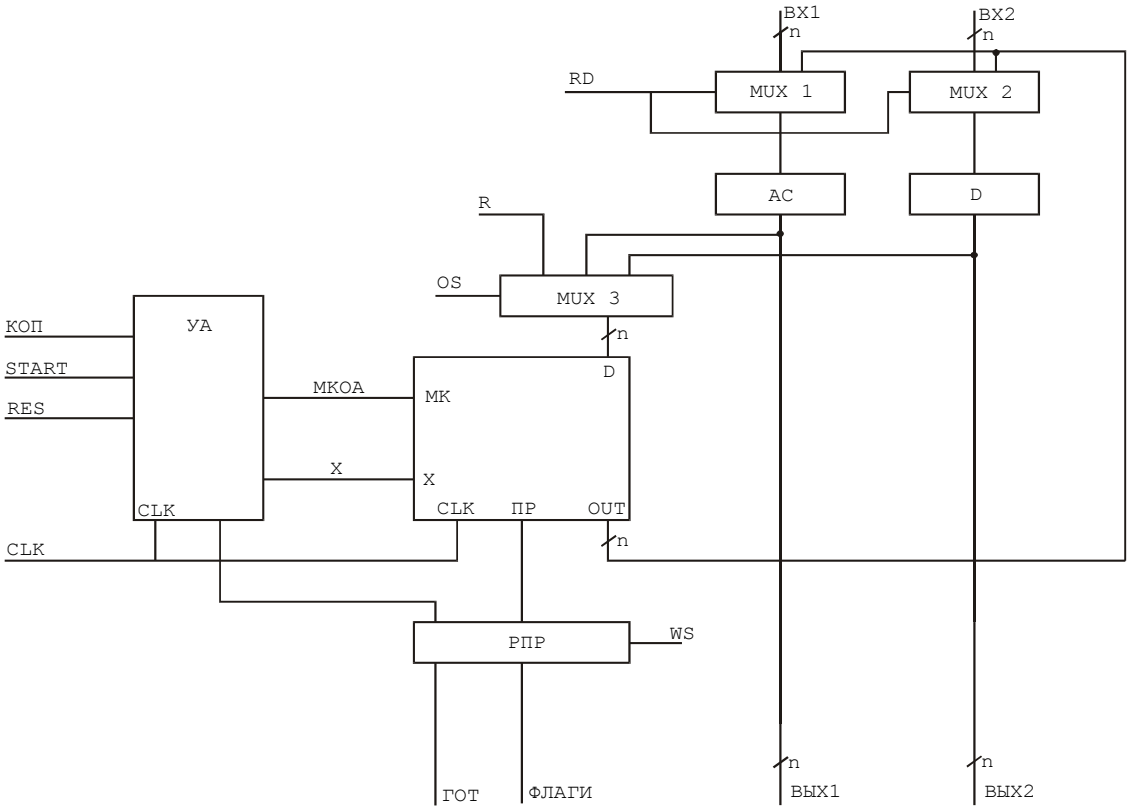


Рисунок 43 – Схема лабораторного проекта АЛУ

В табл. 10 поле R означает поступление на вход D ОА значения операнда из адресной части микрокоманды. Символ X означает безразличное состояние.

Управление записью (приемом) информации в регистры ACC, D обеспечивают мультиплексоры MUX1, MUX2 в соответствии с управляющим полем RD (табл. 10).

Таблица 10 – Сигналы, управляющие записью информации

RD	в ACC	в D
00	ACC:=BX 1	D:=BX 2
01	ACC:=OUT	Нет записи
10	Нет записи	D:=OUT
11	ACC:=OUT	D:=OUT

Следует отметить, что загрузка операндов в регистры ACC, D (с входов BX1, BX2) осуществляется по управляющим сигналам ACC_WRITE, D_WRITE, поступающим из ЦУУ процессора, а запись результата операции с выхода OUT ОА – по сигналам, поступающим из УА АЛУ.

Запись признаков результата в РПР обеспечивает сигнал WS (бит микрокоманды):

WS=0 – нет записи; WS=1 – запись.

Управление работой АЛУ осуществляется микрокомандой формата:

ГОТ	1	УА	m	WS	OS	RD
37	36 R 29	28 P 25		24	23 22	21 20

1	OA					k
19 I(8:6) 17	16 I(5:3) 14	13 I(3:0) 11	10 MS 9	8 C_{in}	7 A 4	3 B 0

Поле УА микрокоманды служит для управления функционированием УА при выполнении микрокоманд переходов различных типов. Длина m и формат поля УА определяются по результатам проектирования (параметрической настройки) УА. Ниже приведен пример формата поля УА для варианта с памятью микропрограмм емкостью 256 ячеек (длина поля УА m = 12).

Поле ОА микрокоманды служит для управления работой ОА. Длина k и формат поля ОА определяются по результатам проектирования (параметрической настройки) ОА. Ниже приведен пример формата поля ОА для варианта с количеством РОНов, равным 16 (длина поля ОА k = 20).

Работа АЛУ осуществляется тактами в соответствии с временной диаграммой, представленной на рис. 44.

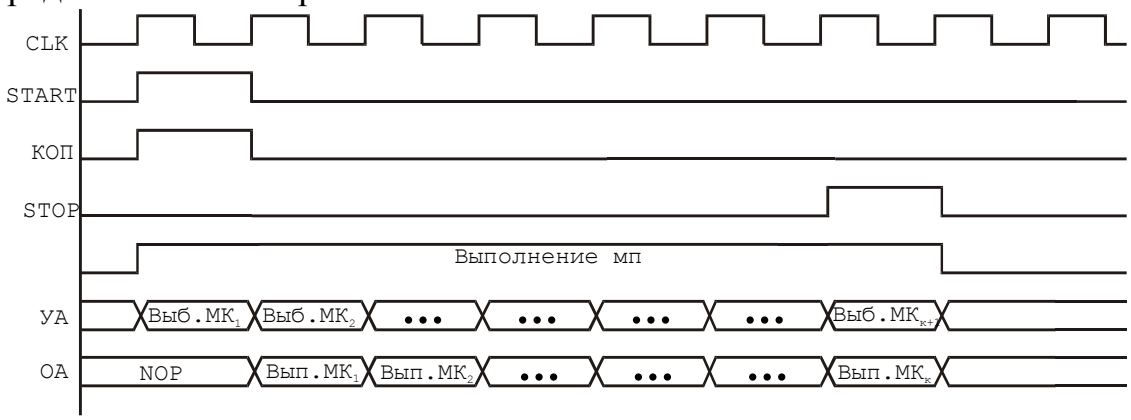


Рисунок 44 – Временная диаграмма работы АЛУ

Операционный и управляющий автоматы в составе АЛУ образуют конвейер: в то время (в том такте), когда ОА выполняет i-ю микрокоманду, УА выбирает из памяти микропрограмм (i+1)-ую МК. Очевидно, что при такой организации в первом такте работы микропрограммы ОА будет простаивать. Поэтому общее количество тактовых импульсов N=M+1, где M – количество тактовых импульсов на выполнение собственно микропрограммы.

5.2 Задание на самостоятельную работу

1. Изучить схему и операционные возможности АЛУ.

2. Разработать микропрограмму операции в соответствии с вариантом задания, указанным преподавателем. Выполнить микропрограмму в шаговом режиме. Результаты выполнения представить в виде временной диаграммы и трассы в виде таблицы.

