МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

В. Ю. Мельцов

Применение САПР Quartus для синтеза абстрактных и структурных автоматов

Учебное пособие

Киров 2011

УДК 004(07)
М486

Мельцов В. Ю. Применение САПР Quartus для синтеза абстрактных и структурных автоматов. –Киров: ГОУ ВПО ВятГУ, 2011, 86 с.

В учебном пособии даются пояснения по выполнению основных этапов синтеза абстрактных и структурных автоматов, реализующих вычислительные алгоритмы для ЭВМ: выбору функциональной схемы операционного автомата, разработке и разметке содержательной ГСА, построению графа автомата и структурной таблицы переходов и выходов, выбору функциональной схемы управляющего автомата, кодированию состояний управляющего автомата, формированию логических выражений для функций возбуждения и функций выходов, построению логической схемы микропрограммного управляющего автомата, реализации операционного и управляющего автомата в САПР Quartus.

Основной целью учебного пособия является помощь в приобретении практических навыков синтеза операционного и управляющего микропрограммных автоматов с жесткой логикой на основе разработки машинных алгоритмов одной из заданных арифметических операций.

Предлагаемое учебное пособие может быть рекомендовано студентам, бакалаврам и магистрам направления 230101 – Информатика и вычислительная техника (профиль «Вычислительные машины, комплексы, системы и сети») для выполнения лабораторных работ, курсовых проектов и самостоятельной работы по дисциплинам, связанным с проектирование цифровых устройств вычислительной техники («Теория автоматов», «Схемотехника», «Микропроцессорные системы» и т.д.), а также студентам других направлений, занимающихся разработкой микропрограммных технических устройств и микроконтроллерных систем.

СОДЕРЖАНИЕ

1. ОСНОВНЫЕ ЭТАПЫ СИНТЕЗА УПРАВЛЯЮЩИХ АВТОМАТОВ

Любое вычислительное устройство может быть представлено композицией взаимодействующих пар автоматов - операционного автомата и управляющего автомата (рис.1).

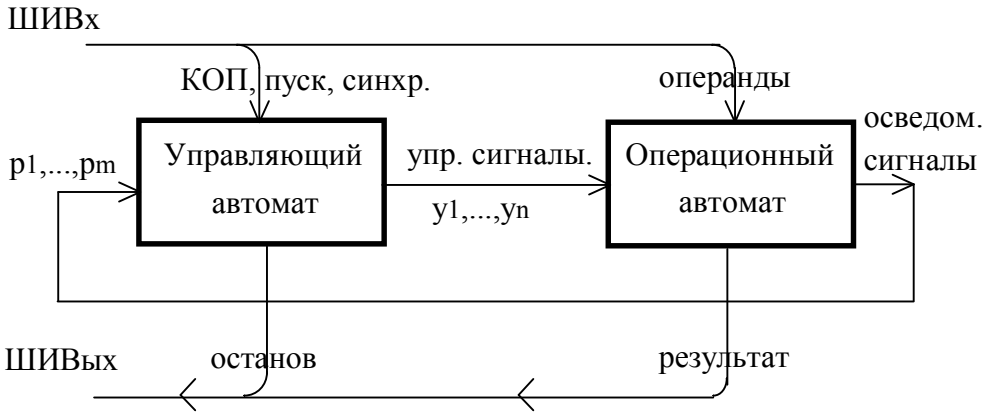


Рис. 1. Структура цифрового вычислительного устройства

Операционный автомат (ОА) содержит операционные устройства - регистры, сумматоры, счётчики, дешифраторы, мультиплексоры и др., на которых выполняется преобразование информации. В операционный автомат из других устройств ЭВМ поступают операнды по входной шине (ШИВх), а после выполнения предписанной операции результат по выходной шине (ШИВых) передается в другие устройства ЭВМ.

Управляющий автомат (УА) в соответствии с кодом операции (КОП) и внешними сигналами (пуск, синхронизация) вырабатывает множество управляющих сигналов, которые поступают в операционный автомат и изменяют состояние операционных устройств в соответствии с реализуемой микропрограммой. Порядок следования управляющих сигналов определяется специальными осведомительными сигналами, называемыми логическими условиями (ЛУ), которые формируются на устройствах операционного автомата и значения которых проверяются в каждом такте работы управляющего автомата. После завершения выполнения операции управляющий автомат посылает на ШИВых сигнал останова.

Далее следует четко определить несколько понятий, которые широко используются при синтезе вычислительных устройств.

Микрооперация (МО) - это элементарный акт обработки информации в операционном автомате на одном устройстве за один такт машинного времени под воздействием одного управляющего сигнала.

Микрокоманда (МК) - это совокупность микроопераций, выполняемых на нескольких устройствах одновременно за один такт машинного времени под воздействием нескольких управляющих сигналов.

Микропрограмма - это совокупность микрокоманд и функций перехода (зависящих от логических условий) реализуемая за несколько тактов машинного времени.

Управляющий автомат, реализующий микропрограмму работы дискретного устройства, называется микропрограммным автоматом (МПА). Существует несколько способов проектирования управляющего МПА, среди которых наиболее известны:

- управляющие автоматы с "жесткой" или схемной логикой;
- управляющие автоматы с хранимой в памяти или программируемой логикой;
- управляющие автоматы на программируемых БИС с матричной структурой.

При проектировании управляющего микропрограммного автомата с жесткой логикой можно выделить следующие основные этапы:

1. Выбор функциональной схемы ОА, определение списка микроопераций и логических условий.

2. Разработка содержательной граф-схемы алгоритма (ГСА) в соответствии со словесным описанием алгоритма заданной операции и выбранной структурой ОА.

3. Разметка содержательной ГСА и формирование отмеченной ГСА для модели Мили и (или) модели Мура.

4. Построение графа автомата и структурной таблицы переходов и выходов.

5. Выбор и обоснование функциональной схемы УА.

6. Выбор способа кодирования внутренних состояний УА, типа элементов

памяти (ЭП) и завершение формирования структурной таблицы.

7. Запись логических выражений для функции возбуждения ЭП, функций выходов и их совместная минимизация.

8. Построение логической (принципиальной) схемы управляющего МПА, цепей начальной установки, синхронизации и запуска.

Ниже будут даны некоторые рекомендации по реализации перечисленных этапов синтеза МПА.

1.1. Функциональная схема операционного автомата

Начинать проектирование управляющего МПА следует с определения структуры операционного автомата, состава операционных устройств, фиксации множества МО, необходимых для реализации алгоритма заданной операции, и множества ЛУ, определяемых в процессе выполнения алгоритма.

Этот этап проектирования выполняется параллельно с разработкой содержательной ГСА по словесному описанию алгоритма.

Далее дано описание основных операционных устройств, используемых в ОА, приведены правила их изображения в функциональных схемах и перечислены основные МО, реализуемые на каждом из устройств.

Шина с функциональной точки зрения представляет линию связи между любыми устройствами. Шина может иметь различное количество каналов в зависимости от разрядности связываемых устройств. Шины разделяют на информационные и управляющие и по-разному изображают в функциональных схемах ОА.

Любое операционное устройство (регистры, сумматоры, счетчики и др.) изображают в функциональных схемах ОА в виде условного графического обозначения (УГО) в соответствии с ГОСТ 2.708-81, ГОСТ 2.743-72 (рис.2).

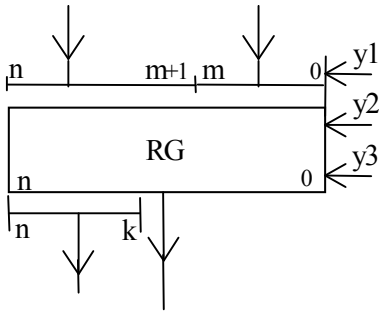


Рис.2

Высота H выбирается из ряда 10, 15 мм и далее через 5 мм, а длина L=1,5 H.

Допускается совмещать УГО функциональных узлов, если выходы одного полностью соответствуют входам другого. Внутри УГО функционального узла указывают его наименование и (или) условное обозначение.

Информационные линии связи следует подводить к большей стороне УГО сверху, а отводить от противоположной стороны УГО снизу. Если информация снимается с части разрядов УГО, то следует изображать параллельную линию с указанием над ней цифрами граничных значений разрядов, как показано на рис.2.

Управляющие линии связи подводят к любой меньшей (обычно правой!) стороне УГО или к линии продолжения этой меньшей стороны.

Р е г и с т р - упорядоченная совокупность запоминающих элементов (триггеров) со схемами управления, предназначенная для записи, хранения и выдачи информации, а также для выполнения некоторых микроопераций над этой информацией. По изображению в функциональных схемах ОА регистры разделяют на несдвиговые, изображаемые в виде прямоугольника (рис.3а). и сдвиговые, на которых может быть реализована МО сдвига содержимого (они изображаются в виде параллелограмма - рис.3б и 3в). В поле изображения необходимо указать номер регистра и его разрядность.

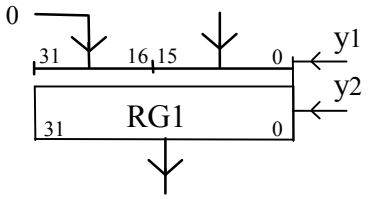


Рис. 3.а

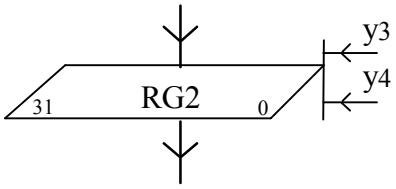


Рис. 3.б

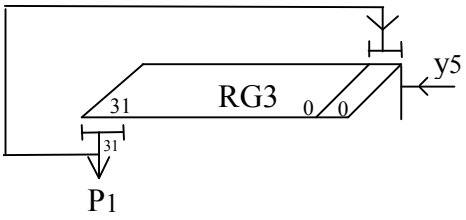


Рис.3в

На регистрах могут быть реализованы следующие МО.

y1 - занесение в RG1 (RG1[15,0] := A, RG1[31,16] := 0).

y2 - RG1 := 0 - сброс всех разрядов в ноль.

y3 - занесение в RG2 (RG2 := B).

y4 - RG2 := R1 (RG2) - сдвиг содержимого вправо на 1 разряд.

y5 - RG3 := L1 (RG3).RG3[31] - циклический сдвиг влево.

Сигнал y1 реализует микрооперацию занесения информации в регистр RG1. При этом внешние данные заносятся в младшие разряды регистра (RG1[15,0]), а в старшие разряды (RG1[31,16]) **всегда заносятся нули** (рис.3а).

Сдвиги вправо изображают заглавной латинской буквой R, влево - L с указанием рядом количества разрядов, на которое выполняется сдвиг. Например, сдвиг на четыре разряда вправо следует записать RG2 := R4 (RG2). Необходимо помнить, что в стандартных микросхемах реализованы сдвиги только на один разряд.

При сдвигах освобождающиеся разряды регистра заполняются необходимой информацией (0 или 1). По умолчанию записывается случайная информация! При реализации операций сдвига в ОК или ДК иногда требуется заполнять освобождающиеся разряды регистра единицами. В этих случаях МО сдвига следует записать так: RG2 := 1.R1 (RG2) или RG3 := L1 (RG3).0.

МО циклического сдвига y5 следует использовать в тех алгоритмах, когда необходимо сохранять операнды до конца операции (например, в алгоритмах умножения с простой коррекцией). В приведенной на рис.3в схеме в нулевой разряд RG3 [0] после сдвига запишется информация, "выталкиваемая" из тридцать первого разряда регистра RG3.

Нередко при выполнении алгоритмов арифметических операций необходимо проверить, не записан ли в регистре 0 (например, проверка делителя на нуль в операциях деления). В этих случаях используют схему логического "И", на n входов которой подают информацию с инверсных выходов триггеров регистра, а с единственного выхода схемы "И" можно снять логическое условие p2. При этом, если p2=1, в регистр занесен 0. На рис.3в показано, как можно снять другое логическое условие P1 = RG3 [31], анализ которого позволит определить, например, знак операнда в RG3.

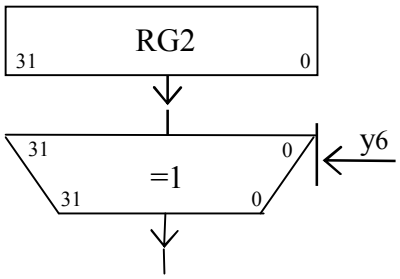


Рис.5

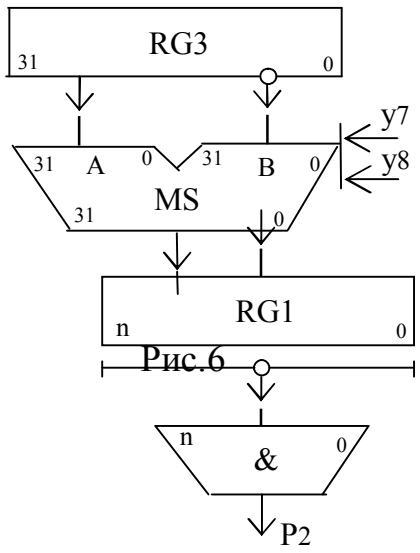


Рис.4

Рис.6

При выполнении машинных алгоритмов операций содержимое регистров необходимо передать в другие операционные устройства, чаще всего в сумматор. Если выходы триггеров регистра (прямые или инверсные) соединены шиной со входом сумматора без каких-либо промежуточных устройств, то такой съем информации называют неуправляемым. При этом содержимое регистра постоянно находится на соответствующих входах сумматора. Для управления съемом информации могут быть использованы различные операционные устройства, включаемые между выходом регистра и соответствующим входом сумматора.

Чаще всего при выполнении алгоритма в одних тактах необходимо передать в сумматор содержимое RG без инверсии, в других - с инверсией. На рис.5 представлен один из возможных вариантов реализации - с использованием совокупности схем сложения по модулю 2. Пока не подан управляющий сигнал y6, на выход совокупности схем поступает неинвертированная информация с выходов регистра.

Для реализации управления съемом информации с регистра может быть использована комбинационная схема специального вида - мультиплексор MS, о котором более подробная информация будет дана позже. На рис.6 представлен мультиплексор, на плечо A которого поступает информация с прямых выходов триггеров регистра, на плечо B - с инверсных выходов. Когда y7 не подается, на выход MS поступает информация с плеча A. Под управлением сигнала y7 к выходу MS будет подключено плечо B. Сигнал y8 позволяет получить на выходе MS нули.

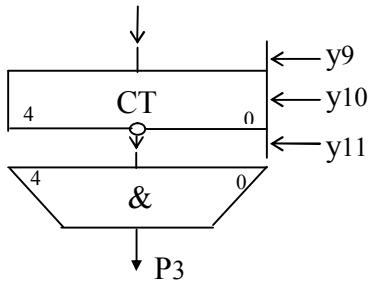


Рис.7

Замечание. Следует обратить внимание на то, что МО y6 и y7, управляющие съёмом информации с регистра, реализуются не на регистре, а на других устройствах операционного автомата.

С ч е т ч и к (двоичный) предназначен для подсчета числа выполненных тактов (например, в циклах умножения и деления) и для хранения результата счета. С точки зрения структуры счетчик (СТ) это регистр, дополненный операциями +1 и -1. В силу этой особенности в функциональных схемах устройств, реализующих операции с ПЗ, удобно порядок (или характеристику) результата операции заносить в счетчик для упрощения операции нормализации мантиссы результата. Счетчики бывают суммирующие, вычитающие и реверсивные, то есть настраиваемые на сложение или вычитание. Если N - максимальное число, которое необходимо занести в счетчик, то его разрядность определяется формулой n >=] log2 N [.

Счетчик имеет цепи установки в нулевое состояние всех триггеров, в него можно занести любое число по установочным входам триггеров, а также выполнять операции счета. Поэтому основные МО на СТ:

y9 - СТ := 0 - сброс.

y10 - занесение информации.

y11 - СТ := СТ+1 или СТ := СТ-1.

Обычно содержимое счетчика необходимо анализировать в цикле умножения (или деления), чтобы, выполнив нужное количество действий, завершить цикл. Для этого выходы триггеров счетчика следует подать на схему анализа, с выхода которой снять логическое условие. Пусть СТ работает на вычитание, и закончить цикл надо, когда содержимое счетчика равно нулю. Это можно сделать, если инверсные выходы триггеров СТ подать на схему "И", с выхода которой снять логическое условие p3 (рис.7). Цикл завершится, когда p3 = 1, т.е. когда во всех разрядах счётчика - нули. Обнаружить это же условие можно, используя вместо схемы "И" элемент Пирса с нужным количеством входов. Аналогичным образом можно обнаружить, когда во всех разрядах счетчика "записаны" единицы.

С у м м а т о р - это комбинационная схема для сложения двух двоичных чисел. В функциональных схемах ОА сумматор изображают так, как показано на рис.8, выделяя два входных плеча A и B, на которые поступают операнды, и выходное плечо S, с которого снимают результат. Кроме того, сумматор имеет вход переноса CRP и выход переноса CR, которые выделяют при изображении в тех случаях, когда они используются

На SM чаще всего реализуется МО подачи единицы на вход переноса: y12 - SMp = 1.

Если выполняется сложение операндов в ДК, это равносильно

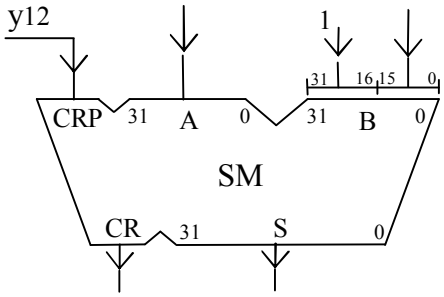


Рис.8

прибавлению единицы к младшему разряду суммы. При выполнении некоторых операций может возникнуть необходимость в занесении единиц в часть разрядов SMa[31,16] = 1.

Главное, о чем следует помнить при реализации такта сложения: сумматор - комбинационная схема, которая формирует результат на выходе S, пока поданы операнды на входы A и B, но **не хранит** результат операции. Поэтому результат операции сложения должен быть передан в том же такте в регистр для запоминания. Например, если операнды занесены в RG1 и RG2, причем операнд в RG2 отрицательный, и нужно выполнить операцию сложения с использованием ДК, то такт сложения должен быть записан так:

SMa = $\underline{\text{RG1}}$
SMв = RG2
SMp = 1
RG3 := SMs

Все МО, реализующие перечисленные выше действия, должны быть включены в одну микрокоманду (МК).

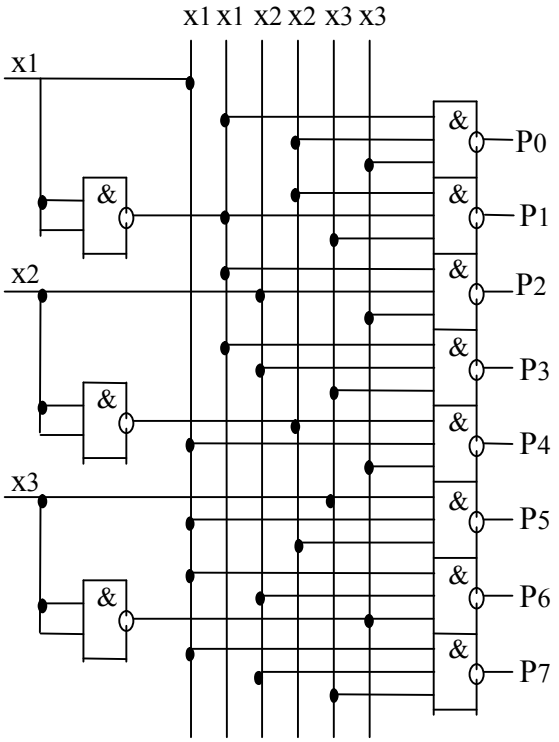


Рис.9

Д е ш и ф р а т о р (DC) - это комбинационная схема, которая преобразует код, подаваемый на его входы, в единичный сигнал на одном из выходов, то есть преобразует n-разрядное входное слово в 2^n-разрядный унитарный код. На рис.9 приведена функциональная схема дешифратора на 3 входа, выполненная на элементах И-НЕ. В функциональных схемах дешифратор условно изображают так, как показано на рис.10.