3. Определить задержки сигналов от входов до выходов и продолжительность такта T работы АЛУ.

4. Написать отчет по результатам лабораторной работы.

Для работы нужен тот же проект АЛУ, составленный на MAX+PLUS II, который использовался в лабораторной работе №3.

5.3 Содержание отчета

Отчет должен содержать цель работы, постановку задачи, блок-схему проектируемой схемы, блок-схему проекта, созданного в ходе выполнения лабораторной работы, а также временные диаграммы с заданными параметрами сигналов. Следует записать длительность периода тактовых импульсов. Также отчет должен содержать значение максимальной рабочей частоты проекта.

Следует сделать выводы по результатам работы.

5.4 Контрольные вопросы

1. Какие сигналы определяют данные на входе D ОА АЛУ?
2. Назначение поля ОА в МК и порядок его использования.
3. Назначение поля УА в МК и порядок его использования.
4. Назначение полей OS, RD и WS в МК.
5. Какие сигналы определяют данные на выходах АЛУ?
6. Назначение и организация работы регистра РПР.
7. Назначение входов START, КОП, RES схемы АЛУ.
8. Назначение схем MUX1, MUX2, MUX3.
9. Назовите особенности записи информации в регистры ACC и D.

ЗАКЛЮЧЕНИЕ

В предлагаемых методических указаниях к лабораторным работам рассмотрены вопросы:

1) принципы действия и структурная организация АЛУ и его составных блоков – операционного и управляющего автоматов;
2) основы работы с современными средствами проектирования цифровых вычислительных систем;
3) процесс функционирования АЛУ во времени (в режиме компьютерного симулятора).

Рассмотрено проектирование двухтактного конвейерного АЛУ на основе принципа микропрограммного управления, применяющегося сейчас практически во всех цифровых вычислительных устройствах. Для изучения использован макет АЛУ, характерной особенностью которого является двухуровневое управление процессом вычислений, также широко применяемое в современных ЭВМ и контроллерах.

В лабораторных работах использованы компьютерные модели, так называемые «проекты», разработанные в среде Max+Plus II студентами и преподавателями СГАУ. Данный термин введен фирмой «Altera», разработавшей указанную САПР. В графическом диалоговом интерфейсе этой САПР проект представлен как структурно-функциональная блок-схема, содержащая условные обозначения, близкие к стандартным. Аналогичный подход применен и при компьютерном моделировании работы проекта. На экране дисплея при этом изображаются временные диаграммы, которые может изменять пользователь. Пользовательский интерфейс дает возможность заглянуть в любую точку проекта, то есть, как бы добраться щупом виртуального осциллографа туда, куда обычно, при интегральном исполнении устройств, это сделать невозможно. Это способствует глубокому изучению предмета.

В данных методических указаниях студенты вначале знакомятся с основами проектирования в MAX+PLUS II. Полученные знания они затем используют при изучении АЛУ. Студенты изучают АЛУ, используя как проекты его отдельных компонентов, таких как ОА и УА, так и проект АЛУ в целом.

Данный цикл лабораторных работ образует практическую часть курсов «Организация ЭВМ и вычислительных систем» и «ЭВМ и периферийные устройства» по специальности «АСОИУ». Для этого он используется совместно с двумя другими лабораторными работами, методические указания к которым издаются отдельно. Это лабораторная работа, посвященная функциональной организации ЭВМ, на примере системы прерываний компьютеров IBM PC, а также лабораторная работа, посвященная системе ввода/вывода компьютеров этого семейства.

Данные методические указания к лабораторным работам предоставляют возможность для обучения студентов принципам структурной организации АЛУ, а также дают основы и практические примеры разработки вычислительных систем с помощью современной САПР.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Орлов, С. А. Организация ЭВМ и систем: учебник для вузов [Текст] / С. А. Орлов, Б. Я. Цилькер. – 2-е изд. – СПб.: Питер, 2011. – 688 с. – ISBN 978-5-498097-862-5.

2. Стешенко, В. Б. ПЛИС фирмы «Altera»: элементая база, система проектирования и языки описания аппаратуры [Электронный ресурс] / В. Б. Стешенко // 3-е изд., стереотипное.- М.: Издательский дом «Додэка-XXI», 2007. - 576 с. - ISBN 978-594120-112-9. – 1 электрон. опт. диск (CD-ROM).

3. Поречный, В. Использование САПР «MAX+PLUS II" для разработки цифровых устройств на ПЛИС фирмы «Altera» [Электронный ресурс] / В. Поречный. - URL: http://www.epos.ua/view.php/pubs_3?subaction=showfull&id=983311200&archive=&start_from=&ucat=3& - (Дата обращения 30.05.2013).

4. Павлов, В. П. Организация ЭВМ и систем: учеб. пособие для студентов заоч. формы обучения / В. П. Павлов : Самар. гос. аэрокосм. ун-т им. С. П. Королева. – Самара: Самар. аэрокосм. ун-т, 2000. – 181, (1) с. – ISBN 5-7883-0122-X.

5. Баранов, С. И. Синтез микропрограммных автоматов [Текст] / С. И. Баранов. – 2-е изд., перераб. и доп. – Л.: Энергия, 1979. – 231 с.

ПРИЛОЖЕНИЕ А
Примеры отчетов по лабораторным работам

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Факультет информатики

Кафедра информационных систем и технологий

Отчет по лабораторной работе №1

по курсу

«Организации арифметико-логических устройств ЭВМ»

«Основы пользовательской работы с системой MAX+PLUS II»

Выполнил: студент группы ____

_____

Проверил:

_____

(звание, должность, Ф. И. О.)

САМАРА 2013

Цель лабораторной работы – приобретение навыков работы в среде MAX+PLUS II.

Постановка задачи

Создать проект 4-х разрядного вычитателя согласно рисунку 1. Старший четвертый разряд – знаковый.

Для демонстрации работы в симуляторе из вертикального меню слева нажать на третью кнопку по порядку снизу. В качестве начальных значений для уменьшаемого выбрать четыре или пять, инкремент выбрать равным единице. Начальное значения для вычитаемого выбрать равным нулю, инкремент выбрать равный двум.

Период смены данных на входах следует выбрать вдвое большим, чем период тактовых импульсов.

Определить задержку времени при срабатывании вычитателя. Определить максимальную частоту, на которой проектируемое устройство работоспособно. Сравнить с тактовой частотой современных компьютеров и микроконтроллеров.

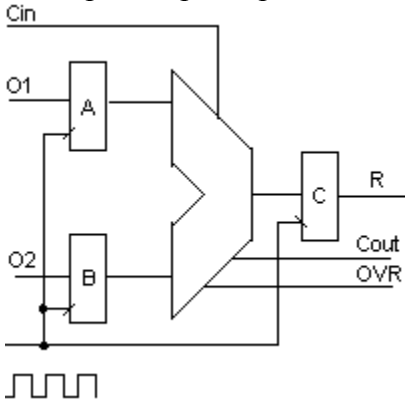


Рисунок 1 – Блок-схема вычитателя

Проекты для выполнения лаб. работ с программой Max+PLUS II находятся в папке «Лаб работа вводная по Max+plus II - Пример выполнения программы для лаб работы».

Готовый проект, относящийся к лабораторной работе 1 находится в папке \firstz.dir. Однако рекомендуется сделать его вновь самостоятельно - он довольно простой. Перед компиляцией рекомендуется выбрать опцию

Assign | Device | MAX7000.

На рисунке 2 приведена блок-схема проекта.

На рисунке 3 приведена временная диаграмма работы проекта при длительности периода тактовых импульсов T=4 нс. При этом частота ТИ: f=250 МГц.

На рисунке 4 приведена временная диаграмма работы проекта при длительности периода тактовых импульсов T=40 нс. При этом частота ТИ: f= 25 МГц.

Из рисунков 3 и 4 видно, что при большой частоте генератора тактовых импульсов устройство работает неправильно. Задержка устройства составила $T+T_D$, где T_D=10 нс. Для нормальной работы должно быть $(1/2)\ T > T_D$. Таким образом, максимальная частота для этого устройства равна $1/ (2\ T_D)$=50 МГц. Для сравнения: тактовая частота центрального процессора современного компьютера: 2 ГГц, что в 40 раз больше.

Выводы:

– изучены основные приемы проектирования цифровых устройств в САПР MAX+_PLUS II;

– проведена оценка технических возможностей цифровых устройств, изготовленных в САПР MAX+PLUS II – они оказались много ниже, чем у современных компьютеров и более характерны для микроконтроллеров.

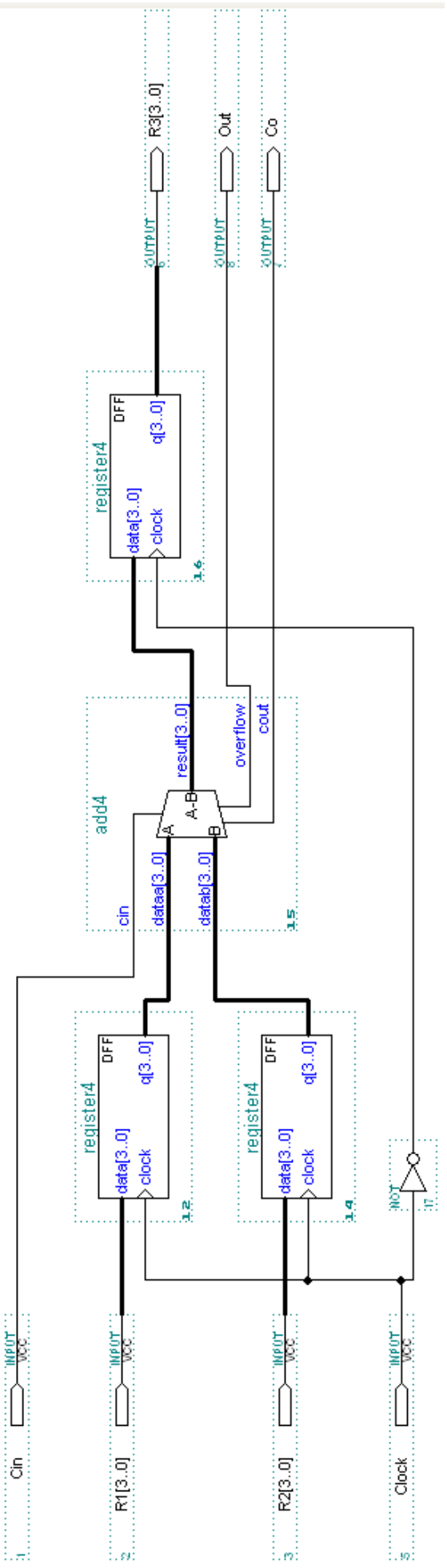


Рисунок 2 – Структурная-схема проекта

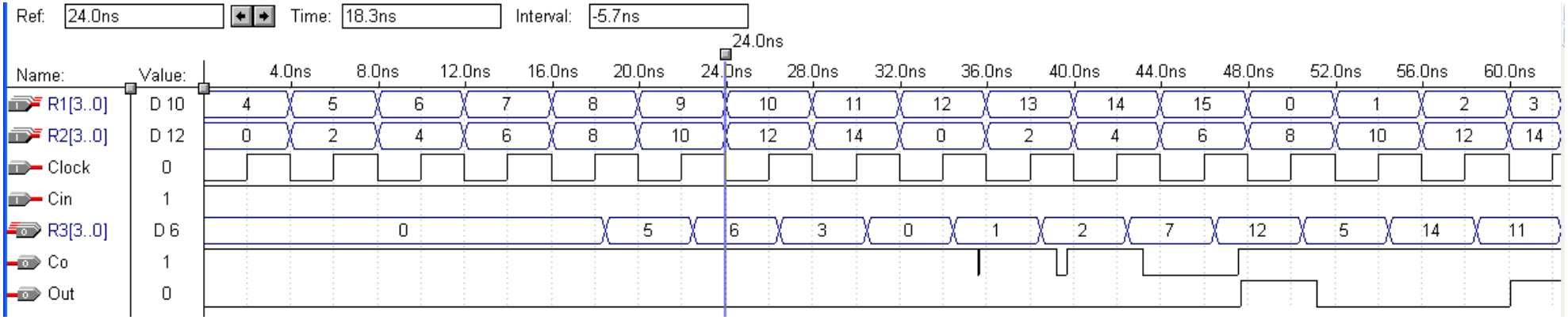


Рисунок 3- Временные диаграммы при неправильной работе устройства

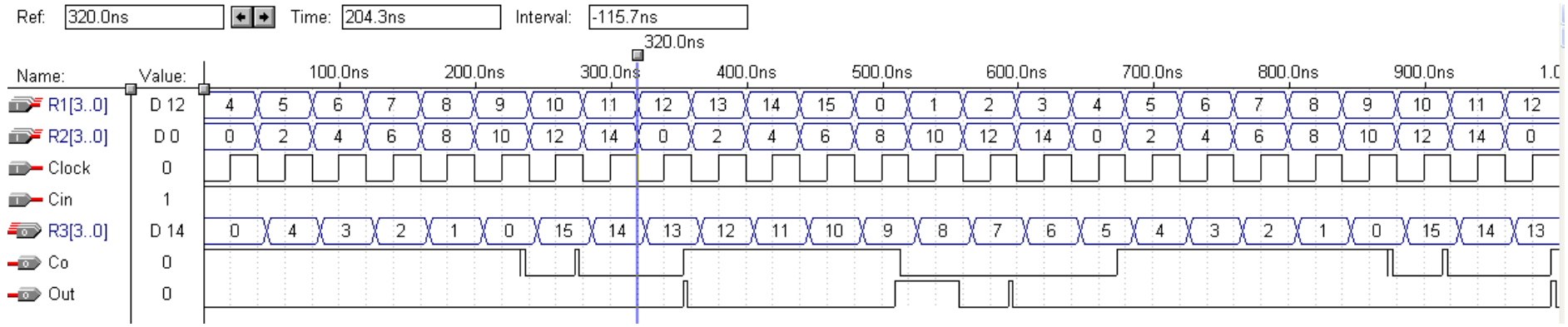


Рисунок 4- Временные диаграммы при правильной работе устройства

Пример отчета по лабораторной работе № 2
«Принципы построения АЛУ»

Цель лабораторной работы – изучение принципов построения АЛУ, исследование АЛУ различных архитектур, получение навыков работы с ними.

Задание на самостоятельную работу :

1. Изучить проект АЛУ в MAX+PLUS II.
2. Провести верификацию этого проекта.
3. Определить основные технические характеристики спроектированного устройства.