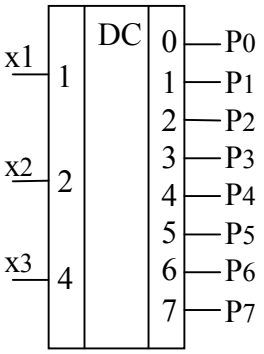


Рис.10

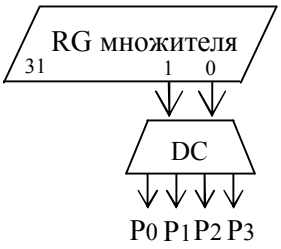


Рис.11

В функциональных схемах ОА используют дешифраторы и при реализации алгоритмов умножения с ускорением. На входы дешифратора подают группу анализируемых разрядов множителя, а с выходов снимают логические условия, анализ которых позволяет выявлять двоичные комбинации на входе.

На рис.11 приведен пример использования дешифратора при выполнении алгоритма умножения с ускорением 2-го порядка. 4 логических условия, снимаемых с дешифратора: p0, p1, p2, p3 позволяют выявить всевозможные двухразрядные комбинации в двух младших разрядах регистра множителя.

М у л ь т и п л е к с о р (MS) - это комбинационная схема, осуществляющая передачу сигналов с одной из входных линии в выходную. Выбор входной информационной линии производится кодом, поступающим на управляющие входы мультиплексора. Мультиплексор с k управляющими входами может управлять 2^k входными информационными линиями.

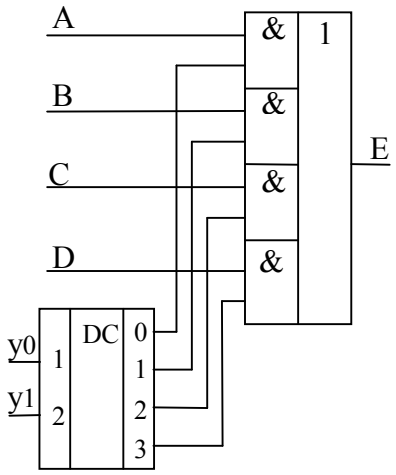


Рис.12

На рис.12 приведена функциональная схема мультиплексора с двумя управляющими входами y0 и y1, подача сигналов на которые позволяет подключать на выход E одну из четырех входных информационных линий A, B, C, Д.

В схеме использован дешифратор на 2 входа, позволяющий распознавать, какие управляющие сигналы поданы на входы y0, y1 для подключения одной из линий на выход мультиплексора.

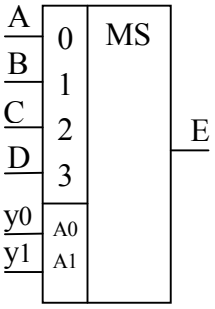


Рис.13а

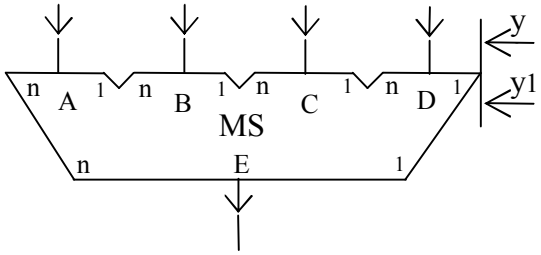


Рис.13б

На рис.13а дано условное обозначение мультиплексора на принципиальных схемах, а на рис.13б - изображение, которое используют на функциональных схемах ОА. Мультиплексор используют, например, в случаях, когда на одно плечо сумматора надо в разных тактах подавать содержимое различных регистров.

Управление выбором плеча MS, подключаемого к выходу (рис.13б), выполняется в соответствии с таблицей 1.

Таблица 1

Управляющие сигналы	-	Y0	Y1	Y0, Y1
Подключить к выходу E	A	B	C	D

Из приведенной таблицы видно, что когда ни один из управляющих сигналов не подан, к выходу E подключено плечо A, подача управляющего сигнала y0 подключает плечо B, сигнала y1 - плечо C, подача же обоих сигналов y0, y1 обеспечивает подключение к выходу E плеча D.

Таким образом, управляющие сигналы y0 и y1 осуществляют выбор одной из четырех входных информационных линий. Если управляющих сигналов больше, то и количество входных линий значительно возрастает. Например, при k = 3 возможен выбор одной из 8 линий, при k = 4 выбор из 16 линий и т.д. Существуют варианты схем MS, в которых имеется дополнительный управляющий сигнал разрешения подключения к выходу, без подачи которого информация ни с одной из информационных линий не поступает на выход (состояние высокого импеданса) или дополнительный сигнал обнуления выходной информации.

Таковы основные операционные устройства, используемые в функциональных схемах ОА. В них, кроме того, могут использоваться отдельные триггеры (например, для фиксации ПРС), сумматоры по модулю 2 (для определения знака результата), инверторы (для формирования цифр частного) и другие.

1.2 Разработка содержательной граф-схемы алгоритма

Микропрограмма работы проектируемого устройства может быть описана в виде ориентированного связного графа, содержащего одну начальную вершину, одну конечную вершину и конечное число операторных и условных вершин. В каждой операторной вершине записывается одна МК, содержащая одну или несколько МО, а в каждой условной вершине - одно из логических условий.

Допускается запись одинаковых МК в различных операторных вершинах и одинаковых ЛУ в различных условных вершинах. Такую запись микропрограммы в виде графа называют содержательной ГСА.

При записи микропрограммы на языке ГСА следует руководствоваться следующими правилами.

1. Начальная вершина должна иметь один выход, конечная - один вход, операторная - один вход и один выход, условная - один вход и не менее двух выходов. В логической вершине допустимо вывести один из выходов на собственный вход, реализовав таким образом режим ожидания.

2. Из любой вершины ГСА должен существовать хотя бы один путь в конечную вершину.

3. К входной стрелке операторной вершины может подходить несколько выходных стрелок от других вершин ГСА, при этом каждый выход вершины соединен точно с одним входом.

Содержательная ГСА составляется в полном соответствии со словесным описанием алгоритма выполнения заданной операции. Далее в соответствии с выбранной структурой ОА составляется список МО и ЛУ. Теперь каждой операторной вершине содержательной ГСА можно сопоставить набор МО, образующих МК.

1.3 Разметка содержательной граф-схемы алгоритма

Для синтеза микропрограммного автомата необходимо выполнить переход от содержательной ГСА к отмеченной ГСА путем выделения состояний автомата (разметки) в соответствии о моделью Мили или моделью Мура.

Предварительно в каждой условной вершине проставляются символы из множества входных сигналов УА - X_1, X_2, ... ,X_M, каждый из которых эквивалентен одному из осведомительных сигналов ОА. Во всех операторных вершинах ГСА проставляют символы из множества выходных сигналов управляющего автомата - $У_1$, $У_2$, ... , $У_N$, эквивалентных одной из МК, поступающих в ОА в соответствующие такты машинного времени. Удобно в

каждой операторной вершине ГСА вслед за символом МК указать в скобках набор МО, образующих каждую МК. Допускается запись одинаковых символов X_i и Y_j в различных условных и операторных вершинах.

Выделение состояний автомата Мили, то есть разметка ГСА в соответствии с моделью Мили, выполняется по следующим правилам:

1. Вход вершины, следующей за начальной, и вход конечной вершины отмечаются символом начального состояния автомата a_0.

2. Входы всех вершин, следующих за операторными, отмечаются символами $a_1, \ldots, a_K$.

3. Если вход вершины отмечается, то только одним символом.

4. Входы различных вершин за исключением конечной отмечаются различными символами.

Выделение состояний автомата Мура, то есть разметка ГСА в соответствии с моделью Мура, выполняется по следующим правилам.

1. Символом начального состояния a_0 отмечаются начальная и конечная вершины.

2. Различные операторные вершины отмечаются различными символами $a_1, \ldots, a_K$.

3. Все операторные вершины должны быть отмечены, то есть каждой МК, отдельно представленной в ГСА ставится в соответствие отдельное состояние автомата Мура.

4. В логических вершинах ГСА, реализующих режим ожидания, существует возвратная дуга, когда один из выходов вершины подан на ее вход. На этой дуге необходимо вводить дополнительное фиктивное состояние автомата Мура.

Для одной и той же ГСА количество состояний для модели Мура, как правило, больше, чем для модели Мили. Однако при проектировании управляющего МПА трудно заранее определить, какая из моделей - Мили или Мура - даст комбинационную схему УА меньшей сложности. Поэтому, чаще всего, на начальном этапе проектирования предлагается исследовать обе модели.

1.4. Построение графа автомата и структурной таблицы переходов и выходов

Имея отмеченную ГСА проектируемого управляющего микропрограммного автомата, следует описать его работу известными способами - графическим и табличным. Если количество состояний автомата и переходов между ними невелико, то задание его в виде графа позволяет наглядно представить работу МПА.

Граф автомата есть ориентированный связный граф, вершины которого соответствуют состояниям, а дуги - переходам между ними. Причем, две вершины графа a_m и a_s - соединены дугой, направленной от a_m (исходное состояние) к a_s (состояние перехода) если в ГСА существует этот переход.

Для автомата Мили каждой дуге приписываются входные и выходные сигналы, если они определены. Для автомата Мура дугам приписаны только входные сигналы; выходные сигналы приписаны вершинам графа.

Замечания.

1. В графе автомата необходимо указывать все возможные переходы между состояниями, "проходя" встречающиеся на пути из am в as условные вершины по всем исходящим из них дугам.

2. При построении графа автомата Мили необходимо избегать "пустых" переходов, то есть переходов, на которых не вырабатываются управляющие сигналы (или на которых не встречается операторной вершины, что эквивалентно). Чтобы не снижать быстродействия МПА, надо в графе показать переход в следующее состояние, если это возможно.

Если автомат имеет большое число состояний и переходов между ними, то наглядность графа теряется. Тогда удобно использовать табличный способ задания автомата. При синтезе МПА строят прямые (или инверсные) структурные таблицы переходов и выходов.

Таблица 2.

Исх. Состояние a_m	Код исх. состояния $K(a_m)$	Состояние перехода a_s	Код состояния перехода $K(a_s)$	Входные сигналы $X(a_m, a_s)$	Выходные сигналы $Y(a_m, a_s)$	Функции возбуждения ЭП $F(a_m, a_s)$
1	2	3	4	5	6	7

В таблице 2 дана прямая структурная таблица для автомата Мили. Для автомата Мура столбец 6 таблицы (выходные сигналы) следует располагать вслед за первым столбцом.

Состояния, перечисляемые в первом столбце таблицы, должны быть упорядочены, то есть сначала следует указать все переходы из a_0, затем из a_1 и т.д. Аналогично и для инверсной таблицы переходов (все переходы в a_0, в a_1 и т.д.). При формировании столбца 5 (входные сигналы) следует указывать конъюнкцию всех входных сигналов, записанных в логических вершинах ГСА на данном переходе. Причем X_i берут без отрицания, если переход выполняется по единичному значению сигнала, и с отрицанием, если по нулевому значению сигнала.

Первоначально столбцы 2 и 4 таблицы (коды состояний) не могут быть заполнены, так как еще не выполнено кодирование состояний автомата. По этой же причине не могут быть определены и функции возбуждения элементов памяти (столбец 7). Формирование структурной таблицы будет завершено позднее.

1.5 Выбор и обоснование структурной схемы управляющего автомата

Этот этап структурного синтеза выполняется параллельно с выбором способа кодирования внутренних состояний и типа элементов памяти для управляющего автомата.

Вариант 1. Классическая структура УА - это совокупность взаимосвязанных элементов памяти (ЭП) и комбинационной схемы (КС), реализующей функции возбуждения ЭП и функции выходов Y (рис.14).

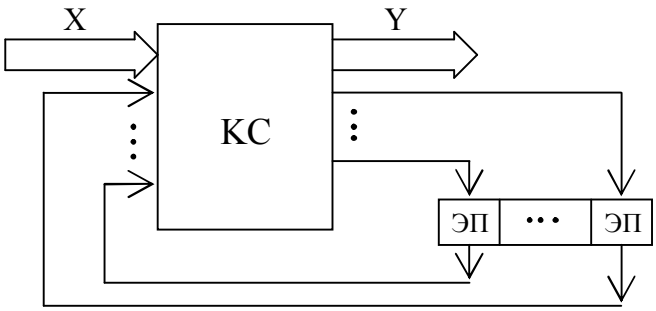


Рис.14

Набор ЭП фиксирует состояния, в которые переходит автомат в процессе выполнения заданной микропрограммы. Синтез УА сводится к синтезу его КС, на вход которой поступает множество входных сигналов X из ОА и сигналы, фиксирующие коды состояний автомата.

Естественным требованием при синтезе КС является минимизация ее цены, что обеспечивается правильным выбором способа кодирования состояний и типа ЭП, а также совместной минимизацией функций возбуждения ЭП и функций выходов.

Вариант 2. Цена КС может быть снижена, если сигналы с выходов ЭП подать на дешифратор ДС, что приводит к структуре, изображенной на рис.15.

В такой структуре УА необходимо стремиться к полному использованию выходов дешифратора, Если дешифратор, подключаемый к выходам всех ЭП, недоиспользуется, целесообразно продумать вариант установки дешифратора, подключаемого к части выходов ЭП.

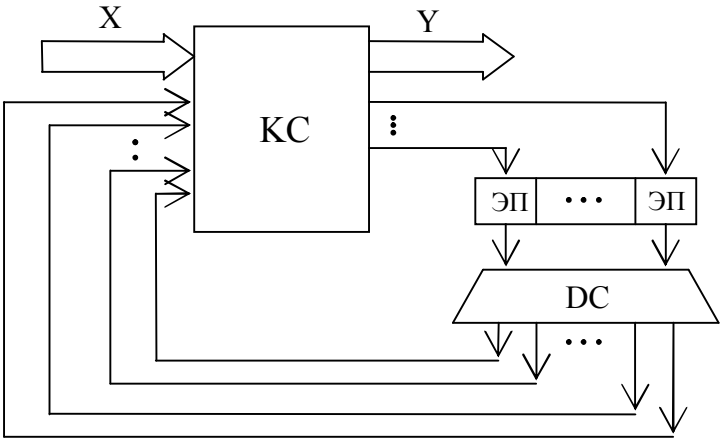


Рис.15

Вариант 3. В случае выбора унитарного способа кодирования внутренних состояний автомата, элементы памяти объединяются в регистр сдвига, что приводит к структуре УА, изображенной на рис.16.

Тогда часть комбинационной схемы, реализующая функции возбуждения элементов памяти, значительно упрощается, так как в этом случае необходимо вырабатывать лишь сигнал первоначальной установки регистра и сигнал сдвига содержимого регистра (в качестве которого часто используют сигнал синхронизации).

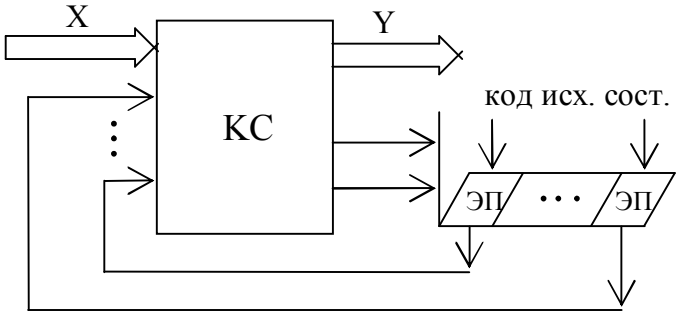


Рис.16

Вариант 4. Когда граф проектируемого автомата имеет большое количество последовательных (стандартных) переходов и незначительное количество нестандартных переходов, целесообразно закодировать состояния автомата последовательными двоичными числами. Это и диктует структуру

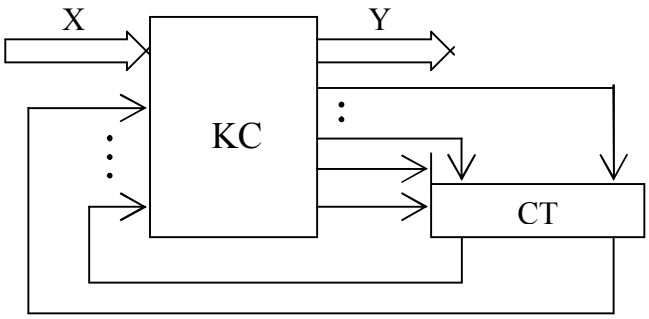


Рис.17

УА на основе двоичного счетчика, работающего в режиме сложения или вычитания. При этом на последовательных переходах необходимо подавать сигнал на счетный вход счетчика, что обеспечит стандартные переходы. На нестандартных переходах следует устанавливать нужный код по установочным входам счетчика. Если число нестандартных переходов невелико, то КС, реализующая функции возбуждения ЭП, будет проста.

Вариант 5. Структуру на основе двоичного счетчика можно модифицировать, используя дешифратор, подключаемый к выходам триггеров счетчика. Получим вариант структуры УА, аналогичный варианту 2 (рис.15) со счетчиком, используемым в качестве ЭП.

В заключение этого параграфа следует отметить, что возможны и другие варианты структур УА. Однако, учитывая содержание заданий, предлагаемых студентам для выполнения курсовой работы, а также то, что данная работа имеет своей целью приобретение первых практических навыков проектирования МПА, которые при дальнейшем обучении будут расширены и углублены, можно на начальном этапе ограничиться предлагаемыми выше вариантами структур УА.

1.6. Кодирование состояний управляющего автомата

Выбор способа кодирования внутренних состояний автомата, типа используемых элементов памяти и варианты структуры МПА - взаимосвязанные вопросы, решаемые параллельно. От их правильного решения зависит сложность

комбинационной схемы проектируемого МПА.

Основным ориентиром при выборе структуры и метода кодирования является граф автомата, наглядно представляющий переходы между состояниями при выполнении автоматом заданной микропрограммы.

Кодирование заключается в сопоставлении каждому состоянию МПА набора состояний элементов памяти одинаковой длины. Если *N* - число состояний автомата, *n* - число ЭП, используемых для кодирования состояний, то *n* определяется из интервала

$$] \log_2 N [<= n <= N ,$$

То есть наименьшее число ЭП **n** есть ближайшее большее целое от $\log_2 N$, а наибольшее число ЭП равно числу состояний автомата N.

Формального способа выбора метода кодирования состояний автомата предложить нельзя, можно лишь дать некоторые рекомендации, а окончательный выбор остается за разработчиком проектируемого МПА, который он делает после сопоставления вариантов.

Наиболее простым является унитарный способ кодирования, при котором n = N. Этот способ кодирования в совокупности с вариантом 3 структуры УА целесообразен в тех случаях, когда]log2 N[ненамного меньше *N*, иначе будут значительными затраты на память автомата, которые поглотят выигрыш от уменьшения цены КС, формирующей функции возбуждения ЭП

Другой простой способ кодирования связан с вариантами 4 и 5 структур УА на основе счетчика. Здесь состояния автомата кодируются последовательными двоичными числами, а формирование функций возбуждения ЭП на нестандартных переходах определяется типом элементов памяти, на которых построен счетчик. Этот способ кодирования и структуры УА целесообразен, как уже указывалось, для графов автомата, имеющих большое количество последовательных переходов.