Проект по лабораторной работе 2: «Лаб работы `Организация АЛУ` - Примеры выполнения программ для лаб работы\2011\alu_full0.dir». Для компиляции использовн файл ..\alu_full.gdf. Выбранный тип микросхем памяти в проекте: FLEX10K. Временные диаграммы - в файле..\\alu_full.scf. Для инициализации памяти выбрано семейство микросхем:| ua2:|pzu:37:lpm_rom:LPM_ROM_component|altrom:srom|content", для прошивки ПЗУ использван файл «Import file | alu_full(6311).mif» из папки проекта.

В таблице 1 приведена прошивка ПЗУ. Нужная нам микропрограмма находится по адресам 01H – 02H. Эта микропрограмма записана в памяти ПЗУ еще раз – по адресам 05H – 06H.

Таблица 1 – Дамп памяти ПЗУ, адреса 01H – 09H

Адрес, HEX	Данные, HEX
01	000486F800
02	2003122100
03	0004020000
04	0004020000
05	000486F800
06	2003122100
07	0004070000
08	0004020000

Порядок выполнения микропрограммы показан в таблице 2, трасса микропрограммы – в таблице 3.

Таблица 2 – Порядок выполнения микропрограммы

ГОТ	УА	WS	OS	RD	ОА	Код МК, HEX
0	cnt	0	10	00	РОН0:=D0, C_{in}=0	000486F00
1	Jd0	1	00	01	РОН0+1 $\rightarrow$ OUT, C_{in}=1	2003122100

Таблица 3 – Трасса микропрограммы

$A_{мк}$	МО в ОА	Входы, HEX				Выходы, HEX						
		Start	Start Adr	ACC WR	Вх1	Out	Acc	Вых1	Z	OVR	C	S
a+0	РОН:= D$\vee$0	1	0	1	FF	FF	FF	00	0	0	0	1
a+1	РОН0+1 $\rightarrow$ OUT	0	1	0	00	00	00	00	1	0	1	0

В таблице 4 приведена расшифровка кода первой и двух команд микропрограммы

Таблица 4 – Коды микрокоманды, двоичные

	Гот	R	P	WS	OS	RD	I(8:6)	I(5:3)	I(3:0)	MS	C_{in}	A	B
00	0	0000000	011	0	00	00	001	000	000	00	0	0000	0000

Обозначения в таблице 4:
ГОТ – сигнал готовности результата;
R – адресная часть;
P – поле, определяющее тип перехода;
WS – бит микрокоманды;
OS – управляющий вход;
RD – управляющее поле;
I(8:6) – приемник результата;
I(5:3) – операция;
I(2:0) – источник операндов;
MS – тип сдвига;
C_{in} – признак переноса из младшегоразряда;
A, B – адреса операндов.

В таблице 5 показано соответствие входов и выходов АЛУ в проекте и в методических указаниях.

Таблица 5 – Соответствие входов и выходов АЛУ

Методические указания	Проект
Start	Pusk
ACC WR	ACC_WRITE
Start Adr	START_ADDR[pmp_a_siz-1..0]
BX1	ACCin[7..0]
OUT	Y[7..0]
ВЫХ1	ACCout[7…0]
Z	Z
OVR	OVR
C	Cout
S	Fhigh

На рисунке 1 приведена временная диаграмма работы проекта АЛУ, полученная симулятором MAX+PLUS II.

Сравнивая показания временной диаграммы с трассой микропрограммы, видим, что сигналы совпадают с данными трассировки, указанной в методических указаниях. Следовательно, устройство работает нормально.

Максимальная рабочая частота была определена из задержки выполнения микрокоманды по временной диаграмме. Результат на выходе появлялся на третьем такте, считая от первого переднего фронта ТИ после фронта сигнала Pusk. Очевидно, до конца третьего такта результат должен успеть появиться на выходе АЛУ. Значит, минимальная продолжительность такта, при котором это еще происходит, определяется как

$T_{min}=(1/3)\times T_{зад.}$:

$T_{min}=(1/3)\times 350=117$ нс.

Отсюда, максимальная частота генератора тактовых импульсов

$F_{max}=1/T_{min}$:

$F_{max}=1/(117\times 10^{-9})=5{,}882\times 10^{6}$ Гц.

Следует округлить полученную расчетную величину в меньшую сторону, учитывая погрешность вычислений и для того, чтобы отступить от предельного значения, для надежности.

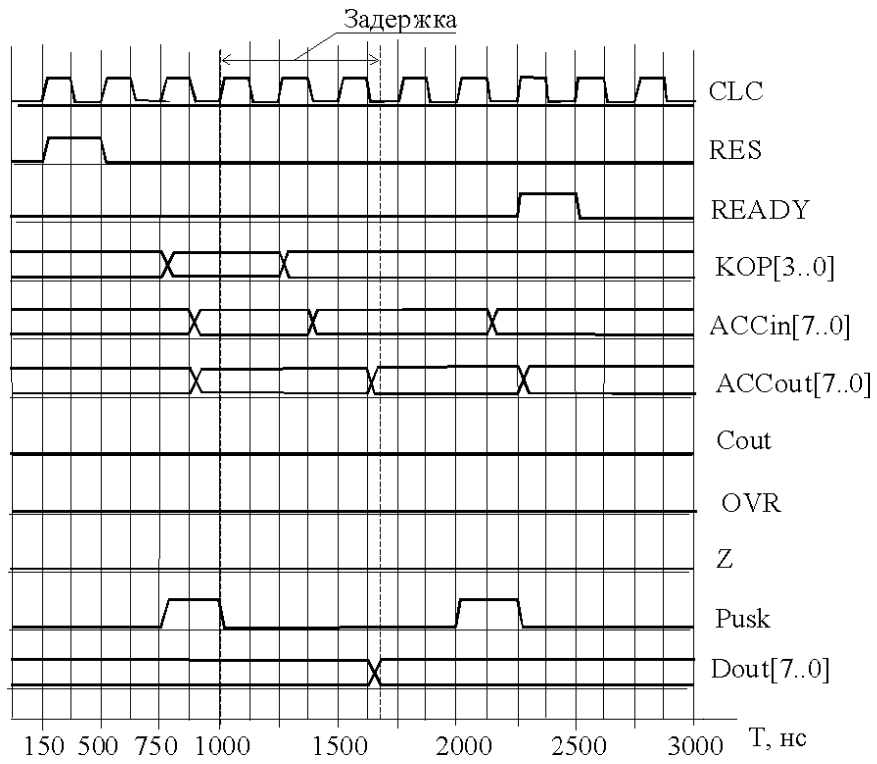
Получаем $F_{max}=5{,}8$ МГц.

Выводы:

– изучена схема и операционные возможности АЛУ;

– определена максимальная частота работы АЛУ, она оказалась много меньше, чем тактовая частота ценрального процессора современных компьютеров, и характерна больше для микроконтроллеров;

– работа АЛУ требует два такта генератора, но за счет конвейерной огранизации после первой команды следующие выполнятся за один такт (если нет ветвления).



задержка: 350 нс; такт: 250 нс

Рисунок 1 – Временная диаграмма работы проекта

Пример отчета по лабораторной работе № 3
«Организация ОА АЛУ»

Цель лабораторной работы – изучение принципов построения и архитектуры АЛУ.
Задание на самостоятельную работу :
выполнить вариант №1 задания: $Y=N_1-2N_2$ (для n=4).
Коды микропрограммы приведены в таблице 1.