Далее несколько слов о связи типа ЭП со способом кодирования. Среди различных типов триггеров особое место занимают синхронизируемые D-триггеры, имеющие вход данных D и вход синхронизации C. По сигналу C-

входа триггер переключается в состояние, предписываемое D входом.

При использовании D-триггеров в качестве ЭП для получения смены состояний на каждом переходе (a_m -> a_s) сигналы возбуждения должны быть поданы на те триггеры, которые в коде состояния перехода a_s содержат "1". Отсюда основное требование к выбору кодов состояний: чем больше переходов в какое-либо состояние, тем меньше "1" должен содержать код этого состояния. Здесь удобно строить инверсные таблицы переходов. Этот способ кодирования позволит получить КС меньшей сложности.

Для триггеров, имеющих отдельные входы установки в "1" и в "0", (RS - триггеры, JK - триггеры) целесообразно использовать метод кодирования, минимизирующий число переключений ЭП, в сочетании с методом соседнего кодирования. При этом сначала следует выделить пары соседей 1-го и 2-го рода и закодировать их соседними кодами, используя диаграмму Вейча-Карно. А затем применить метод, минимизирующий число переключений ЭП.

Замечание.

При выборе числа элементов памяти следует учитывать, что в стандартных сериях логических элементов одна микросхема может содержать 2, 4 или 8 триггеров. Поэтому, если по расчетам, например, минимальное число ЭП получилось равным 3, то можно использовать 4 элемента памяти. Это позволит в большинстве случаев уменьшить цену КС для функций возбуждения ЭП.

Таким образом, после выбора типа ЭП и кодирования внутренних состояний автомата следует вернуться к структурной таблице переходов автомата и заполнить столбцы кодов состояний и столбец обязательных функций возбуждения ЭП.

При определении сигналов, которые должны быть поданы на входы триггеров для получения требуемой смены состояний на каждом переходе автомата, следует учитывать следующее.

При использовании D-триггеров сигналы возбуждения подаются на те триггеры, для которых в коде состояния перехода a_s записаны "1", то есть

анализируется столбец кодов состояний перехода $K(a_s)$ структурной таблицы.

При использовании RS- и JK-триггеров сигнал возбуждения подают на S-вход (J-вход), если на переходе триггер требует смены состояний 0 -> 1; сигнал возбуждения надо подать на R-вход (K-вход), если требуемая смена состояний триггера 1 -> 0.

При использовании T-триггеров сигналы возбуждения подают на те триггеры, которые изменяют свое состояние на переходе (0 -> 1, 1 -> 0).

1.7. Формирование логических выражений для функций возбуждения и функций выходов

По структурной таблице переходов и выходов можно построить логические выражения для всех выходных сигналов, формируемых КС.

Для формирования функций возбуждения ЭП из последнего столбца структурной таблицы выбираются строки, соответствующие определенной функции возбуждения. Для каждой строки составляются конъюнкции кодов исходных состояний и входных сигналов, записанных в данной строке. Образованные таким образом конъюнкции объединяют знаками дизъюнкций для всех строк, содержащих формируемую функцию возбуждения.

Аналогично записывают булевы выражения для функций выходов ориентируясь на столбец выходных сигналов $Y(a_m, a_s)$ структурной таблицы переходов и выходов. Для автомата Мили каждый управляющий сигнал содержит конъюнкции кодов состояний и входных сигналов, объединенных знаками дизъюнкций для всех строк, содержащих формируемый управляющий сигнал. Для автомата Мура управляющие сигналы есть дизъюнкции состояний автомата, отмеченных соответствующими выходными сигналами.

После совместной минимизации полученной системы логических выражений для функций возбуждения ЭП и функций выходов можно перейти к построению КС в заданном логическом базисе.

1.8. Построение функциональной схемы управляющего МПА

Полученные на предыдущем этапе логические выражения для функций возбуждения ЭП и функций выходов позволяют построить комбинационную схему, реализующую эти функции. При этом может быть использован как основной логический базис И, ИЛИ, НЕ, так и любой другой базис по заданию преподавателя.

Построенная комбинационная схема в совокупности с набором ЭП и, быть может, другими элементами, устанавливаемыми в соответствии с выбранной структурой управляющего автомата, и дают функциональную схему управляющего микропрограммного автомата. При изображении схемы следует руководствоваться соответствующим ГОСТ.

Схема МПА должна иметь цепи начальной установки автомата в исходное состояние и цепи включения автомата на однократное выполнение алгоритма по запускающему сигналу. Кроме того в схему поступает сигнал синхронизации от генератора тактовых импульсов.

Для реализуемых в курсовой работе алгоритмов первой, после начальной вершины ГСА, является вершина ожидания поступления операндов с ШИВх. Единичный выход этой логической вершины и является фактически сигналом запуска автомата на однократное выполнение алгоритма. Так как при разметке ГСА начало и конец микропрограммы отмечены начальным состоянием a_0, то автоматически происходит сброс в начальное состояние после завершения микропрограммы. Таким образом обеспечивается многократное повторение алгоритма с поступлением следующих операндов в МПА.

Цепи начальной установки необходимы в связи с тем, что после включения питания состояния элементов памяти могут быть произвольными, а для правильного функционирования автомата его необходимо установить в начальное состояние сигналом *b*.

При формировании цепей начальной установки следует учитывать как код исходного состояния, так и тип триггеров, используемых в качестве ЭП.

Пусть, например, исходное состояние автомата имеет код 01. Тогда, если в

качестве ЭП используются D-триггеры, то реализовать цепи начальной установки следует так, как показано на рис.18а, где D_1 и D_2 -соответствующие функции возбуждения D-триггеров. Если же в качестве

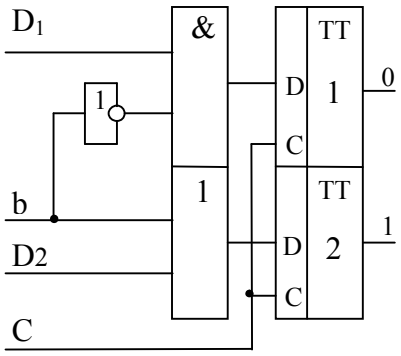


Рис.18а

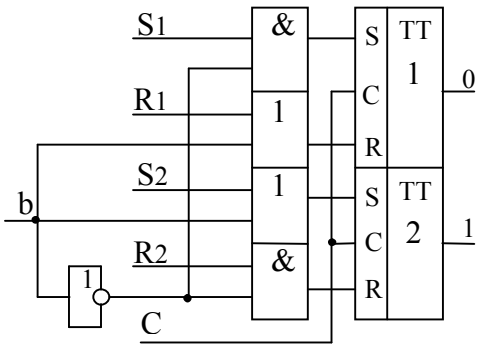


Рис.18б

ЭП используется RS-триггер, то цепи начальной установки показаны на рис.18б, где R_1S_1, R_2S_2 - функции возбуждения для 1 и 2 триггеров.

Когда в схемах D-триггеров есть отдельные асинхронные входы установки и сброса, то их используют для подачи сигнала начальной установки, а функции возбуждения ЭП поступают на соответствующие синхронные входы триггеров.

2. СИНТЕЗ АБСТРАКТНЫХ АВТОМАТОВ

Внимание! Данный пример приведен с целью показа функционирования максимально широкого числа элементов схемотехники и не является оптимальным для реализации заданной операции. Основным требованием к курсовому проекту, выполняемого студентами по курсу «Теория автоматов», остается **минимизация аппаратурных затрат** (как УА так и ОА) при приемлемом быстродействии!

Используемые коды. Операнды разрядностью 4 байта (24 разряда мантисса и 8 разрядов порядок) поступают по входной шине (ШИВх) в прямом коде (ПК), результат также в ПК выводится по выходной шине (ШИВых). В процессе выполнения операции может возникнуть ситуация переполнения разрядной сетки (ПРС).

2.1 Описание алгоритма умножения чисел с ПЗ

В качестве основной операции выступает операция умножения двоичных чисел первым способом в прямом коде (ПК) с плавающей запятой (ПЗ), с характеристикой. Главной особенностью данного способа умножения является возможность использования **n-разрядных** регистров для хранения мантисс операндов и результата и, соответственно, **n-разрядного** сумматора мантисс.

Алгоритм состоит из следующих пунктов:

1. Определить знак произведения сложением по модулю два знаковых разрядов сомножителей, и далее использовать модули операндов.

2. Проверить множимое на равенство нулю: если равно нулю, операцию умножения следует прекратить, т.к. результат будет также равным нулю.

3. Проверить множитель на равенство нулю: если равен нулю, операцию умножения следует прекратить, т.к. результат будет также равным нулю.

4. Сложить характеристики сомножителей. При этом могут возникнуть следующие ситуации: переполнение разрядной сетки (ПРС), временное ПРС или потеря младших разрядов (ПМР). Если возникло ПРС (признаком ПРС является получение единицы переноса и единицы в старшем разряде результирующей характеристики), то необходимо зафиксировать её появление и прекратить операцию. Временное ПРС может возникнуть, когда в старший разряд равен единице, образовалась единица переноса, но все разряды характеристики, за исключением старшего, равны нулю. При этом нужно продолжить алгоритм умножения. Если возникла ситуация ПМР (признаком ПМР является отсутствие единицы переноса и ноль в старшем разряде результирующей характеристики), то необходимо зафиксировать её появление и выдать нулевой результат. В противном случае переходим к пункту 5.

5. Анализ младшей цифры очередного разряда множителя: если цифра множителя «1», то суммировать множимое с накопленной суммой частичных произведений (ЧП). В результате суммирования может возникнуть ситуация временного ПРС в мантиссе, которая устраняется последующим сдвигом вправо.

6. Выполнить в основном цикле - сложение мантисс операндов и сдвиги: множителя на один разряд вправо, сумы частичных произведений на один разряд влево.

7. После цикла умножения необходимо провести проверку на необходимость нормализации результата. Если произведение денормализовано, провести нормализацию результата: сдвинем произведение на 1 разряд влево, вычтем “1” из характеристики. При этом, если ранее было зафиксировано временное ПРС, оно устраняется. Если после нормализации мантиссы произошло ПМР, нужно зафиксировать её появление и вывести результат равный «0»; в противном случае переходим к пункту 8.

Денормализация возможна лишь на один разряд, т.к. операнды поступают на входную шину уже нормализованными. Если результат нормализован, необходимо проверить, было ли зафиксировано временно ПРС. Если да, то установить признак ПРС и операцию необходимо прекратить.

8. Присвоить знак модулю произведения из п. 1 алгоритма. Если после нормализации результата зафиксирован признак ПМР, то в качестве знака результата необходимо выдать ноль.

2.2 Численный пример

Рассмотрим алгоритм умножения на числовом примере:

$A = -2{,}375_{10} = 1{,}10010\ \ 10011000_2$

$B = 0{,}171875_{10} = 1{,}01110\ \ 10110000_2$

1) $1 \oplus 0 = 1$

2) Суммируем характеристики

$$\begin{array}{r} 1{,}0010 \\ 0{,}1110 \\ \underline{10},0000 \end{array}$$

ПРС и ПМР не возникло.

3) Последовательность операций умножения представлена в табл. 3

Таблица 3 – Умножение двоичных чисел первым способом с ПЗ

Множитель	Сумма ЧП	Комментарии
0,100 11000	0,00000000 00000000	Сдвиг вправо
0,010 01100	0,00000000 00000000	Сдвиг вправо
0,001 00110	0,00000000 00000000	Сдвиг вправо
0,000 10011	0,00000000 00000000 0,10110000 0,10110000 00000000	Сложение Сдвиг вправо
0,000 01001	0,01011000 00000000 0,10110000 1,00001000 00000000 (временное ПРС)	Сложение Сдвиг вправо
0,000 00100	0,10000100 00000000	Сдвиг вправо
0,000 00010	0,01000010 00000000	Сдвиг вправо
0,000 00001	0,00100001 00000000 0,10110000 0,11010001 00000000	Сложение Сдвиг вправо
0,000 00000	0,01101000 10000000	

Сумма ЧП = 0,01101000 10000000 – требуется нормализация. Сдвинем результат на 1 разряд влево, и вычтем единицу из суммы характеристик.

0,0000

1,1111

1,1111 0,1111 (-1) - ПМР не возникло.

Результат: 1,01111 $11010001_2 = -0{,}408203_{10}$

Проверка -2,375 * 0,171875 = -0,408203

2.3 Разработка функциональной схемы операционного автомата

Операнды разрядностью 32 байта поступают в операционный автомат (ОА) в ПК по входной шине. Первый операнд - множимое. Запись мантиссы множимого производится в RG1 и RG2. Знак числа записывается в Т1 и в 23-й разряд RG1. В RG4 происходит запись характеристики (со знаком) множимого. Выполняется проверка операнда на ноль, если операнд равен нулю, то на выходную шину подаем ноль. В счетчик СТ1 записывается значение регистра RG4. В счетчик циклов СТ2 записывается значение «001001». Вторым операндом приходит множитель. Запись знака и мантиссы множителя осуществляется в RG1, запись его характеристики в RG4. Выполняется проверка операнда на ноль, если операнд равен «0», то на выходную шину подается «0». В СТ1 записывается значение суммы характеристик. Если возникло ПРС характеристик, триггер Т2 устанавливаем в единицу. При этом может возникнуть временная ПРС, которая впоследствии может быть исправлена при нормализации мантиссы. Если же ПРС не возникло, то в цикле умножения производится анализ младшего разряда множителя, если он равен единице, то в RG3 заносим сумму множимого и частичной суммы. После этого производится сдвиг RG1 и RG3 вправо, а значение СТ2 увеличивается на 1. Цикл умножения заканчивается, когда в старшем разряде СТ2 появится «1». Если старший разряд RG3 равен нулю, производится нормализация. В противном случае необходимо проверить а было ли зафиксировано временное ПРС. Если да, то устанавливаем триггер Т2 в единицу. Если после нормализации возникло ПМР, то обнуляем RG3, RG1 и счетчик СТ1 и выдаем на выходную шину ноль.

Для выдачи результата на выходную шину содержимое RG3, СТ1 подается на усилитель формирователь. Причем старший разряд СТ1 инвертируется.

Для организации работы операционной части из управляющей части автомата (УА) подаются следующие управляющие сигналы:

y_0 – запись в RG1, RG4;

y_1 – запись в RG2, T1, установка T2 в положение «0», обнуление RG3 и CT1, запись CT2;

y_2 – запись в CT1 значения выхода SM2;

y_3 – сдвиг RG3 влево, CT1:=CT1-1;

y_4 – запись в RG3 значения выхода SM1, запись в T2 значения выхода переноса SM1;

y_5 – сдвиг RG1 и RG3 вправо, CT2:=CT2+1;

y_6 – обнуление T1 и RG1;

y_7 – установка T2 в положение «1»;

y_8 – выдача результата на выходную шину.

Из ОА в УА необходимо передавать осведомительные сигналы о состоянии ОА, которые определяются следующим списком логических условий:

X – проверка наличия операндов на входной шине;

p_1 – проверка на ноль;

p_2 – проверка на временное ПРС;

p_3 – младший разряд RG1 (проверка очередной цифры множителя);

p_4 – старший разряд RG3 (проверка нормализации результата);

p_5 – проверка на ПМР;

p_6 – проверка на окончание операции умножения;

p_7 – проверка на ПРС;

Z – проверка возможности выдачи результата на шину выхода.

Таким образом, УА должен вырабатывать 9 управляющих сигналов и посылать их в ОА в нужные такты машинного времени в соответствии с алгоритмом выполнения операции умножения, учитывая 8 осведомительных сигналов, поступающих из ОА. Функциональная схема (ФС) ОА изображена на рисунке 3.1.

2.4 Разработка содержательной ГСА

Содержательная граф-схема алгоритма представлена на рисунке 4.1.

В первом такте производится проверка наличия на входной шине множимого (блок 1). При поступлении множимого, его мантисса заносится в RG2 и RG1, знак заносится в T1, в RG4 заносится значение характеристики, RG3, CT1 и T2 обнуляются, а в CT2 заносится значение «001001» (блок 2). Затем производится проверка на ноль мантиссы множимого (блок 3). Если P1=1, то выполняется обнуление CT1, RG3, T1, RG1 (блок 19) и переход к блоку 21, иначе в CT1 записывается значение выхода SM2. (блок 4). Производится проверка наличия на входной шине множителя (блок 5). При поступлении множителя, знак и мантисса заносятся в RG1, характеристика записывается в RG4 (блок 6). Производится проверка мантиссы на ноль (блок 7). Если P1=1, осуществляется переход к блоку 19, иначе в CT1 заносится значение выхода сумматора SM2 (блок 8). Производится проверка на ПРС (блок 9). Если P2=1, то триггер T2 устанавливаем в единицу (Блок 20) и переходим к блоку 21, иначе происходит проверка на ПМР (блок 10), если P5=1 то осуществляется переход к блоку 19, иначе начинается цикл умножения. Производится проверка младшего разряда регистра множителя RG1 (блок 11). Если P3=0, то осуществляются сдвиги на 1 разряд вправо RG1 и RG3, а также увеличение значения CT2 на единицу (блок 13), иначе в RG3 заносится результат суммы значений регистра множимого и регистра частичных сумм (блок 12) и осуществляется переход к блоку 13. Далее проверяется условие окончания цикла умножения (блок 14). Если P6=0, то осуществляется переход к блоку 11, иначе заканчивается цикл умножения и проверяется условие нормализации мантиссы (блок 15). Если P4=1, то выполняется проверка на ПРС (блок 17), если P2=1 то переход к блоку 20, иначе проверка условия ПМР (блок 18); если P4=0 то выполняется нормализация (блок 16) и переход к блоку 18. Если P5=1, то переход к блоку 19, иначе выполняем переход к блоку 21. Выполняется проверка возможности выдачи результата на выходную шину (блок 21) и выдача результата на выходную шину (блок 22).

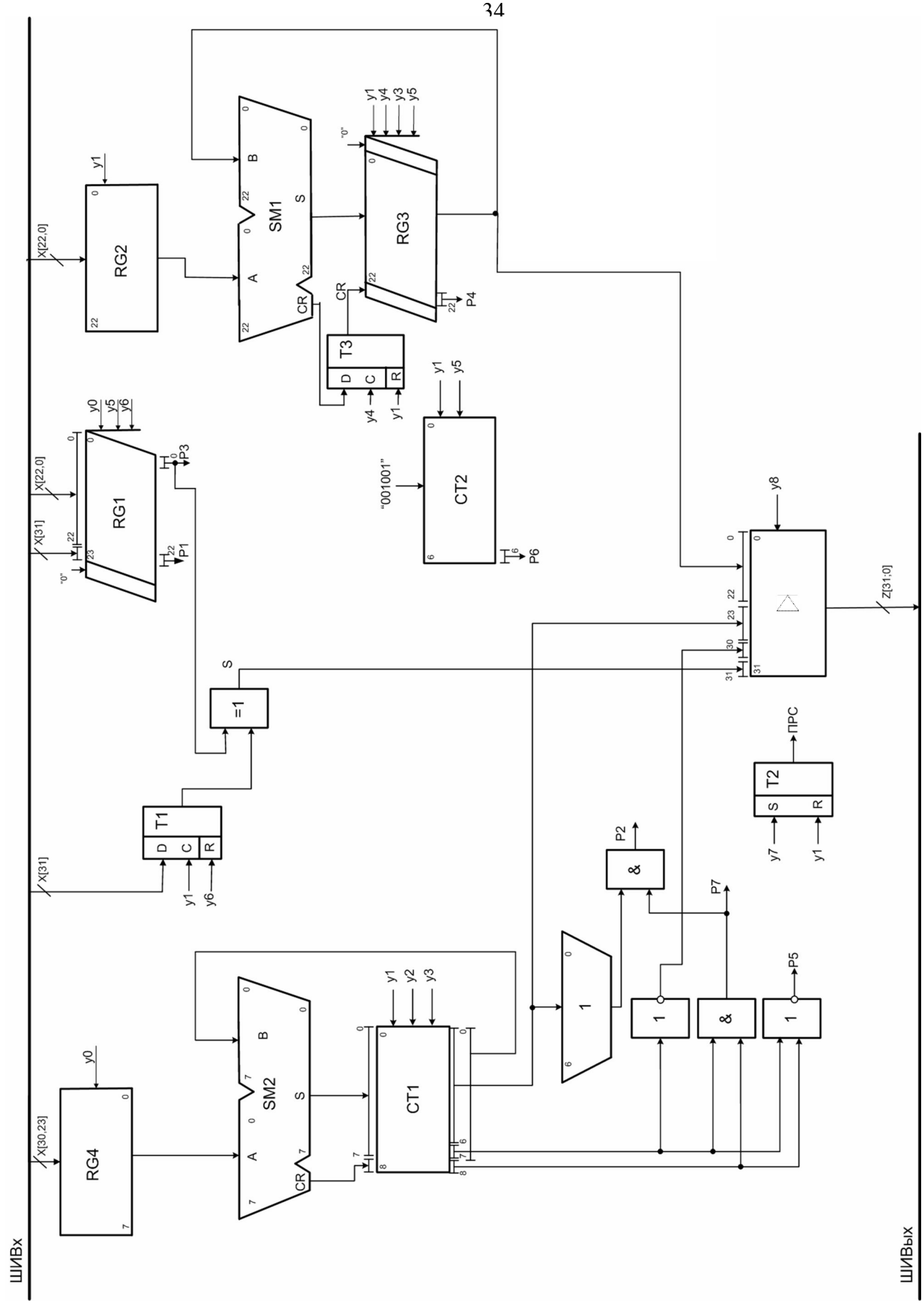


Рис. 19 – ФС операционного автомата

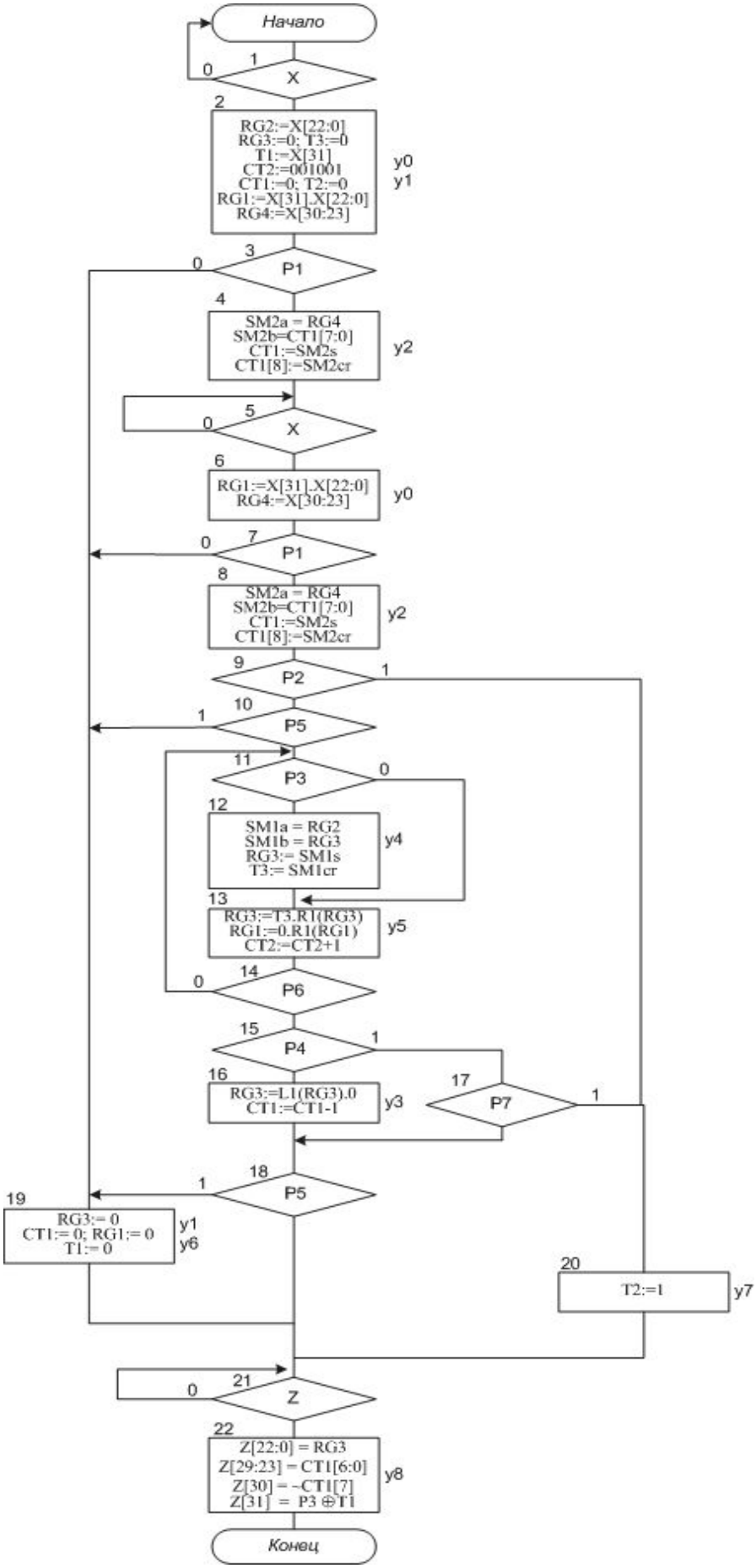


Рис. 20. ГСА алгоритма умножения

2.5 Построение отмеченной граф-схемы алгоритма

Для разметки граф-схемы алгоритма каждой совокупности микроопераций, находящихся в операторных вершинах, ставятся в соответствие управляющие микрокоманды (МК) $Y_1 \ldots Y_n$. Эти МК являются выходными сигналами УА и обеспечивают выполнение требуемых действий в соответствии со списком микроопераций (МО) ОА. Совокупность МО для каждой операторной вершины образуют микрокоманды, список которых представлен в таблице 4. Каждой условной вершине содержательной ГСА ставится в соответствие один из входных сигналов управляющего автомата $X_1 \ldots X_m$ [1].

Таблица 4 – Список микрокоманд

МК	Совокупность МО
Y0	y_0, y_1
Y1	y_2
Y2	y_0
Y3	y_2
Y4	y_4
Y5	y_5
Y6	y_3
Y7	y_1,y_6
Y8	y_7
Y9	y_8

Далее в полном соответствии с правилами разметки содержательной ГСА (см. ниже) строится отмеченная ГСА.

Предварительно в каждой условной вершине проставляются символы из множества входных сигналов УА – X_1, X_2, …, X_M (таблица 5). Во всех операторных вершинах ГСА проставляют символы из множества выходных сигналов УА – $У_1$, $У_2$, …, $У_N$ (таблица 5.1). Удобно в каждой операторной

вершине ГСА вслед за символом МК указать в скобках набор МО, образующих каждую МК.

Таблица 5 – Список входных сигналов для УА

Входной сигнал УА	Логическое условие ОА (осведомительные сигналы)
X_1	X
X_2	P_1
X_3	P_2
X_4	P_5
X_5	P_3
X_6	P_6
X_7	P_4
X_8	P_7
X_9	Z

Разметка ГСА в соответствии с моделью Мили, выполняется по следующим правилам:

1. Вход вершины, следующей за начальной, и вход конечной вершины отмечаются символом начального состояния автомата a_0.

2. Входы всех вершин, следующих за операторными, отмечаются символами a_1, …, a_K.

3. Если вход вершины отмечается, то только одним символом.

4. Входы различных вершин за исключением конечной, отмечаются различными символами.

Разметка ГСА в соответствии с моделью Мура, выполняется по следующим правилам:

1. Символом начального состояния автомата a_0 отмечаются начальная и конечная вершины.

2. Различные операторные вершины отмечаются различными символами a_1, …, a_K.

Рис 21

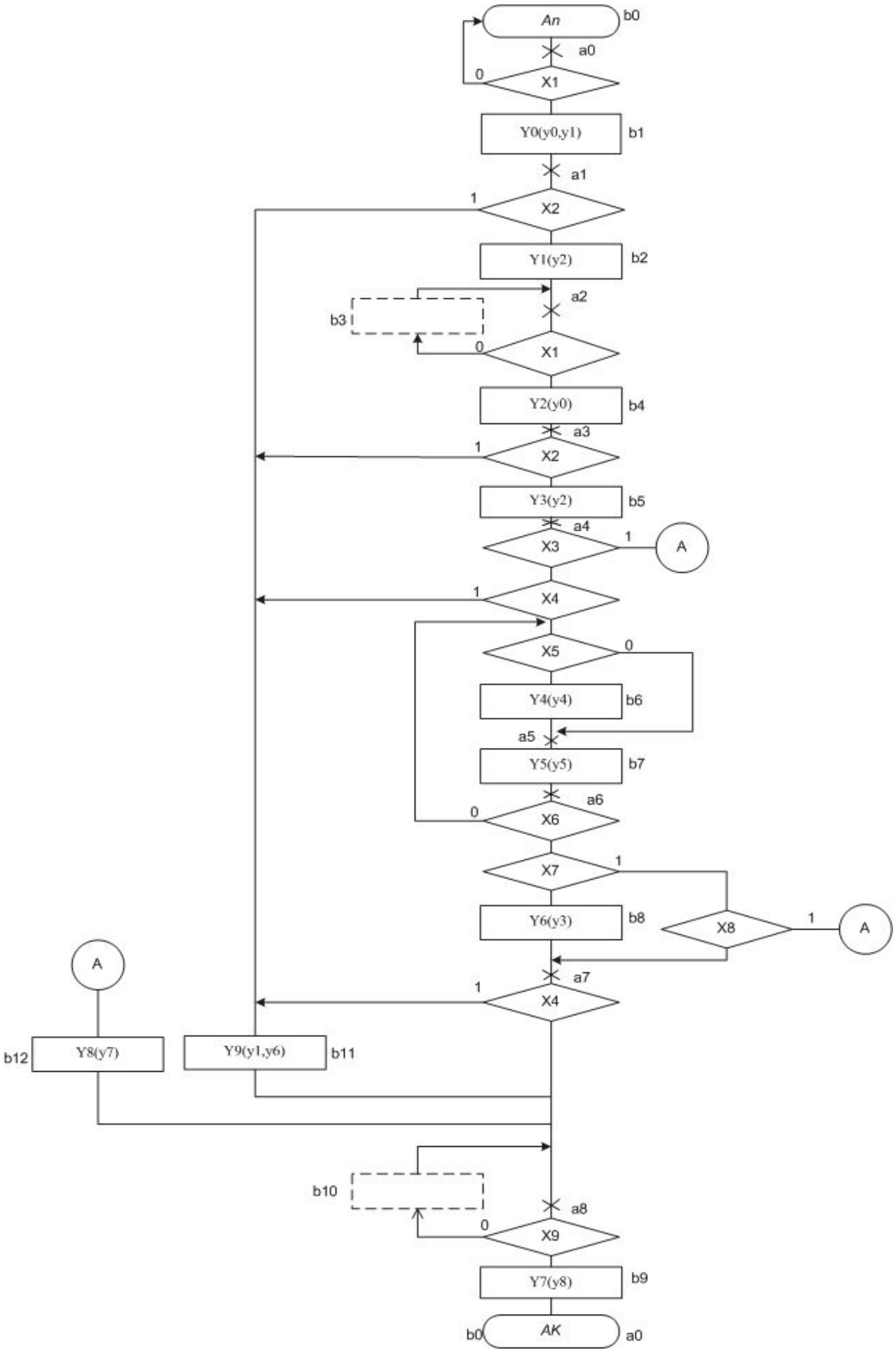


Рис. 21
Отмеченная ГСА для алгоритма умножения

Все операторные вершины должны быть отмечены, то есть каждой МК, отдельно представленной в ГСА ставится в соответствие отдельное состояние автомата Мура.

В логических вершинах ГСА, реализующих режим ожидания, существует возвратная дуга, когда один из выходов вершины подан на ее вход. На этой дуге необходимо вводить дополнительное фиктивное состояние автомата Мура.

Получается ГСА, размеченная для модели Мили символами $a_0..a_8$, для модели Мура символами $b_0..b_{13}$.

2.6 Построение графа автомата

На основе отмеченной ГСА построим граф автомата Мили (рис. 22). Граф автомата Мили имеет 9 вершин, соответствующие состояниям автомата a_0, $a_1,...,a_8$.

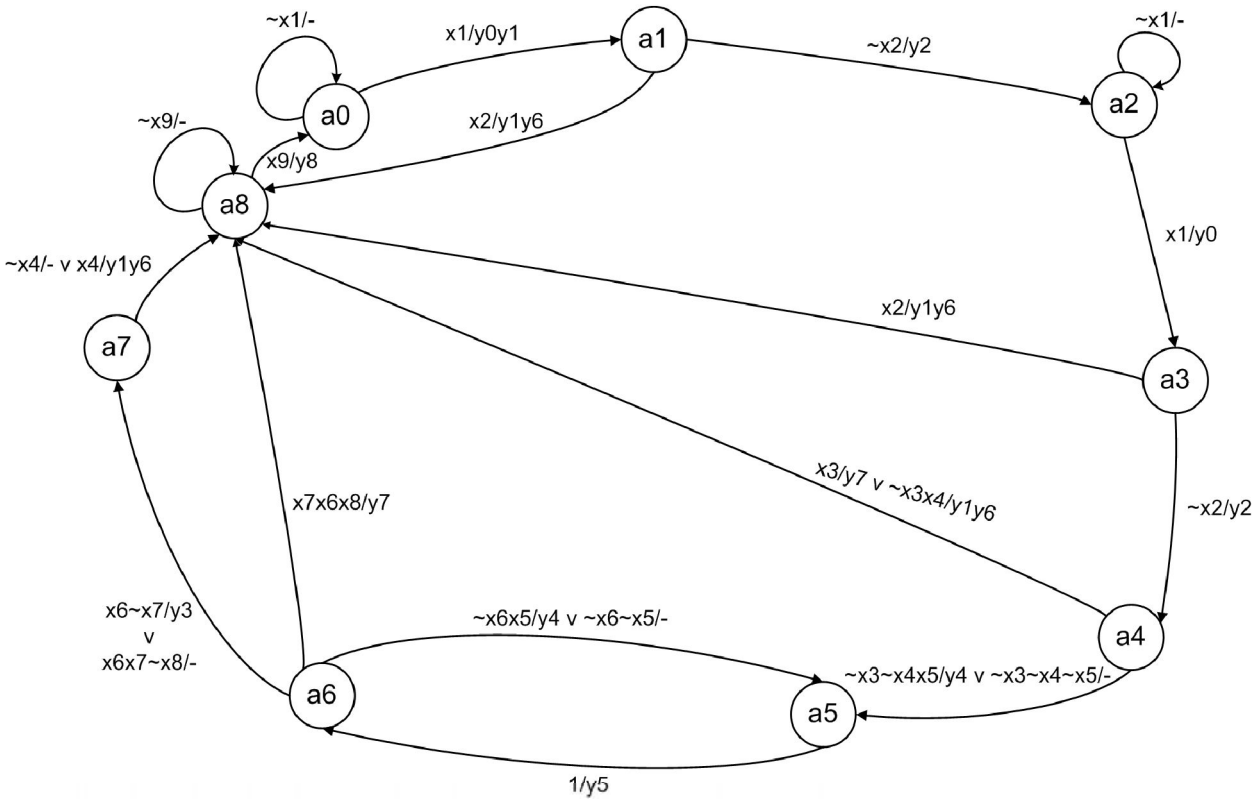


Рис. 22. Граф автомата Мили

Дуги графа отмечены входными сигналами, действующими на каждом переходе (числитель), и набором выходных сигналов, вырабатываемых УА на данном переходе (знаменатель).

2.7 Кодирование состояний автомата, выбор элементов памяти

Основываясь на том виде, который принимает граф Мили (большое количество последовательных переходов и незначительное число нестандартных), можно сделать вывод, что при использовании соседнего кодирования, счетчика и дешифратора есть вероятность построения варианта УУ, близкого к оптимальному. Но для более точной оценки необходим более детальный анализ. А именно, сравнение схем УУ, построенных на D-триггерах с дешифратором, на RS-триггерах с дешифратором с использованием соседнего кодирования, на счетчике с дешифратором, а также на сдвиговом регистре с использованием унитарного кодирования.

Реализуем управляющий автомат для основной операции на счетчике с дешифратором.

Для кодирования состояний автомата на счётчике необходимо, чтобы разность кодов между соседними состояниями составляла единицу. Данная кодировка представлена в таблице 7.

Таблица 7 – Коды внутренних состояний УА

a0	a1	a2	a3	a4	a5	a6	a7	a8
0001	0010	0011	0100	0101	0110	0111	1000	0000

Составляем прямую структурную таблицу переходов и выходов автомата Мили (таблица 8) и формируем логические выражения для функций возбуждения, которые получают по таблице как конъюнкции соответствующих исходных состояний a_m и входных сигналов, которые объединены знаками дизъюнкции для всех строк, содержащих данную функцию возбуждения.

Таблица 8 – Прямая структурная таблица переходов и выходов автомата Мили.

Исходное состояние a_m	Код a_m	Состояние перехода a_s	Код a_s	Входной сигнал $X(a_m..a_s)$	Выходной сигнал $Y(a_m..a_s)$	Функция возбуждения счетчиков
1	2	3	4	5	6	7
a0	0001	a0	0001	~x1	-	-
		a1	0010	x1	y0y1	+1
a1	0010	a2	0011	~x2	y2	+1
		a8	0000	x2	y1y6	R
a2	0011	a2	0011	~x1	-	-
		a3	0100	x1	y0	+1
a3	0100	a4	0101	~x2	y2	+1
		a8	0000	x2	y1y6	R
a4	0101	a5	0110	~x3~x4x5	y4	+1
		a5	0110	~x3~x4~x5	-	+1
		a8	0000	x3	y7	R
		a8	0000	~x3x4	y1y6	R
a5	0110	a6	0111	1	y5	+1
a6	0111	a5	0110	~x6x5	y4	-1
		a5	0110	~x6~x5	-	-1
		a7	1000	x6~x7	y3	+1
		a7	1000	x6x7~x8	-	+1
		a8	0000	x7x6x8	y7	R
a7	1000	a8	0000	~x4	-	R
		a8	0000	x4	y1y6	R
a8	0000	a8	0000	~x9	-	-
		a0	0001	x9	y8	+1

Получение логических выражений для функций возбуждения счетчика

+1 = a0x1 v a1~x2 v a2x1 v a3~x2 v a4~x3~x4 v a5 v a6x6~x7 v a6x6x7~x8 v
v a8x9

-1 = a6~x6

R = a1x2 v a3x2 v a4x3 v a4~x3x4 v a6x6x7x8 v a7

y0 = a0x1 v a2x1

y1 = a0x1 v a1x2 v a3x2 v a4~x3x4 v a7x4

y2 = a1~x2 v a3~x2

y3 = a6x6~x7

y4 = a4~x3~x4x5 v a6~x6x5

y5 = a5

y6 = a1x2 v a3x2 v a4~x3x4 v a7x4

y7 = a4x3 v a6x7x6x8

y8 = a8x9

После упрощения получим:

+1 = y0 v a5 v y8 v y3 v y2 v r~x4 v qx7~x8

-1 = m

R = y6 v y7 v a7~x4

y0 = (a0 v a2)x1

y1 = a0x1 v y6

y2 = h~x2

y3 = q~x7

y4 = (r~x4 v m)x5

y5 = a5

y6 = hx2 v (r v a7)x4

y7 = (a4 v qx7)x8

y8 = a8x9

h = a1 v a3

m = a6~x6

r = a4~x3

q=a6x6

По данным логическим функциям видно, что запись в счетчик во время работы УУ не производится (она нужна только в момент сброса УУ до начала его работы). Следовательно, в процессе работы УУ используются только счетные входы и вход сброса. Логическую функцию для счетного входа «+1» можно еще упростить. Если в определенный момент времени формирования следующего внутреннего состояния УУ поступает сигнал на R, то в этот же момент времени информация не должна повлиять на срабатывание счетных входов. Если используется счетчик с дополнительным входом разрешения счета и счетный вход работает по принципу «1» - счетчик работает на сложение, «0» - счетчик работает на вычитание, то логическая функция для входа разрешения счета E = ~R. В тот момент времени, когда информация влияет на срабатывание счетных входов, необходимо разграничить две ситуации: +1 и -1. Поскольку «+1» не используется в момент, когда срабатывает «-1», то для «+1» логическая функция будет равна ~m.