Таблица 1 – Микропрограмма

цифры в столбцах, начиная со второго – в бинарном коде

Описание МО	Источники операндов	Операция	Приемники результата	Адреса		Вход перен.	Тип сдвига
	y_0, y_D	ALO	y_w, y_{out}	A1	A2	C_{in}	SH
1. POH0:=D∨0	01	011	10	-	0000	-	000
2. POH1:=2(D∨0)	01	011	10	-	0001	-	011
3.POH0:=POH0-POH1	10	010	11	0001	0000	1	000

Первая микрооперация обеспечивает загрузку (ввод) в регистр POH0 значения первого операнда N_1, подаваемого извне на вход D.

Вторая микрооперация обеспечивает загрузку (ввод) в POH1 значения второго операнда N_2 со сдвигом влево на один разряд.

Третья микрооперация обеспечивает вычитание $N_1- 2N_2$ и запись результата в POH0.

Кодирование микроопераций выполнено в соответствии с таблицей 1 (источники операндов), таблицей 2 (реализуемая УКС операция), таблицей 3 (тип сдвига) и таблицей 4 (приемники результата).

Результаты выполнения (трасса) микропрограммы для N_1= -3 (в дополнительном коде это 1101) и N_2= +3 = 0011 представлены ниже в таблице и на временной диаграмме. Значения операндов выбраны таким образом, чтобы результат переполнил разрядную сетку.

Таблица 2 - Трасса микропрограммы

цифры в столбцах, начиная со второго – в бинарном коде

Микрооперация	Входы		Результаты (выходы)								
	D	C_{in}	F	B	POH0	POH1	SL_0	CF	ZF	OF	SF
1. POH0:=D∨0	1101	0	1101	1101	1101	-	-	-	0	-	1
2. POH1:=2(D∨0)	0011	0	0011	0110	1101	0110	0	-	0	-	0
3.POH0:=POH0-POH1	-	1	0111	0111	0111	0110	-	1	0	1	0

Временные диаграммы работа проекта ОА представлены на рисунке 1.

Микрооперации, выполняемые ОА:

1. D∨0 - 1101∨0000=1101 ⇒ POH0), флаг ZF=0, так как результат не нулевой, SF=1 – старший разряд результата равен 1,

2. 2(D∨0) ⇒ SHL1(0011∨0000)=0110⇒POH1, SL_0=0, ZF=0, SF=1, флаги CF, OF – не формируются,

3. -3-L1(+3)=-3-6= -9, POH0-POH1 ⇒ 1101 – 0110 = 0111⇒ POH0, флаги ZF=0, SF=0, CF=1, OF =1, так как результат переполняет разрядную сетку. Операция вычитания производится путем формирования дополнительного кода вычитаемого (0110⇒1001+1=1010) и сложения его с первым операндом (1010+1101=0111=+7) . В процессе сложения формируется сигнал переноса из старшего разряда сумматора - C_{OUT}, равный 1. Старший (знаковый) разряд суммы при этом становится равным нулю (т.е. портится), что свидетельствует о переполнении разрядной сетки.

В таблице 3 приведены обозначения входов и выходов в проекте и в методических указаниях.

Проект по заданной работе находится в папке «\oa_full.dir»

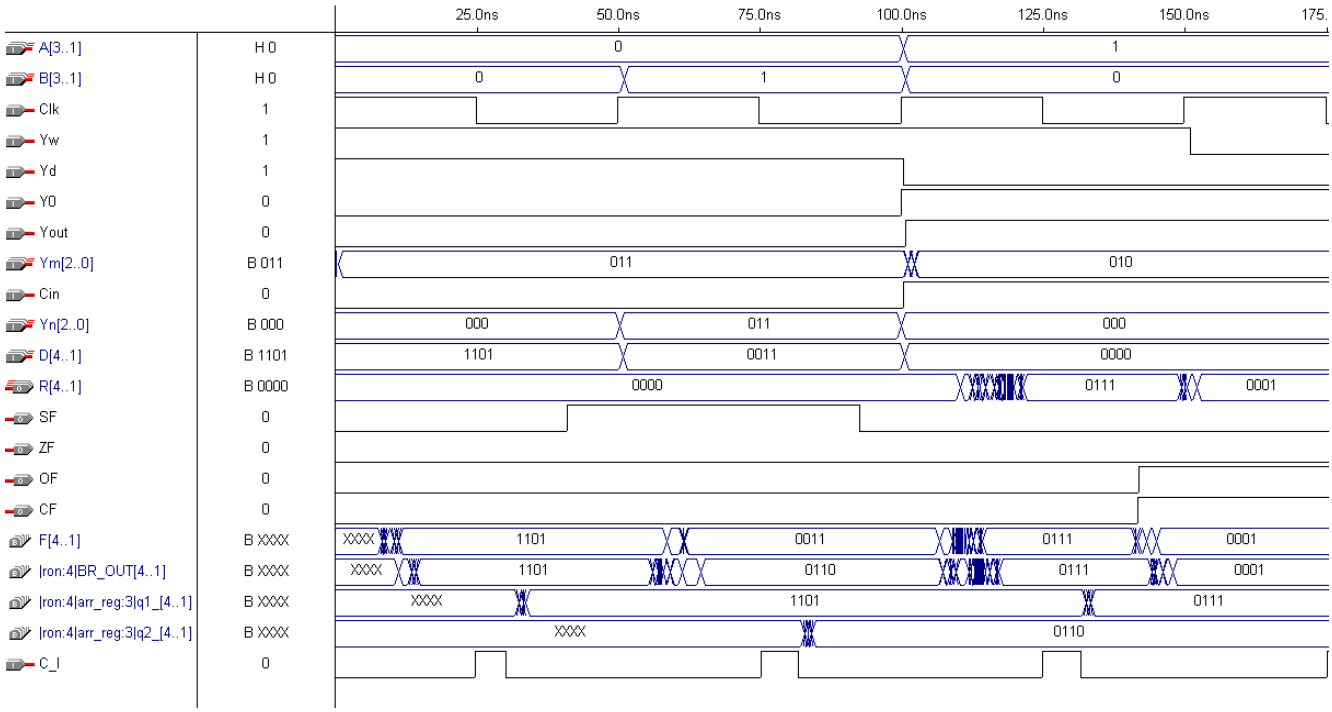


Рисунок 1 – Временные диаграммы работы ОА

Таблица 3 – Обозначения входов и выходов ОА АЛУ в проекте и в методических указаниях

Методические указания	Проект
D	D
C_0	Cin
Y	Y[3..0]
PQ	PQ[3..0]
POH	Lq0_[3..0]
PR3	PRhi
C_4	Cout
Z	Z
OVR	OVR
F_3	Fhigh

Продолжительность такта: 50 нс. Откуда тактовая частота: 20 МГц. Время задержки определено между передним фронтом ГТИ (вход Clc) и выходом R[4..0], так как на нем позднее всего появляются результаты расчетов. Передний фронт ГТИ, откуда начинаем отсчет, первый после фронта C_I, запускающего ОА. Время задержки составило 20 нс. Оно немного меньше, чем половина периода ГТИ, поскольку расчеты должны быть завершены в течение первой половины такта, во второй половине такта результаты записываются в память ОА. По внутреннему выводу |ron_4|arr_reg3|q2_[4..1], определяем быстродействие ОА, так как на этом выводе результаты появились позже всего. Итак, время задержки выполнения обеих операций в ОА составило 30 нс. Тактовая частота проекта выбрана максимальной, из расчета длительности первой из операций. Длительность этой операции определяется временем чтения и записи данных в ПЗУ. Итак, максимальная тактовая частота работы ОА АЛУ составила 20 МГц. Это много меньше, чем тактовая частота микропроцессора у современных компьютеров, но характерно для современных микроконтроллеров.

Выводы:

– изучены принципов организации ОА АЛУ;
– выполнен вариант задания в «полуавтоматическом» режиме, то есть, без УА;
– успешно проведена верификация проекта на тестовой микропрограмме;
– быстродействие ОА ограничивают микросхемы ОЗУ.

Пример отчета по лабораторной работе № 4
«Организация управляющих автоматов АЛУ»

Цель лабораторной работы – изучение принципов построения операционных автоматов арифметико-логических устройств (ОА АЛУ).

Задание на самостоятельную работу:

выполнить вариант №1 задани;. коды микропрограммы приведены в таблице 1.

Таблица 1 - Микропрограмма (для g=1 и пускового адреса a=0Ah)

Микропрограмма		Код микрокоманды, HEX	
Адрес МК	Мнемоника МК	Тип микрокоманды X	Адрес микрокоманды A
a+0 (0Ah)	Cnt (начало)	0001	-
a+1 (0Bh)	Jz 11h	1000	11h
a+2 (0Ch)	Push	0101	-
a+3 (0Dh)	Cnt	0001	-
a+4 (0Eh)	Endz	1100	-
a+5 (0Fh)	Cnt	0001	-
a+6 (10h)	Endmp (конец)	0110	-
a+7 (11h)	Cnt	0001	-
a+8 (12h)	Jmp 0Fh	0000	0Fh

На рисунке 1 приведена схема алгоритма микропрогаммы.

Результаты ее выполнения (трасса и временная диаграмма) представлены в таблице 2 и рисунке 2.

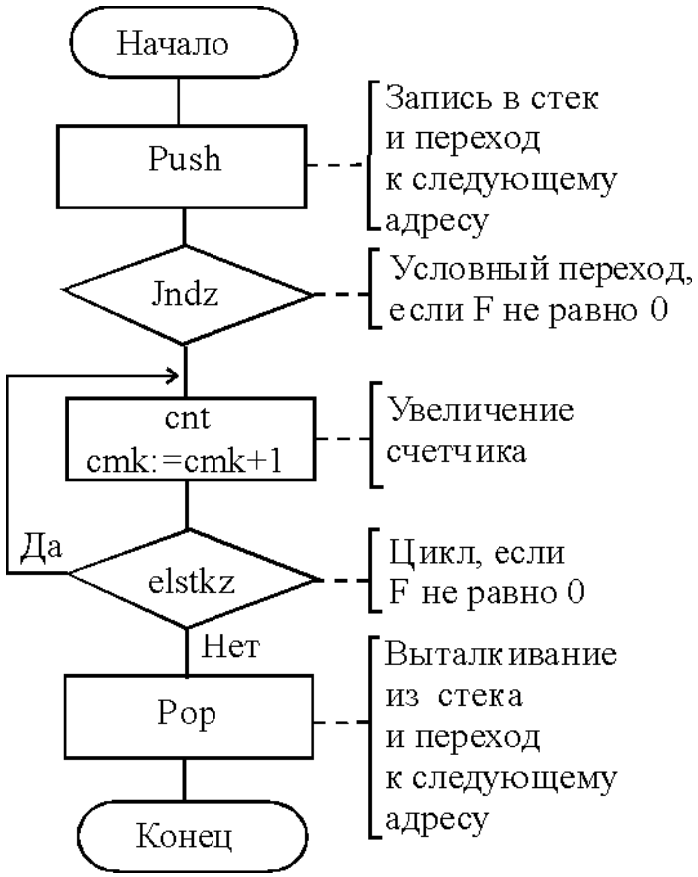


Рисунок 1 –Алгоритм микропрограммы ОА АЛУ

Таблица 2 - Трасса микропрограммы

Микрокоманда		Входы (h)				Выходы (h)					Стек (h)	
Адрес МК (h)	Мнемо-ника МК	RES	Start	g	Z	Усл. TST	Адрес сл. МК	ready	Регистр МК		R0	R1
									X	A		
01	Jmp 0Ah	1	1	1	0	0	0A	1	0000	0A	→	
0A	Cnt	0	0	0	0	0	0B	0	0001	00	→	
0B	Jz 11h	0	0	0	0	0	0C	0	1000	11h	→	
0C	Push	0	0	0	0	0	0D	0	0101	00		→0Dh
0D	Cnt	0	0	0	0	0	0E	0	0001	00		→0Dh
0E	Endz	0	0	0	0	0	0D	0	1100	00		→0Dh
0D	Cnt	0	0	0	0	0	0E	0	0001	00		→0Dh
0E	Endz	0	0	0	1	1	0F	0	1100	00	→	0Dh
0F	Cnt	0	0	0	0	0	10	0	0001	00	→	0Dh
10	endmp	0	0	0	0	0	00	1	1100	00	→	0Dh
01	Jmp 0Ah	0	1	1	0	0	0A	1	0000	0A	→	0Dh
0A	Cnt	0	0	0	0	0	0B	0	0001	00	→	0Dh
0B	Jz 11h	0	0	0	1	1	11	0	1000	11h	→	0Dh
11	Cnt	0	0	0	0	0	12	0	0001	00	→	0Dh
12	Jmp 0Fh	0	0	0	0	0	0F	0	1000	11h	→	0Dh
0F	Cnt	0	0	0	0	0	10	0	0001	00	→	0Dh
10	endmp	0	0	0	0	0	00	1	1100	00	→	0Dh
00	Jmp 00h	0	0	0	0	0	00	1	0000	00	→	0Dh
П р и м е ч а н и е - знак “→” показывает положение указателя стека УС.												

В работе использован проект , который находился в папке «Лаб работы `Организация АЛУ` - Примеры выполнения программ для лаб работы\2011\alu_full6310s.dir». В нем для компиляции использован файл ..\ua.gdf. Микропрограмма сначала была записана в память. Для этого перед компиляцией была выбрана опция Assign | Device | FLEX10KA. Далее, после компиляции, при исполнения проекта на симулятора MAX+PLUS II использован файл ..\\ua.scf. После выбора опции «Simulator», но до нажатия его клавиши «Start», была инициализирована память путем выбора опции «Initialise | Initialise memory». При этом была выбрана опция «Memory name | pzu:37:lpm_rom:LPM_ROM_component|altrom:srom|content», и затем в открывшемся диалоговом окне была выбран файл «Import file | ua(1)й.mif».

Результаты исследований

Продолжительность такта: 130 нс.

Задержка определена как время от фронта ГТИ, первого после фронта сигнала START, и до установления уровня после перехода из 0 в 1 (окончания длительности переднего фронта) вывода памяти ПЗУ: |sfam:16|A[7..0]. Этот переход был выбран, как последний на временной диаграмме из всех выводов, по которым мы определяли реакцию макета УА.