В результате получим:

+1 = ~m

-1 = m

R = y6 v y7 v a7~x4

Функционально-логическая схема управляющего автомата представлена на рисунке 23.

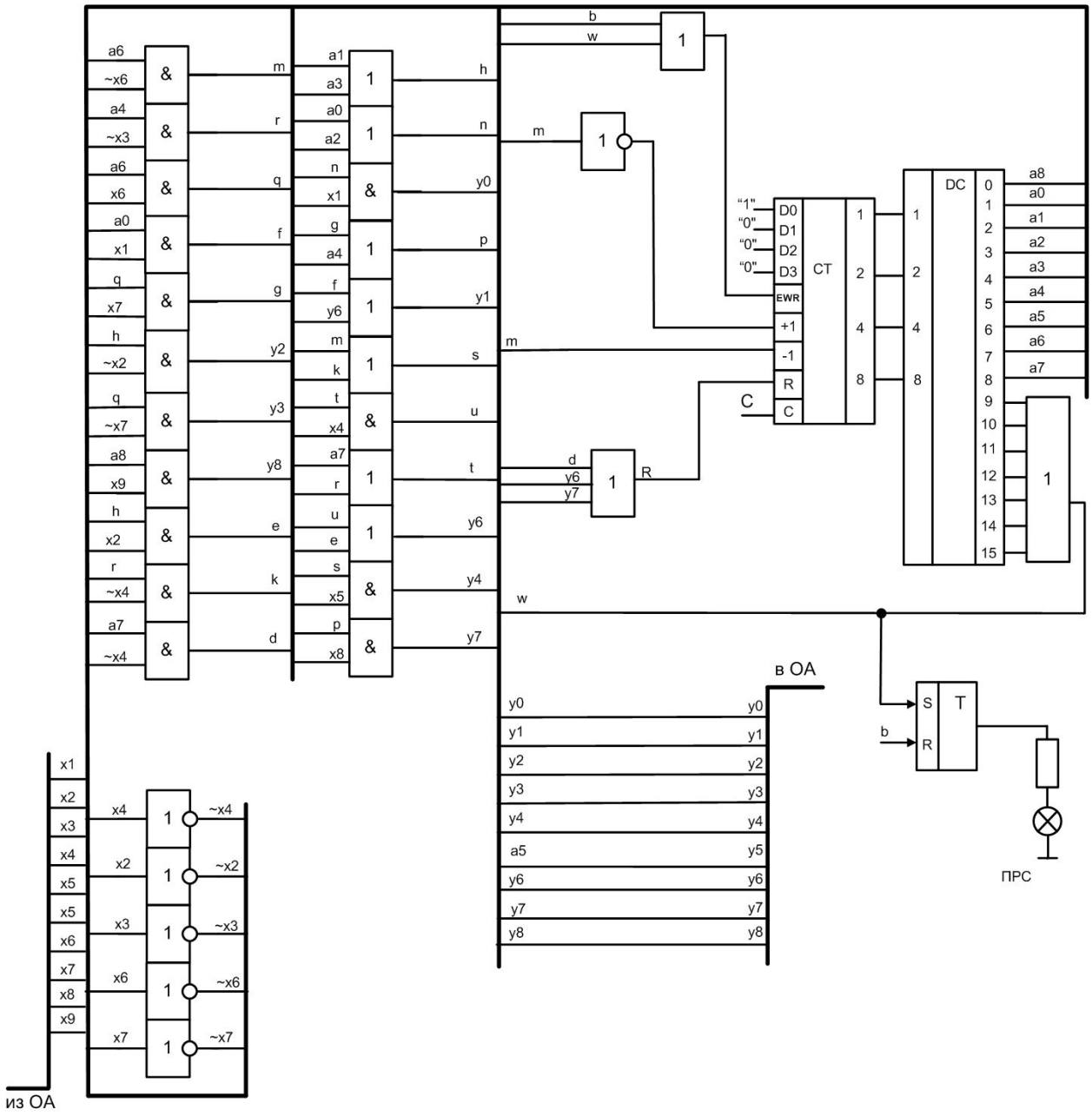


Рис 23 – Схема УА

3. РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ СХЕМЫ ОПЕРАЦИОННОГО АВТОМАТА В САПР QUARTUS

Рассмотрим построение схемы ОА на примере функциональной схемы умножения двоичных **16-тиразрядных** чисел первым способом в ПК с плавающей запятой, с характеристиками.

Создайте проект в Quartus следующим образом. Запустите Quartus II Web Edition. Выберите пункт меню **`File>New Project Wizard`**. Нажмите **Next**. В следующем окне (рисунок 24) выберите директорию, в которой будет находиться Ваш проект, и задайте имя проекту. Нажмите **Next**.

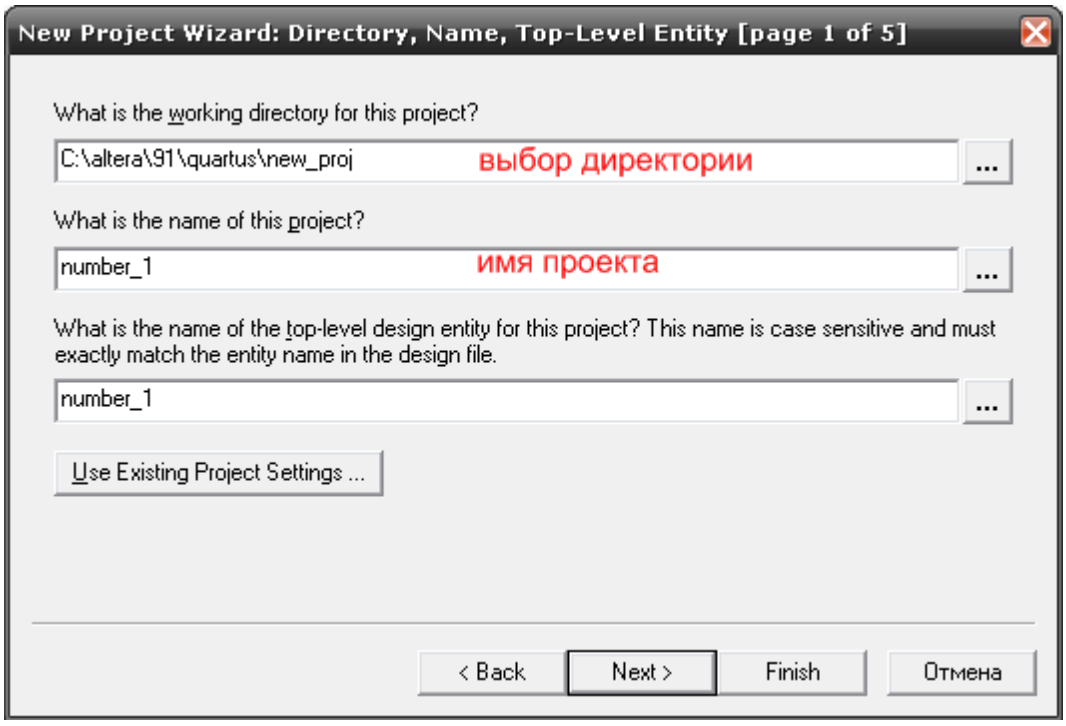


Рис. 24 Выбор директории и задание имени проекту

В появившемся окне (рисунок 25) можно включить в свой новый проект файлы из других проектов. Нажмите **Next**.

В графе Family (рисунок 26) выберите Cyclone III, а в таблице Available devices укажите EP3C5E144C8. Нажмите 2 раза **Next** и **Finish**.

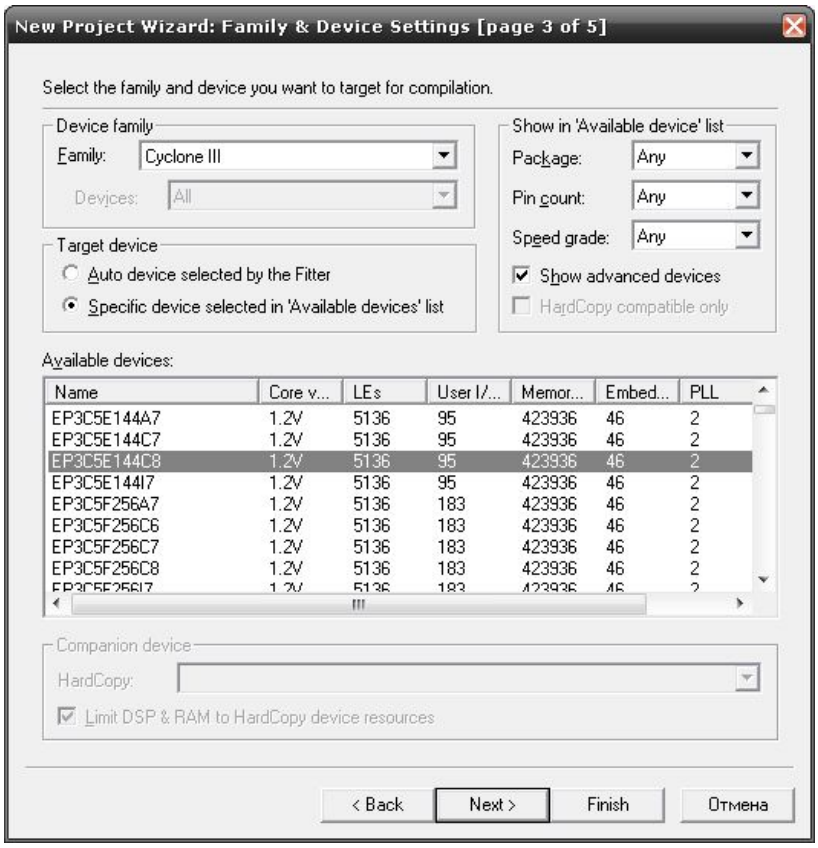


Рис. 25. Включение в проект файлы из других проектов

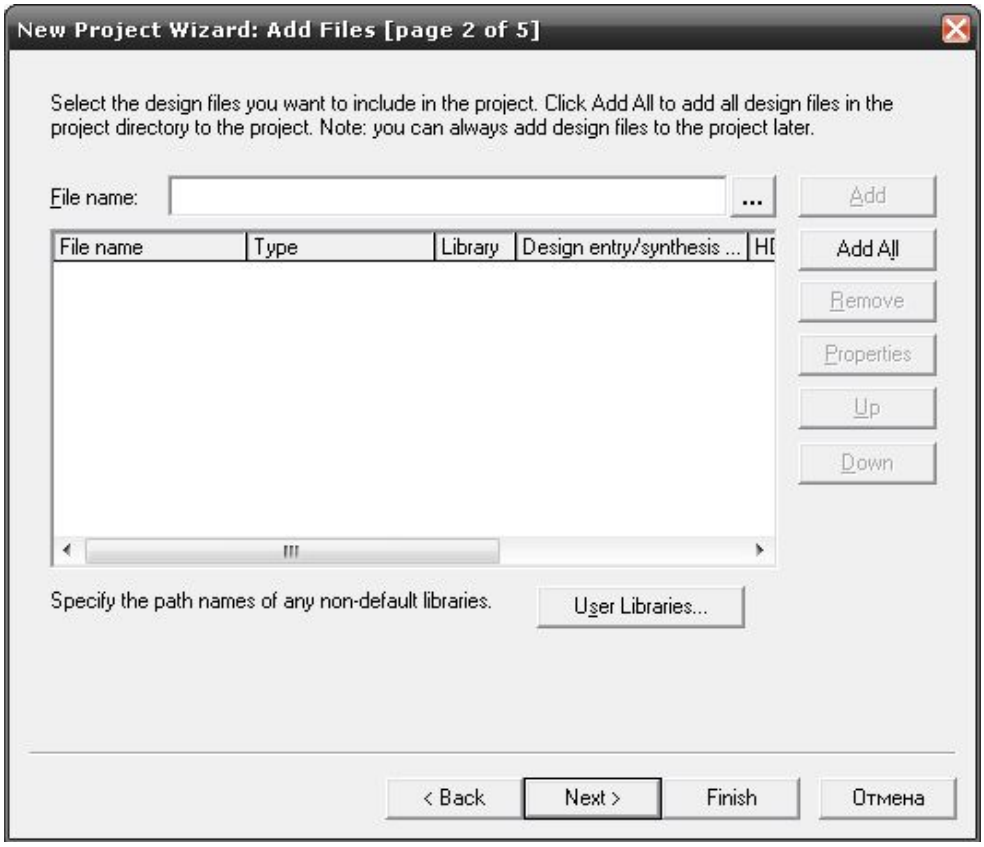


Рис. 26. Выбор устройства

Выберите в меню **File>New>Block Diagram/Schematic File**. Нажмите ОК. Перед Вами появится окно «в точечку», где и будет располагаться Ваша схема.

3.1 Построение блока входных данных

В первую очередь необходимо на свободном пространстве схемы *.bdf по двойному щелчку мыши вызвать окно библиотеки символов **Symbol** (рисунок 27). Эти символы представляют собой либо уже конкретную схему, либо ее отдельные элементы (некоторые из них можно создать из готового шаблона мегафункций).

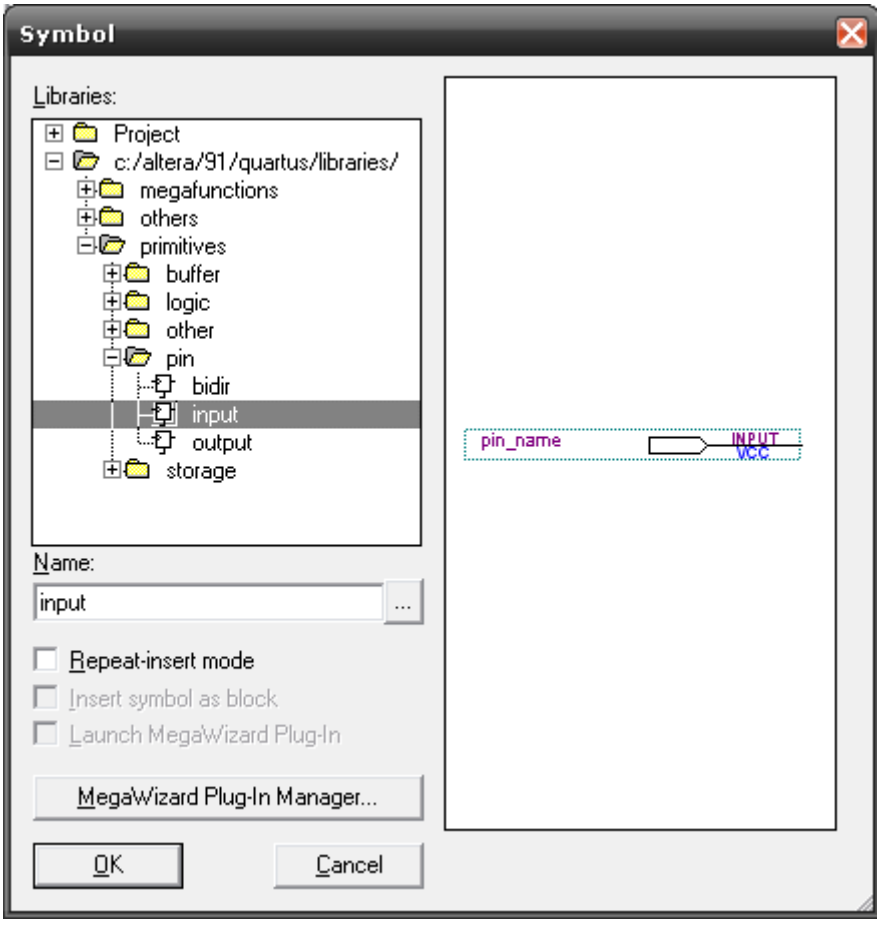


Рис. 27. – Окно стандартных библиотек проекта

В поле Name наберите «input». Программа сразу найдет в списке нужный элемент. Нажмите ОК, и на схеме появится входная шина. Щелкните по pin_name дважды или вызовите из контекстного меню команду **Properties** и укажите

название шины и ее разрядность. В нашем случае это будет “x[15..0]”. Выберите слева, на панели инструментов, компонент **Orthogonal Bus Tool** (рисунок 28) и протяните линию от выхода элемента Input и назовите ее соответственно “x[15..0]” (*разрядности в имени input/output и названии соединенной с ними шин должны совпадать*!).

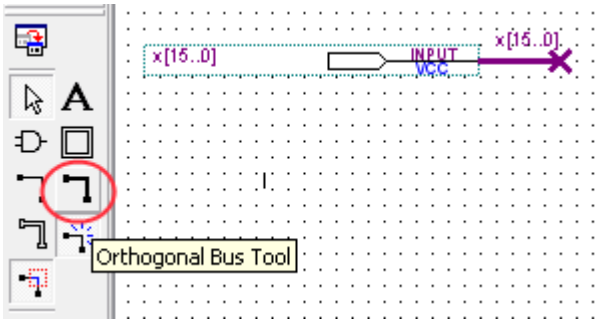


Рис. 28. Инструмент Orthogonal Bus Tool

Далее необходимо создать регистры для хранения множимого и множителя, их характеристик, а также триггер для хранения знака множимого. В соответствии с первым способом умножения регистр множителя должен быть сдвиговым (сдвиг вправо) n-разрядным, регистр для хранения множимого также n-разрядный и регистр для хранения характеристик обоих операндов. Начнем со сдвигового регистра множителя (сдвиговый регистр сумм частичных произведений строится аналогично). Для этого снова, по двойному щелчку мыши в свободном пространстве схемы, вызовите библиотеку символов. Нажмите **MegaWizard Plug-In Manager**. Нажмите **Next**. Выберите раздел *Storage/LPM_SHIFTREG* и задайте в поле What name do you want for the output file? имя нового символа (рисунок 29). В данном случае это будет *rgshift*.