Таким образом, тактовая частота в макете была практически максимальной. При этом частота ГТИ составила 7,7 МГц. Если сравнить с ОА, то получается, что УА работаем медленнее. Очевидно, быстродействие ограничивают блоки ОЗУ.

Выводы:

– изучены принципов организации УА АЛУ;

– выполнен проект устройства без УА;

– успешно проведена верификация проекта на тестовой микропрограмме;

– оказалось, чтоУА работает медленнее, чем ОА, очевидно, это связано с быстродействием ОЗУ;

– быстродействие УА ограничивают микросхемы ОЗУ.

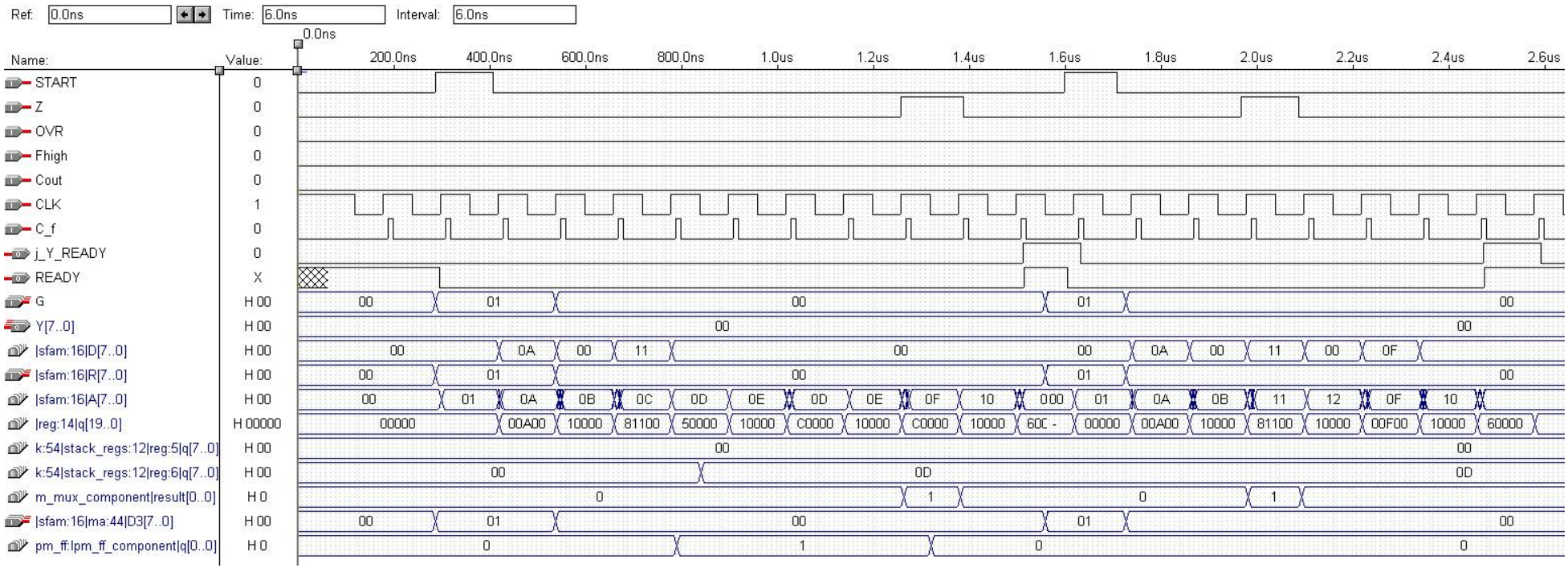


Рисунок 2—Временные диаграммы рабты проекьа УА АЛУ

Пример отчета по лабораторной работе № 5
«Организация АЛУ»

Цель лабораторной работы – изучение принципов построения и операционных возможностей арифметико-логических устройств (АЛУ)

Задание на самостоятельную работу:

1. Изучить схему и операционные возможности АЛУ.
2. Разработать микропрограмму операции в соответствии с вариантом задания, указанным преподавателем. Выполнить микропрограмму в шаговом режиме. Результаты выполнения представить в виде временной диаграммы и трассы в виде таблицы.
3. Определить задержки сигналов от входов до выходов и продолжительность такта T работы АЛУ.

Проект по лабораторной работе №5: «Лаб работы `Организация АЛУ` - Примеры выполнения программ для лаб работы\2011\alu_full0.dir». Для компиляции использовн файл ..\alu_full.gdf. Выбранный тип микросхем памяти в проекте: FLEX10K. Временные диаграммы - в файле..\\alu_full.scf. Для инициализации памяти выбрано семейство микросхем:| ua2:|pzu:37:lpm_rom:LPM_ROM_component|altrom:srom|content", для прошивки ПЗУ использван файл «Import file | alu_full(6311).mif» из папки проекта.

В таблице 1 приведена прошивка ПЗУ. Нужная нам микропрограмма находится по адресам 01H – 02H. Эта микропрограмма записана в памяти ПЗУ еще раз – по адресам 05H – 06H.

Таблица 1 – Дамп памяти ПЗУ, адреса 01H – 09H

Адрес, HEX	Данные, HEX
01	000486F800
02	2003122100
03	0004020000
04	0004020000
05	000486F800
06	2003122100
07	0004070000
08	0004020000

Порядок выполнения микропрограммы показан в таблице 2, трасса микропрограммы – в таблице 3.

Таблица 2 – Порядок выполнения микропрограммы

ГОТ	УА	WS	OS	RD	ОА	Код МК, HEX
0	cnt	0	10	00	РОН0:=D0, C_{in}=0	000486F00
1	Jd0	1	00	01	РОН0+1 $\rightarrow$ OUT, C_{in}=1	2003122100

Таблица 3 – Трасса микропрограммы

$A_{мк}$	МО в ОА	Входы, HEX				Выходы, HEX						
		Start	Start Adr	ACC WR	Вх1	Out	Acc	Вых1	Z	OVR	C	S
a+0	РОН:= D $\vee$ 0	1	0	1	FF	FF	FF	00	0	0	0	1
a+1	РОН0+1 $\rightarrow$ OUT	0	1	0	00	00	00	00	1	0	1	0

В таблице 4 приведена расшифровка кода первой и двух команд микропрограммы

Таблица 4 – Коды микрокоманды, двоичные

	Гот	R	P	WS	OS	RD	I(8:6)	I(5:3)	I(3:0)	MS	C_{in}	A	B
00	0	0000000	011	0	00	00	001	000	000	00	0	0000	0000

Обозначения в таблице 4:
ГОТ – сигнал готовности результата;
R – адресная часть;
P – поле, определяющее тип перехода;
WS – бит микрокоманды;
OS – управляющий вход;
RD – управляющее поле;
I(8:6) – приемник результата;
I(5:3) – операция;
I(2:0) – источник операндов;
MS – тип сдвига;
C_{in} – признак переноса из младшегоразряда;
A, B – адреса операндов.