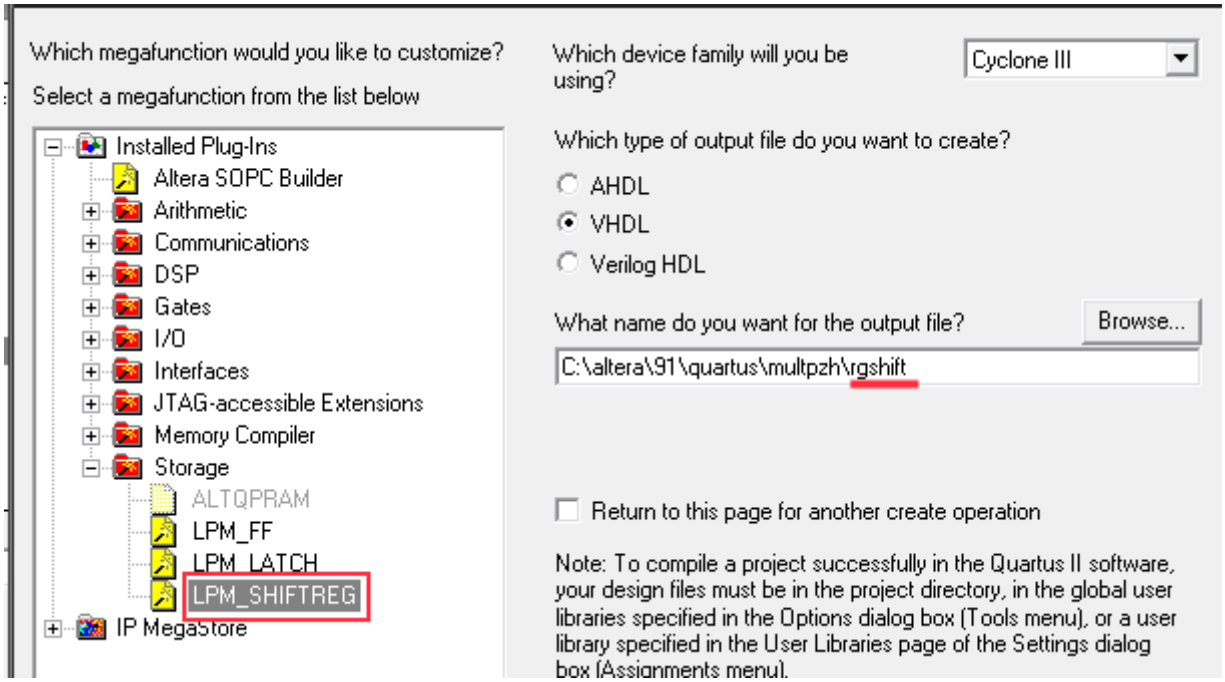


Рис. 29. MegaWizard Plug-In Manager

Разрядность данного регистра множителя будет равна одиннадцати (т.к. в программе для проверки операционной части входные данные представлены следующим образом: 11 разрядов выделено под мантиссу со знаком, 5 разрядов для характеристики). После цикла умножения (10 тактов) знак множителя будет в младшем разряде регистра и его можно использовать для определения знака результата. Следовательно, его можно считать позже для формирования знака результата без использования дополнительного триггера. Укажите параметры как на рисунке 30.

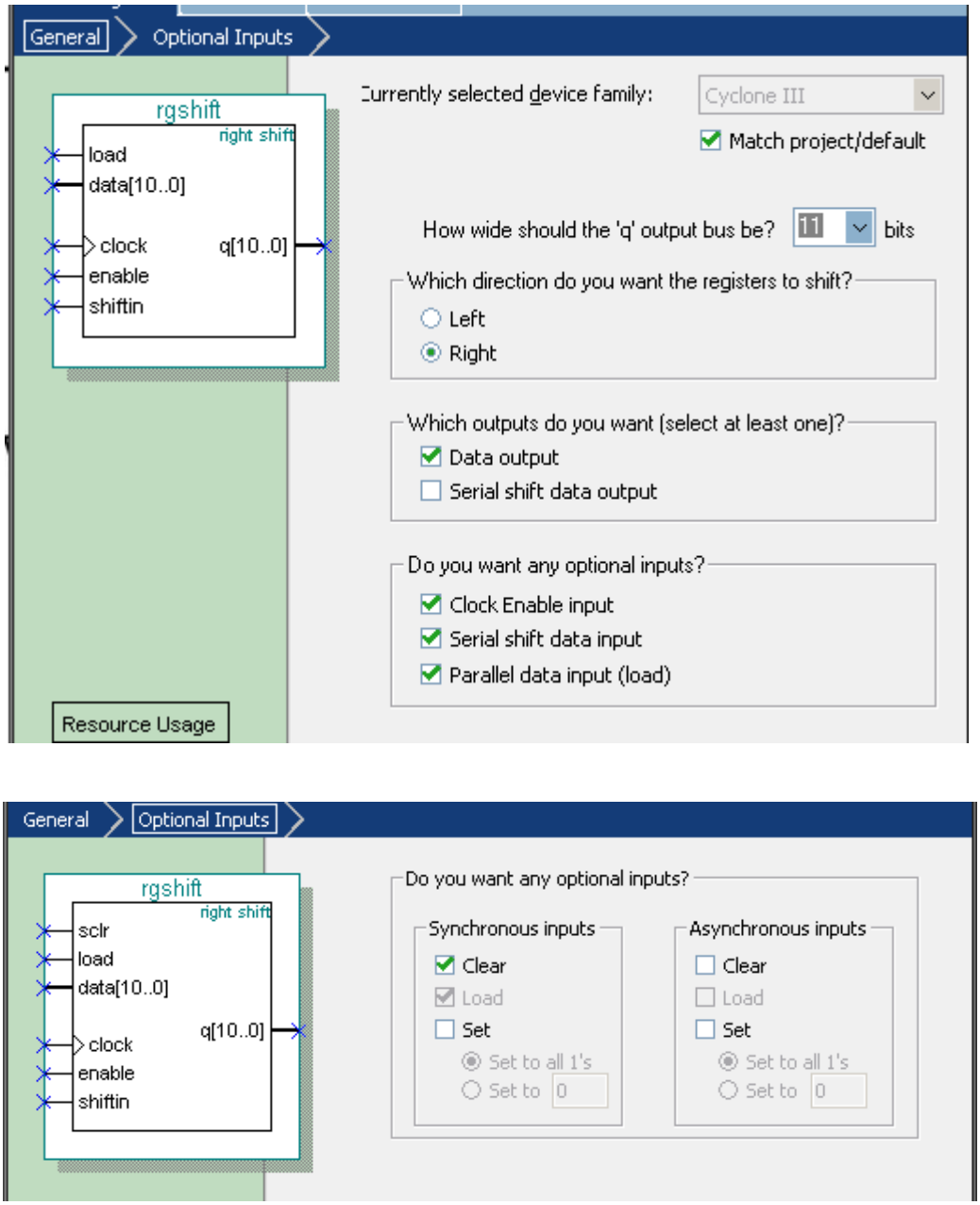


Рис. 30. Параметры сдвигового регистра

Далее нажмите **Next, Next, Finish**. Теперь Вы можете воспользоваться этим элементом когда угодно, т.к. он находится в библиотеке символов в каталоге Вашего проекта **Project**. На вход *load* необходимо подавать сигнал записи по положительному фронту сигнала. Вход *shiftin* – информационный вход старшего разряда. Для данного примера это будет «0», поэтому этот вход необходимо заземлить (в библиотеке символов выберите *gnd* и соедините его с помощью

Orthogonal Node Tool в свою схему). Если на вход *load* не поступает сигнал записи, то регистр по стробирующему импульсу будет сдвигать данные на 1 разряд вправо.

Для того, чтобы избежать непредусмотренных сдвигов, необходимо использовать дополнительный вход *enable*, который отвечает за разрешение реакции на стробирующий сигнал. На вход *data[10..0]* необходимо подавать данные с входной шины. Для этого выберите **Orthogonal Bus Tool**, проведите линию шины и соедините ее со входом *data[10..0]*, после чего укажите название данной шины. Особенность построения связей между входами и выходами элементов в том, что ***если задать одинаковые имена на выходе одного элемента и на входе другого, эти элементы считаются соединенными!*** В данном примере необходимо указать в имени входной шины данных регистра множителя сначала «откуда взять» знак операнда, а затем, через запятую, «откуда взять» мантиссу. Выход *q[10..0]* назовите *rg1[10..0]* (рисунок 31). Вход *sclr* – синхронный сброс регистра.

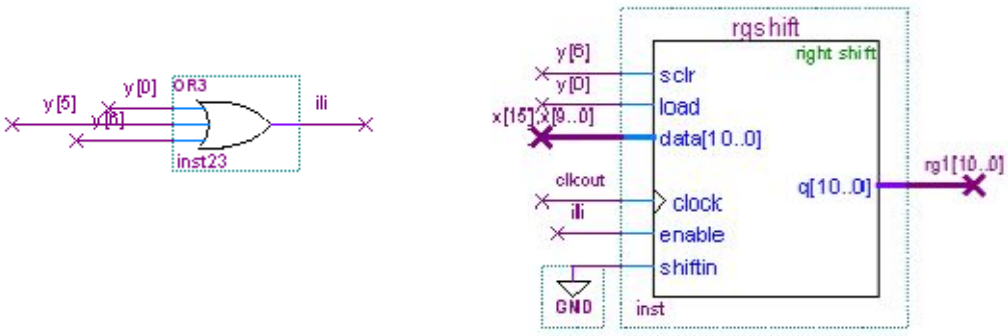


Рис. 31 Изображение модели сдвигового регистра в Quartus

Создадим регистры для хранения мантиссы и характеристики множимого (рисунок 32). **`MegaWizard Plug-In Manager>Next`**. Из раздела Storage выберите *LPM_FF* и задайте в поле What name do you want for the output file? имя символа. В данном случае это будет *rg10* (мантисса множимого) и *rg5* (характеристики). В How many flipflops do you want? укажите разрядность регистров (10 для мантиссы множимого и 5 для характеристики). Поставьте

галочку в разделе Create a Clock Enable input. Далее 3 раза нажмите **Next** и затем **Finish**. Входы *data* обозначьте в соответствии с тем, какие данные должны заноситься в эти регистры (для характеристики – x[14..10], для мантиссы множимого – x[9..0]).

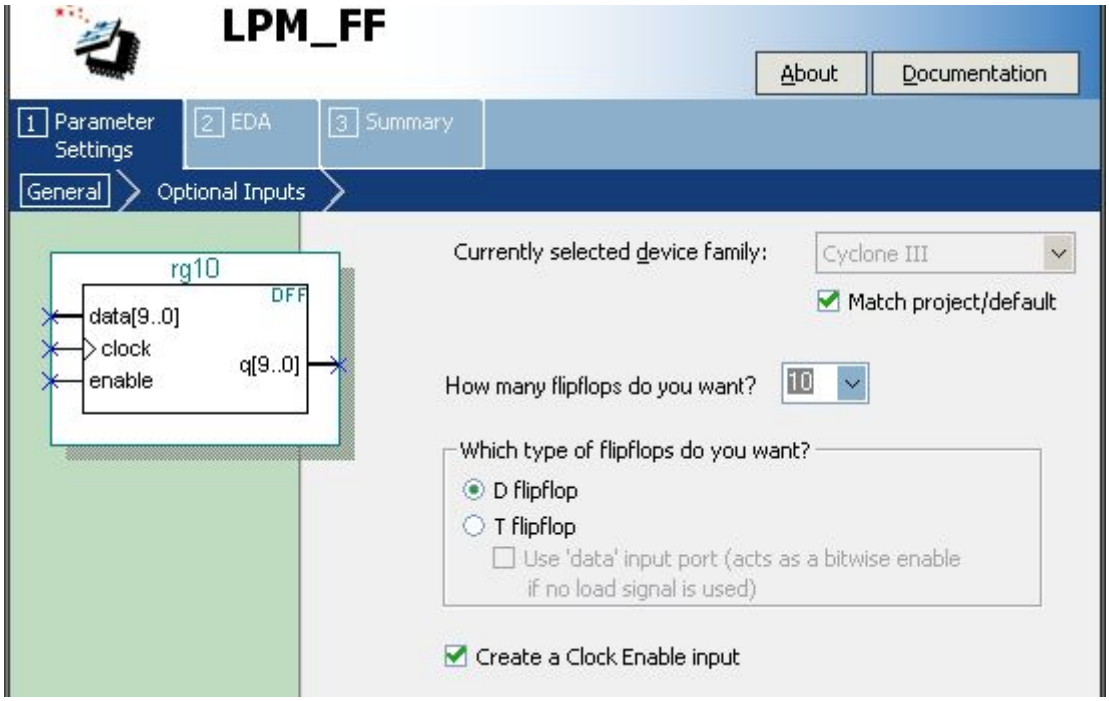


Рис. 32. Окно построения регистров хранения

Создадим триггер для хранения знака множимого (рисунок 33). Двойной щелчок мыши по пустому полю схемы. Укажите в поле Name «dff» (D – триггер). Нажмите ОК. Вход *PRN* триггера – установка в единицу, вход *CLRN* – сброс триггера. На информационный вход триггера подайте x[15], а выход триггера соедините с одним из входов логического элемента *XOR* (в библиотеке символов наберите в поле Name «xor»). На второй вход логического элемента XOR подайте значение знакового разряда мантиссы множителя из регистра *rg1* (rg1[0]), который по своей сути также является осведомительным сигналом p[3]). Выход логического элемента *XOR* подпишите как «sign». Это будет знаком результата умножения.

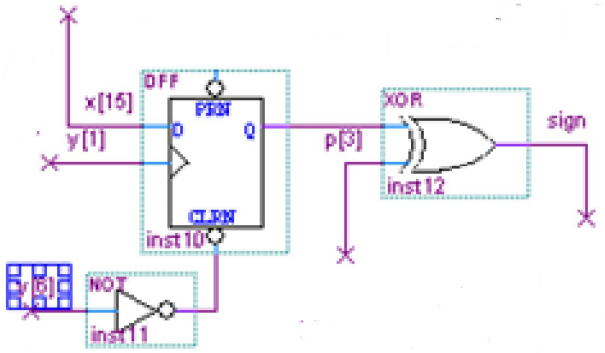


Рис. 33. Триггер для хранения знака

3.2 Построение блока выполнения операции

Создадим сумматоры для формирования сумм частичных произведений и для формирования характеристики результата. Нажмите **`MegaWizard Plug-In Manager>Next`**. Слева раскройте раздел Arithmetic и выберите *LPM_ADD_SUB*. Задайте имя в поле What name do you want for the output file? (пусть это будет *sm1* в блоке обработки мантисс и *sm2* в блоке обработки характеристик). Разрядность сумматора в блоке обработки мантисс совпадает с разрядностью регистра хранения мантиссы множимого. Укажите параметры как на рисунке 34.

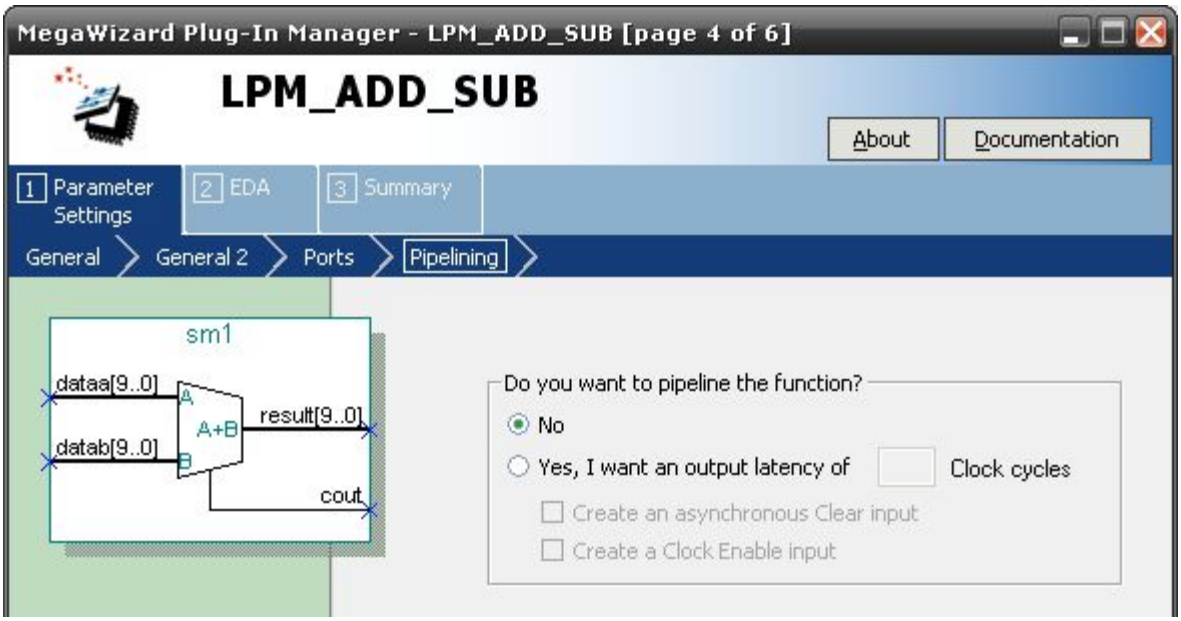


Рис. 34. Сумматор

Схема проверки операндов на равенство нулю. **MegaWizard Plug-In Manager>Next**. Раскройте слева раздел Gates. Выберите *LPM_OR*. Задайте имя (*ili10*). Укажите параметры как на рисунке 35.

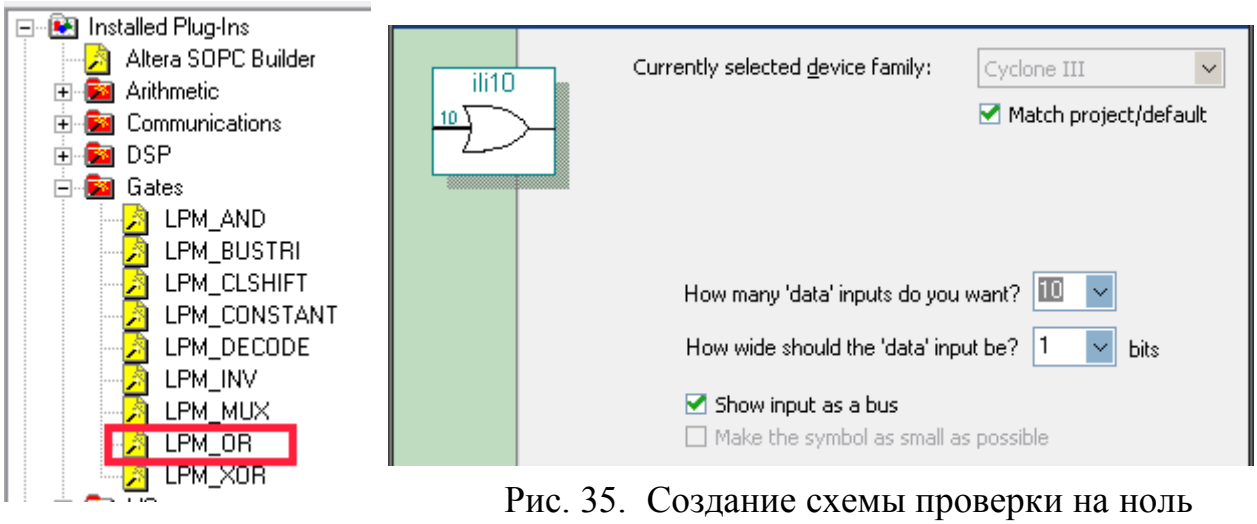
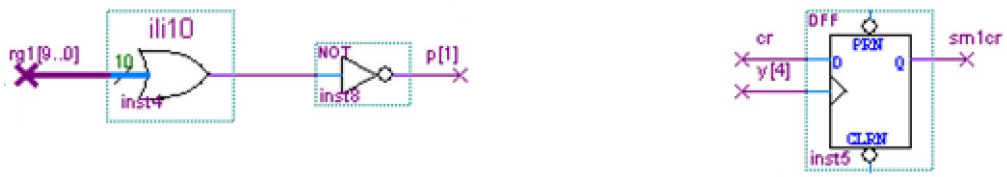


Рис. 35. Создание схемы проверки на ноль

Два раза нажмите **Next**, затем **Finish**. Добавьте к схеме элемент *not* из библиотеки символов, чтобы схема срабатывала по сигналу “1” (рисунок 36а). Для сохранения значения выхода сумматора *CR* формирования сумм частичных произведений необходим D – триггер (рисунок 36б).