На рисунке 1 приведена схема алгоритма микропрограммы

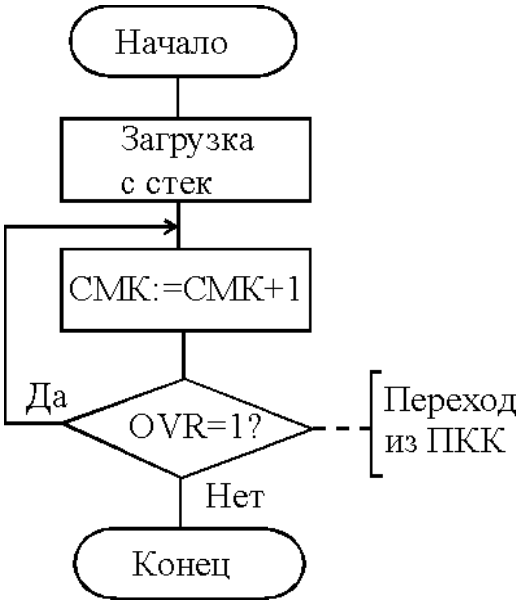


Рисунок 1 –Алгоритм микропрограммы АЛУ

В таблице 5 показано соответствие входов и выходов АЛУ в проекте и в методических указаниях.

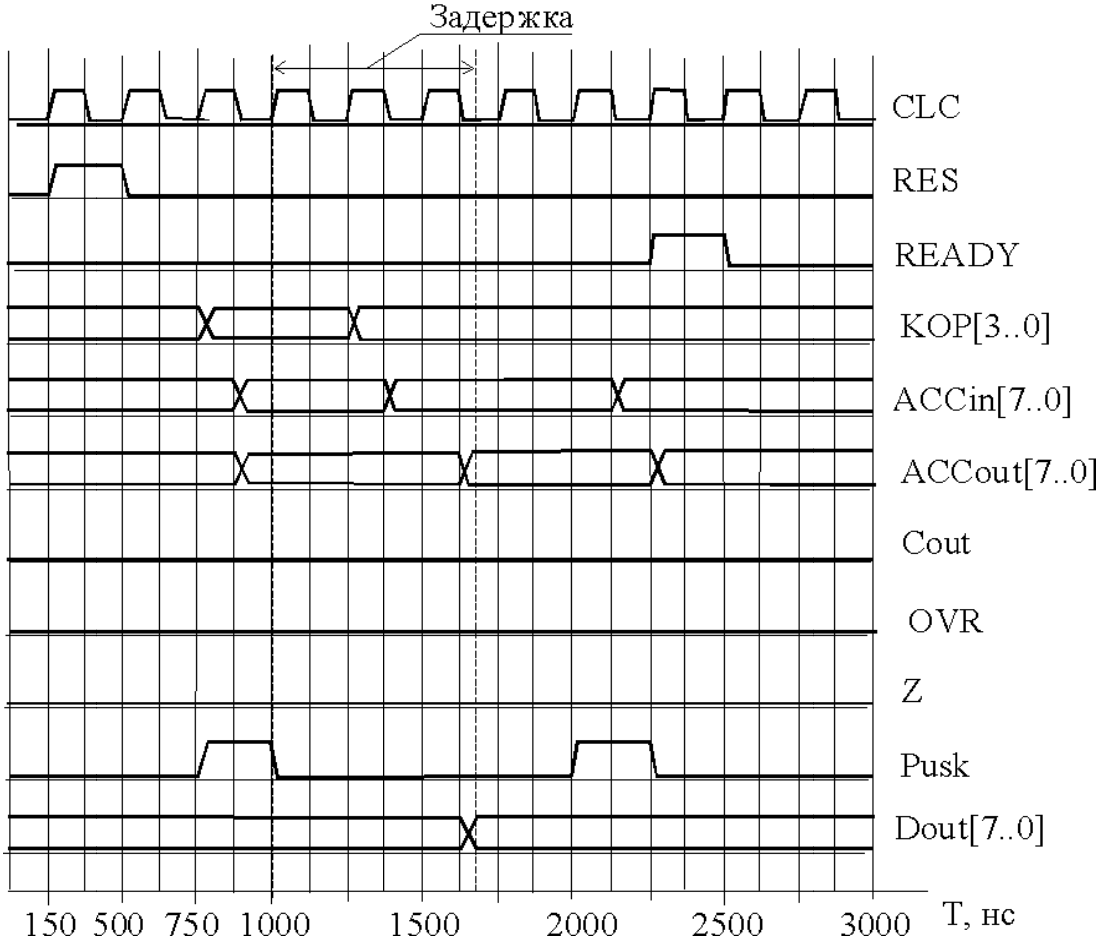
Таблица 5 – Соответствие входов и выходов АЛУ

Методические указания	Проект
Start	Pusk
ACC WR	ACC_WRITE
Start Adr	START_ADDR[pmp_a_siz-1..0]
ВХ1	ACCin[7..0]
OUT	Y[7..0]
ВЫХ1	ACCout[7…0]
Z	Z
OVR	OVR
C	Cout
S	Fhigh

На рисунке 2 приведена временная диаграмма работы проекта АЛУ, полученная симулятором MAX+PLUS II. Приведены только интересующие нас временные диаграммы.

Операционный и управляющий автоматы в составе АЛУ образуют конвейер. В то время, когда ОА выполняет i-ую команду, УА выбирает из памяти i+1-ую микрокоманду.

При такой организации в 1-ом такте работы АЛУ по микропрограмме ОА будет простаивать, поэтому общее количество тактовых импульсов N=M+1, где M – количество тактовых импульсов на выполнение самой микропрограммы непосредтсвенно.



задержка: 350 нс; такт: 250 нс

Рисунок 2 – Временная диаграмма работы проекта

Сравнивая показания временной диаграммы с трассой микропрограммы, видим, что сигналы совпадают с данными трассировки, указанной в методических указаниях. Следовательно, устройство работает нормально.

Максимальная рабочая частота была определена из задержки выполнения микрокоманды по временной диаграмме. Результат на выходе появлялся на третьем такте, считая от первого переднего фронта ТИ после фронта сигнала Pusk. Очевидно, до конца третьего такта результат должен успеть появиться на выходе АЛУ. Значит, минимальная продолжительность такта, при котором это еще происходит, определяется как

$T_{min}=(1/3)\times T_{зад.}$:

Tmin=(1/3) ×350=117 нс.

Отсюда, максимальная частота генератора тактовых импульсов

$F_{max}=1/T_{min}$:

$F_{max}=1/(117\times10^{-9})=5{,}882\times10^{6}$ Гц.

Следует округлить полученную расчетную величину в меньшую сторону, учитывая погрешность вычислений и для того, чтобы отступить от предельного значения, для надежности. Получаем F_{max}=5,8 МГц.

Выводы:

– изучена схема и операционные возможности АЛУ;

– разработана микропрограмма операции в соответствии с вариантом задания, указанным преподавателем;

– Выполнена микропрограмма в проекте, составленном на MAX+PLUS II, с помощью симулятора из этого САПР;

– определены задержки сигналов от входов до выходов и продолжительность такта T работы АЛУ;

– определена максимальная частота работы АЛУ, она оказалась много меньше, чем тактовая частота ценрального процессора современных компьютеров, и характерна больше для микроконтроллеров;

– работа АЛУ требует два такта генератора, но за счет конвейерной организации после первой команды следующие выполнятся за один такт (если нет ветвления).

Учебное издание

ОРГАНИЗАЦИЯ АРИФМЕТИКО-ЛОГИЧЕСКИХ УСТРОЙСТВ ЭВМ

Методические указания

Составители: ***Заякин Олег Александрович,***
Павлов Владимир Павлович

Редактор Ю.Н. Литвинова
Доверстка А.В. Ярославцева

Электронный ресурс

Самарский государственный
аэрокосмический университет.
443086 Самара, Московское шоссе, 34.

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.