а) б)

Рис. 36. Схема проверки на ноль (а) и D-триггер (б)

Сформируем счетчик циклов. **MegaWizard Plug-In Manager>Next**. Раскройте слева раздел Arithmetic. Выберите элемент *LPM_COUNTER*. Задайте имя. Нажмите **Next**. Укажите параметры как на рисунке 37. Нажмите **Next>Next>Finish**.

Для подачи на некоторые входы счётчика “1”, а на другие “0” необходимо ввести схемы, выдающие данные сигналы. Создайте элементы *VCC* (логическая 1) и *GND* (логический 0) из библиотеки символов и соедините их выходы с шинами. Задайте им имена (рисунок 38).

На информационные входы (*data[4..0]*) счетчика (рисунок 39) необходимо подать комбинацию «00110» (*gr[9..8],v[1..0],gr[7]*).

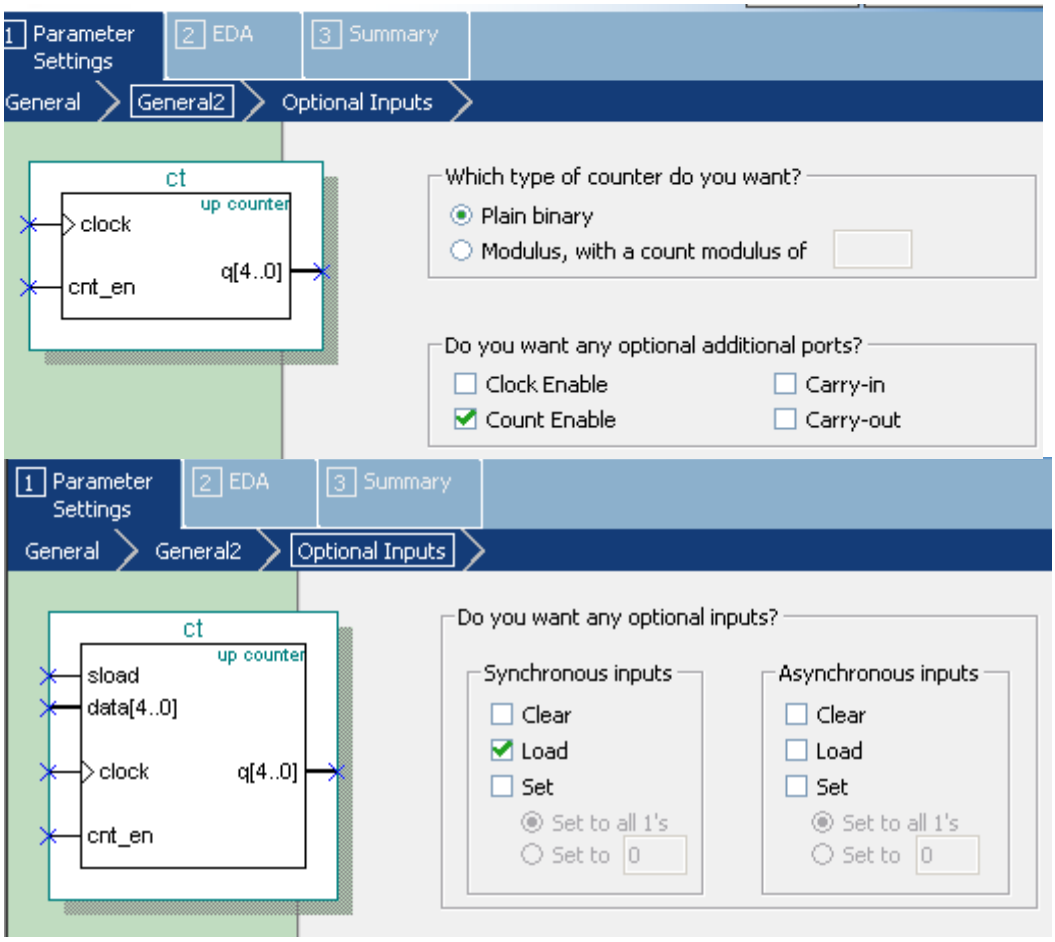


Рис. 37. Параметры счетчика циклов

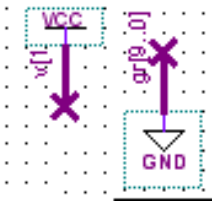


Рис. 38. Логический ноль и логическая единица

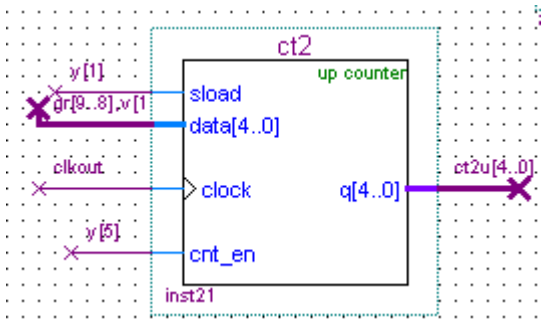


Рис. 39. Схема модели счетчика в Quartus

Еще один вариант реализации счетчика циклов. Можно воспользоваться четырехразрядным счетчиком, и начинать считать не с нуля, а с 14 (1110) и далее до восьми (1000). Таким образом нет необходимости в наращивании разрядности счетчика. Но данный метод требует доработку схемы, что позволило бы после первой итерации цикла умножения игнорировать значение старшего разряда счетчика, чтобы это не повлияло на ранний выход из цикла реализации процедуры умножения.

Построим комбинационную схему для определения ПРС и ПМР (рисунок 40).

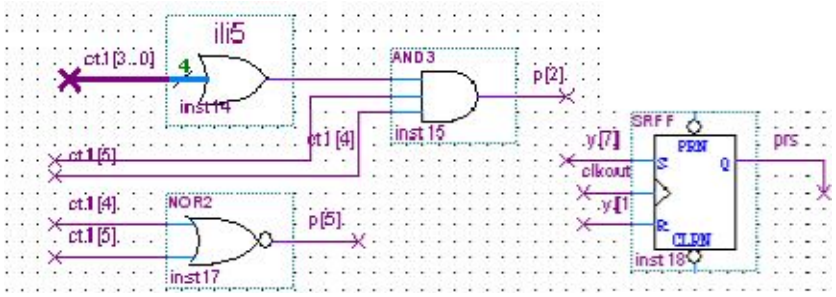


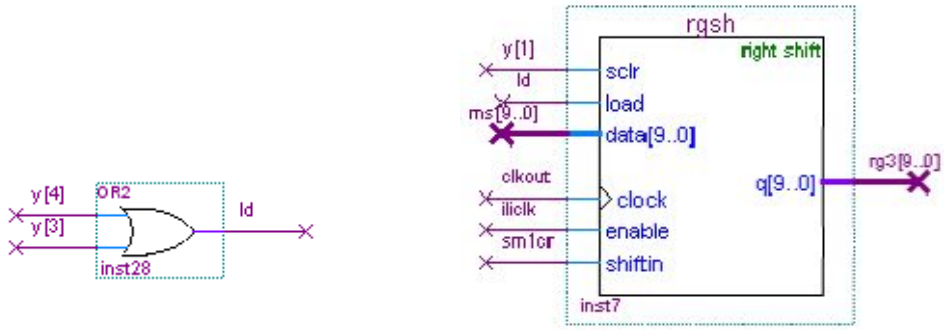
Рис. 40. Схема определения признаков ПРС и ПМР

Ct1[5..0] – информационный выход счетчика для формирования характеристик. Подаются на вход схемы ili5 (без инверсии выхода, строится аналогично схеме проверке на ноль в блоке входных данных). *And3*, *srff* (RS – триггер для формирования признака ПРС), *nor2* (двухвходовое ИЛИ-НЕ) берутся

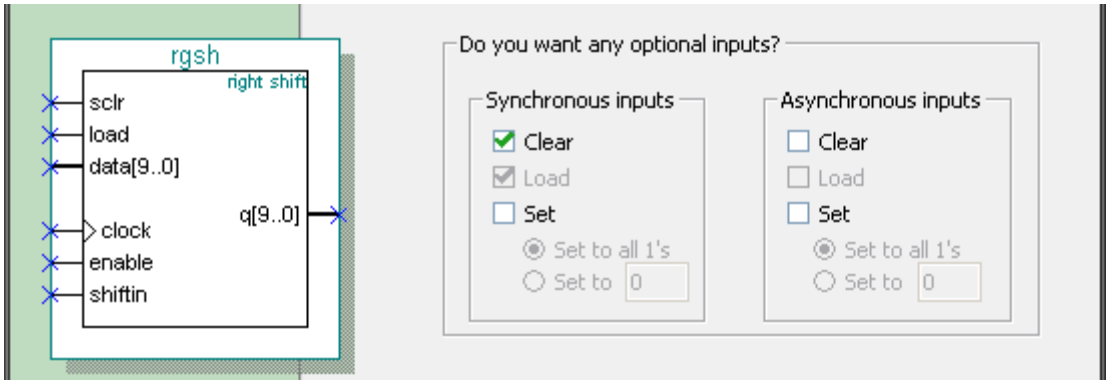
из библиотеки символов. P[2], p[5] – осведомительные сигналы возникновения ситуации ПРС и ПМР соответственно.

3.3 Блок выдачи результата

Сдвиговый регистр суммы частичных произведений формируется аналогично регистру мантиссы множителя (рисунок 41).



а)



б)

Рис. 41. Регистр произведений схема в Quartus (а) и свойства в окне MegaWizard Plug-In Manager (б)

В результате может возникнуть ненормализованная мантисса, и для нормализации необходим сдвиг влево. Поэтому следует использовать схему логического сдвига с последующей записью в регистр *rgsh*.

Поскольку в Quartus нет возможности синтеза реверсивного регистра, необходимо осуществить сдвиг влево с помощью схемы логического сдвига, с последующей записью результата в регистр хранения сумм частичных произведений. Т.к. на вход данного регистра должны также поступать данные с

сумматора *sm1*, в схему необходимо включить двухплечевой мультиплексор, на первое плечо которого подаются данные с сумматора, а на второе - результат сдвига влево содержимого регистра.

Для создания мультиплексора нажмите **MegaWizard Plug-In Manager>Next**. В разделе Gates укажите *LPM_MUX*. Задайте имя (*ms2*) и укажите параметры как на рисунке 42.

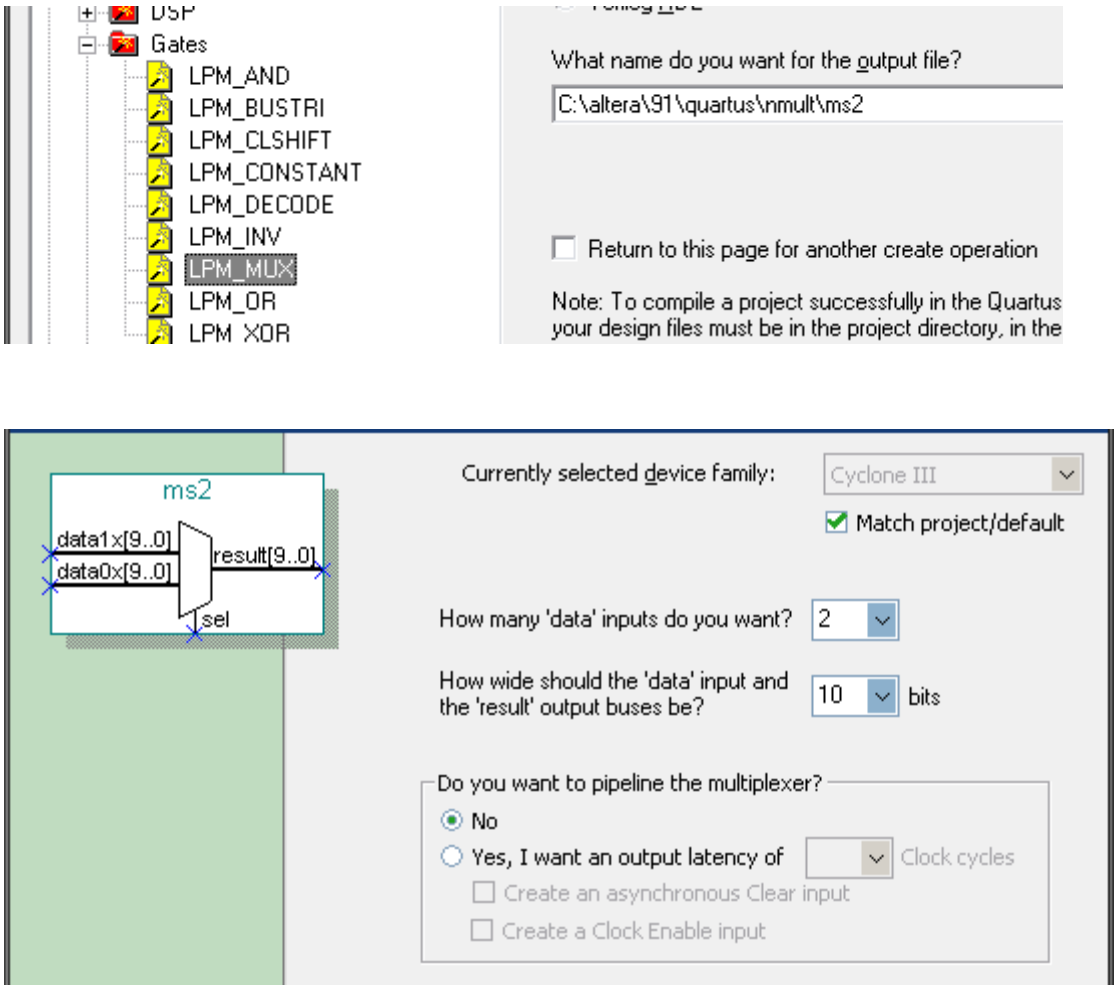
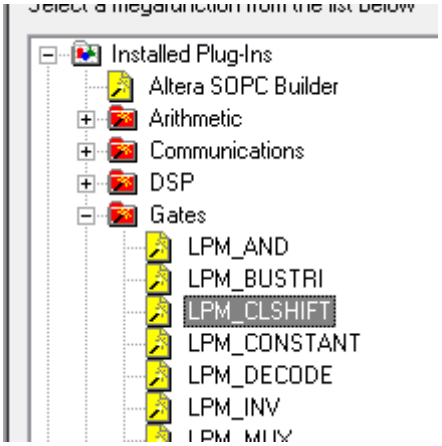
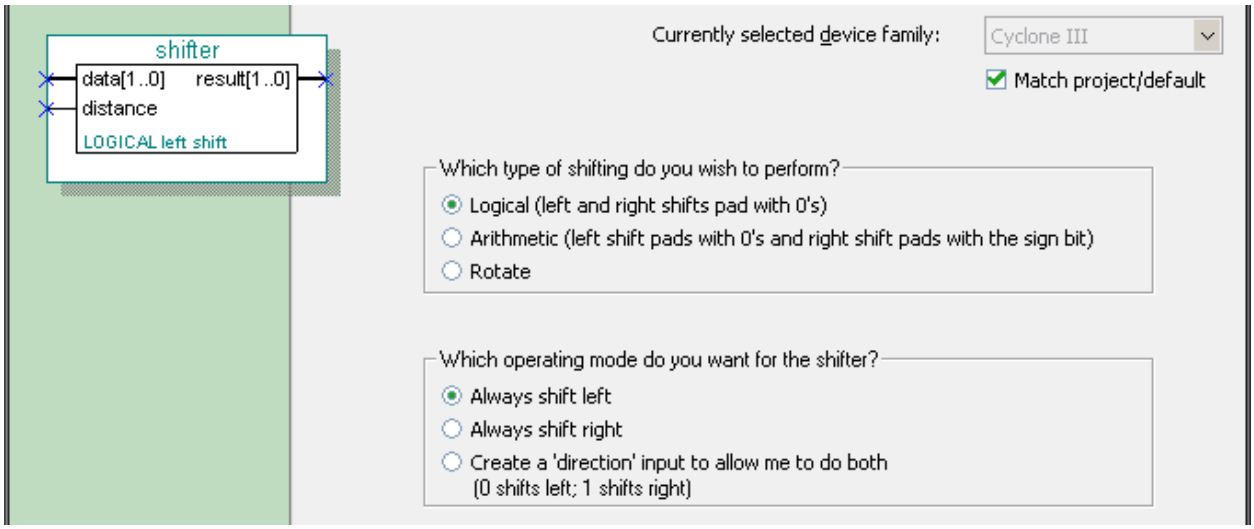


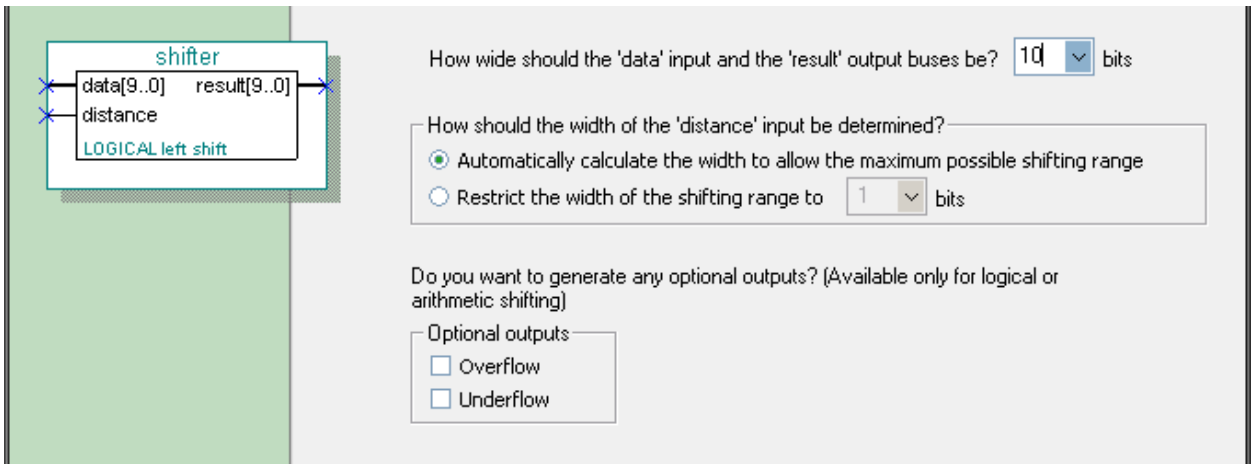
Рис. 42. Создание мультиплексора в Quartus

Для создания схемы логического сдвига нажмите **MegaWizard Plug-In Manager>Next**. Затем в разделе Gates укажите *LPM_CLSHIFT*. Задайте имя (например *shifter*) и нажмите **Next**. Укажите параметры как на рисунке 43а и 43б.



а)





б)

Рис. 43. Создание схемы логического сдвига

Далее нажмите три раза **Next**, затем **Finish**.

На вход схемы логического сдвига подается выход сдвигового регистра для хранения сумм частичных произведений *rgsh*. Выход схемы логического сдвига подсоединяется на второй вход мультиплексора (рисунок 44).

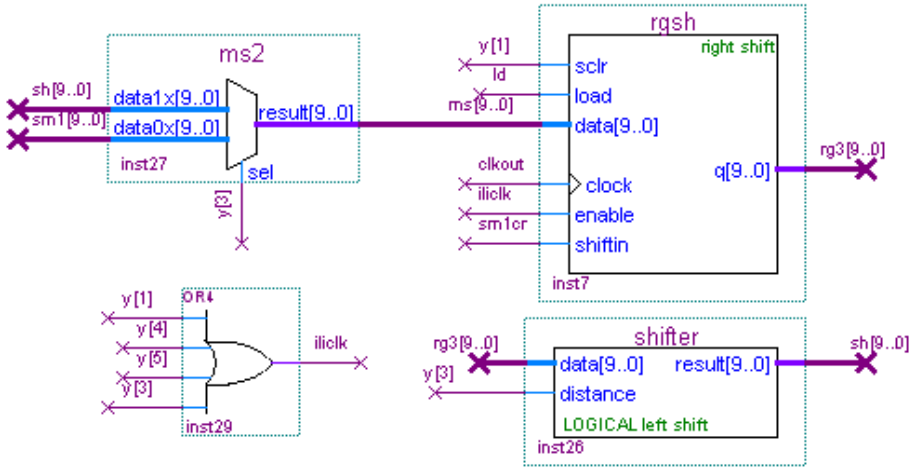


Рис. 44. Соединение схемы логического сдвига, мультиплексора и регистра хранения сумм частичных произведений в Quartus

Счетчик для хранения характеристики результата (рисунок 45) формируется аналогично счетчику тактов. Отличия состоят в том, что его разрядность 6, счетчик работает на вычитание и к нему добавляется вход синхронного сброса *sclr*. Для правильного формирования характеристики результата разряд ct1[4] необходимо инвертировать (является старшим значащим разрядом характеристики). Кроме того, старший разряд в счетчике хранит единицу переноса от суммы характеристик операндов (необходим для определения ПРС и ПМР).

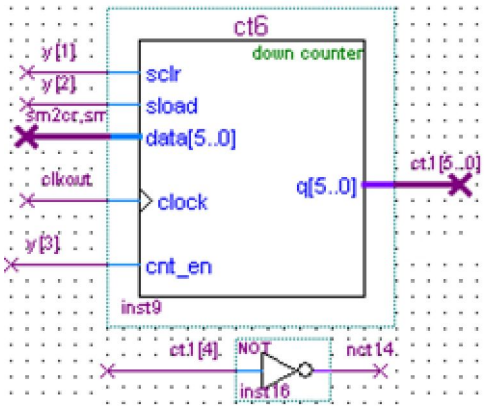


Рис. 45. Счетчик для хранения характеристики результата

Для того, чтобы можно было наблюдать за изменением данных на входах и выходах элементов в процессе работы схемы, а также «видеть» значения осведомительных и предупреждающих сигналов, необходимо организовать массив (шины) «контрольных точек». Для этого можно воспользоваться компонентом *output* из библиотеки символов. Пример представлен на рисунке 46.

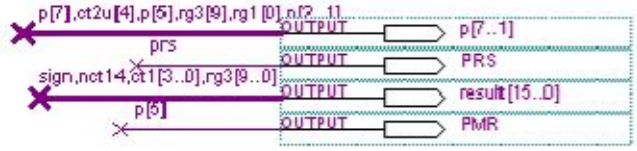


Рис. 46. Выходные шины осведомительных сигналов, признака ПРС и результата операции

Для формирования осведомительных сигналов в имени выходной шины следует указать все разряды, значения которых представляют собой осведомительные сигналы:

p[7] – признак ПРС;

ct2u[4] – старший разряд счетчика тактов;

p[5] – признак ПМР;

rg3[19] – старший разряд регистра сумм частичных произведений;

rg1[0] – младший разряд регистра мантиссы множителя;

p[2..1] – признак временного ПРС и равенства нулю соответственно.

Для шины результата необходимо указать sign – знак, nct14 – инвертированный старший разряд характеристики результата, ct1[3..0] – 4 разряда характеристики результата, rg3[9..0] – мантисса результата из регистра *rgsh*.

После того, как схема передачи данных построена, необходимо подключить к управляющим входам элементов схемы сигналы, сформированные управляющим автоматом.

3.4 Построение управляющего автомата

На первом этапе курсовой работы необходимо построить **операционное устройство**. Но чтобы проверить правильность его работы, необходимо иметь

последовательность управляющих сигналов, подаваемых на входы элементов ОУ в каждом такте. Для этого создадим модель управляющего автомата.

Выберите на панели инструментов элемент Block.  . Далее правой кнопкой мыши вызовите контекстное меню и выберите пункт **`Block Properties>Вкладка I/Os`**. Укажите наименования входов и выходов, а также их тип. Затем присоедините следующие линии связи к блоку (рисунок 47).

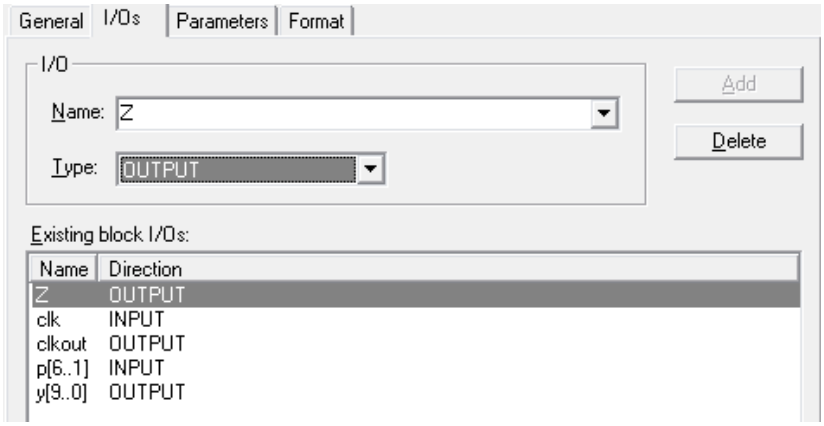


Рис. 47. Формирование сигналов блока УА

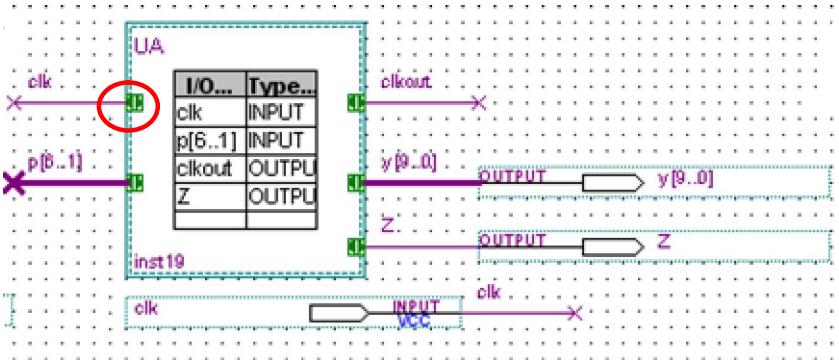


Рис. 48. Вид блока УА в Quartus (б)

Наведите курсор на “зеленые стрелочки” и кликните правой кнопкой мыши (рисунок 48). Выберите свойство Mapper Properties. На вкладке General укажите Type = INPUT или OUTPUT (в зависимости от типа сигнала). На вкладке Mappings поставьте соответствие между сигналом шины и входом блока. Нажмите **Add**, затем ОК. Повторите операцию для всех остальных линий связи, соединенных с данным блоком (рисунок 49).

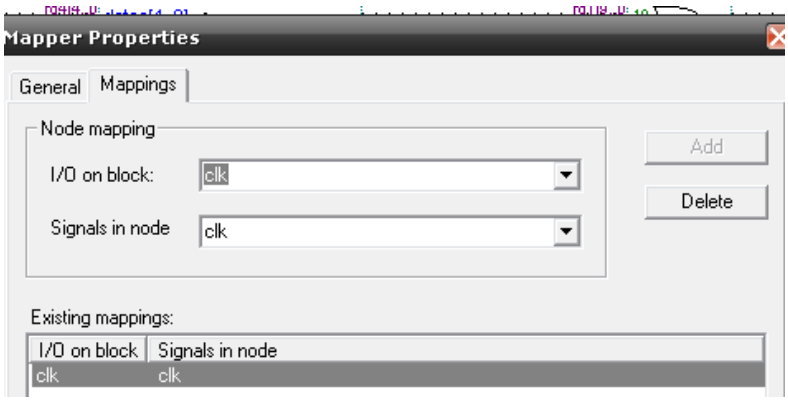


Рис. 49. Окно Mapper Properties

Выделите блок. Правой кнопкой мыши в контекстном меню выберите **Create Design File from Selected Block**…Укажите имя файла и его тип (рисунок 50). Нажмите ОК.

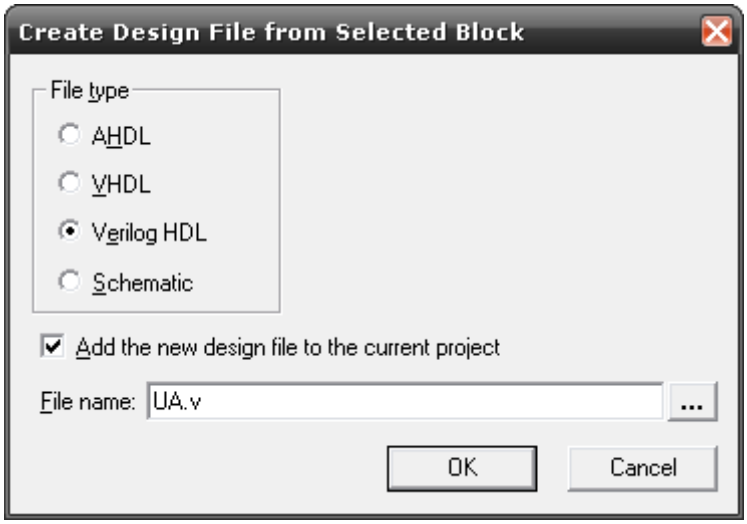


Рис. 50. Создание файла микропрограммы УА

Далее правой кнопкой мыши вызовите контекстное меню, выберите пункт Open Design File. В этом файле с помощью языка описания аппаратуры Verilog Вам предстоит написать микропрограмму работы вашего управляющего автомата.

Идея заключается в следующем. Управляющий автомат срабатывает по положительному фронту синхросигнала. Операционный автомат срабатывает по

положительному фронту сигнала *clkout*, который выдается управляющим автоматом и является инверсией внешнего синхросигнала. Граф-схема алгоритма делится на такты. В каждом такте проверяются определенный набор условий, значения которых задают осведомительные сигналы, поступающие из ОУ. В соответствии с условиями, на выходе управляющего автомата выдаются определенная последовательность управляющих сигналов. Затем осуществляется переход к следующему такту.

Когда Вы откроете файл, то вся микропрограмма должна располагаться после строчек

```
// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        input clk;
        input [6:1] p;
        output clkout;
        output Z;
        output [9:0] y;
// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
//Ваша микропрограмма
endmodule
```

Пример микропрограммы.

```
integer pc=1;  //Объявление счетчика тактов, переменная типа integer
reg [9:0] y;       //Объявление регистра для хранения массива управляющих сигналов
wire clkout;    //Объявление проводника clkout
reg Z=0;            //Объявление регистра для хранения признакак окончания операции
assign clkout=!clk; //clkout является инвертированным сигналом clk, указываем, как система должна вычислять значение clkout
always @(posedge clk)  //Данная функция срабатывает при положительном (pos) перепаде (edge) сигнала clk (по фронту)
```

```
begin
case (pc)
1:begin       //что выполняется на первом такте
y=10'b0000000011;; //число «10» указывает на количество управляющих сигналов, «b» указывает на то, что данные бинарные, «1» указывает на то, что на этом такте УА вырабатывает данный управляющий сигнал
pc= pc + 1; //переход на след такт
end
2:begin       //что выполняется на втором такте
if (p[1]==0) begin //проверка значения осведомительных сигналов «!» - символ отрицания; «&» – логическое И
y=10'b0000000100;
pc = pc + 1; end
else begin    //иначе…
y=10'b1000000010;
pc=9;  end   //переход к такту 9
end
…
end
default y=10'b000000000;       //что выполнять, если тактов больше 9
endcase;
end
endmodule
```

После этого, проект необходимо скомпилировать (Ctrl+L). Если ошибок не возникло, то операционный автомат готов к работе.

Прежде чем открывать схему в программе для проверки работоспособности ОА, необходимо все выводы схемы (элементы *input* и *output*) назвать соответствующим образом (это улучшит производительность и процесс исследования схемы):

x[15..0] – входные данные (размерность шины постоянна и составляет 16 разрядов);

clk или ***c*** – сигнал синхронизации;

result[15..0] – выходные данные (размерность шины постоянна и составляет 16 разрядов);

y[n..0] – управляющие сигналы (разрядность шины согласно разработанной схеме);

p[m..0] – осведомительные сигналы (разрядность шины согласно разработанной схеме);

Z – признак окончания операции;

PRS – признак ПРС;

PMR – признак ПМР;

DEL – признак деления на ноль;

RESET – сброс УА (при построении объединенной схемы).

Для того, чтобы процесс назначения выводов в программе был проще и быстрее, рекомендуется для дополнительных выводов (значения выходов сумматоров, регистров, счетчиков) задавать имена (для элементов *output*), оканчивающиеся словом «out» (рисунок 51).

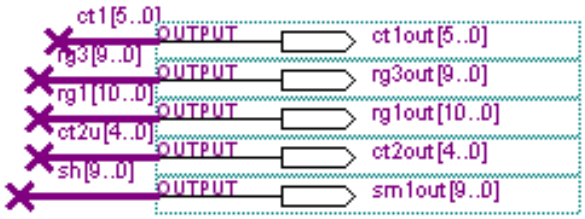


Рис. 51. Дополнительные выводы

4 СИНТЕЗ УПРАВЛЯЮЩЕГО АВТОМАТА В САПР QUARTUS

Синтез УА с использованием САПР Quartus в данной курсовой работе аналогичен синтезу ОА. При этом, в основном, используются логические элементы, элементы памяти (счетчики, триггеры, сдвиговые регистры) и дешифраторы.

Специально подготовленные для данной работы макрофункции находятся в каталоге с программой. Из этого каталога в проект (с помощью пункта меню **Project>Add/Remove Files in Project**…) добавьте необходимые файлы (регистр *reg*, дешифраторы *dc*, счетчики *ct*). Тогда папка с файлами проекта будет выполнять также функции из каталога пользователя. Имена и назначение элементов:

Dc3 – дешифратор с тремя входами;

Ct4 – четырехразрядный счетчик, работающий только на сложение;

Ct3pm – трехразрядных реверсивный счетчик.

Кроме того Вы можете воспользоваться элементами из следующих групп:

Primitives\pin;

Primitives\storage (разрешено использовать только dff – D триггер, jkff – JK триггер, srff – RS триггер, tff – T триггер, в противном случае, программа УА не будет правильно работать с данными);

Primitives\logic;

Primitives\buffer\wire;

Others\maxplus2\74198 (реверсивный сдвиговый регистр).

При указании имени шины, необходимо помнить, что нумерацию ее линий следует начинать с нуля! Имя шины содержит квадратные скобки, где указываются номера, или диапазон номеров линий. Ниже представлена таблица условных обозначений входов и выходов внутри комбинационной схемы (КС), а также входных и выходных шин.

Таблица 9 – Условные обозначения входов и выходов КС для УА

Обозначение	Пояснения
clk	Обозначение тактовых импульсов
reset	Обозначение сброса
x[n..0]	Обозначение осведомительных сигналов (где n любое натуральное число)
xpin[n..0]	Обозначение входной шины осведомительных сигналов
y[n..0]	Обозначение управляющих сигналов
yy[n..0]	Обозначение управляющих сигналов при выдаче на выходную шину
ypin[n..0]	Обозначение выходной шины управляющих сигналов
a[n..0]	Обозначение внутренних состояний
apin[n..0]	Обозначение выходной шины внутренних состояний
d[n..0] (t[n..0])	Обозначение входов на D-триггере (T-триггере)
s[z..0]	Обозначение входов на RS-триггере
j[z..0]	Обозначение входов на JK-триггере
d(t; j; s) pin[z..0]	Обозначение выходной шины входов триггеров
ct[z..0] (rg[z..0])	Обозначение входов счетчика (регистра)
Out[z..0]	Обозначение выходов триггеров и счетчиков
Outpin[z..0]	Обозначение выходной шины триггеров и счетчиков

Индексные номера (указываются в квадратных скобках) при обозначении входов элементов памяти (счетчиков, триггеров, регистров) не должны совпадать!

Если возникает ситуация, когда одной и той же шине данных необходимо задать разные имена, то для этого удобно использовать элемент *wire* из стандартной библиотеки Quartus. Пример использования показан на рисунке 52.

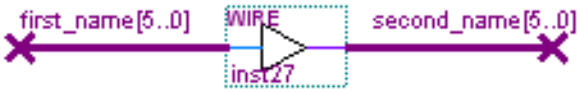


Рис. 52. Использование *wire*

Необходимо отметить, что у элементов памяти, а также дешифратора, необходимо изменить значение «inst» на «UAinst» (это необходимо для правильного функционирования программы «Управляющий автомат»). Для этого просто дважды щелкните на надписи, и внесите изменения (рисунок 53).

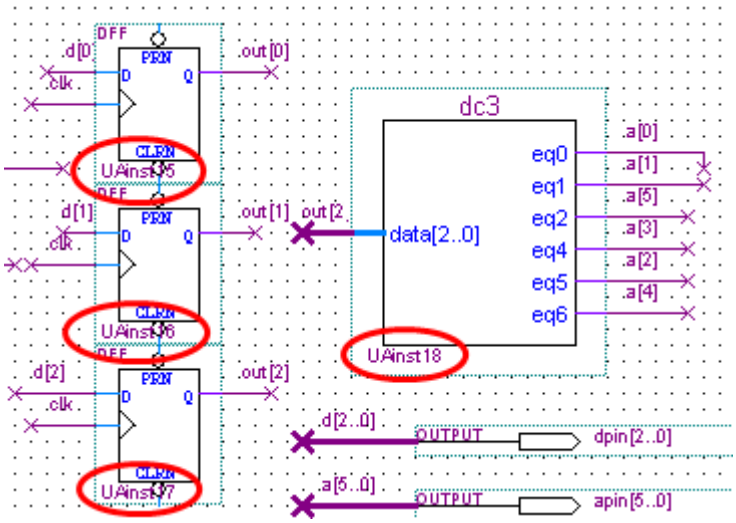


Рис. 53. Правильные значения «inst» у элементов схемы

Следующим шагом после того, как Вы завершите построение схемы, будет создание дополнительного регистра, который необходим для устранения гонок. Скопируйте в проект файлы с именем *reg* из каталога с программой и разместите регистр на схеме. Разрядность данного регистра должна соответствовать количеству выходных сигналов управляющего автомата (управляющие сигналы для ОА «y»). В окне **MegaWizard Plug-In Manager** – *LPM_FF* (рисунок 54) в пункте How many flipflops do you want? (Сколько входов необходимо?) укажите

количество выходных сигналов для Вашей схемы (учитывая, что нумерация начинается с нуля). Нажмите **Finish** два раза.

Если в схеме значения некоторых выходных сигналов напрямую зависят от внутренних связей или состояния выходов КС, например y(4) формируется сигналом, поступающим с выхода a(5) дешифратора, то входную шину данных на регистре можно обозначить следующим образом: вместо y[12..0] Вы можете написать y[12..5],a[5],y[3..0].

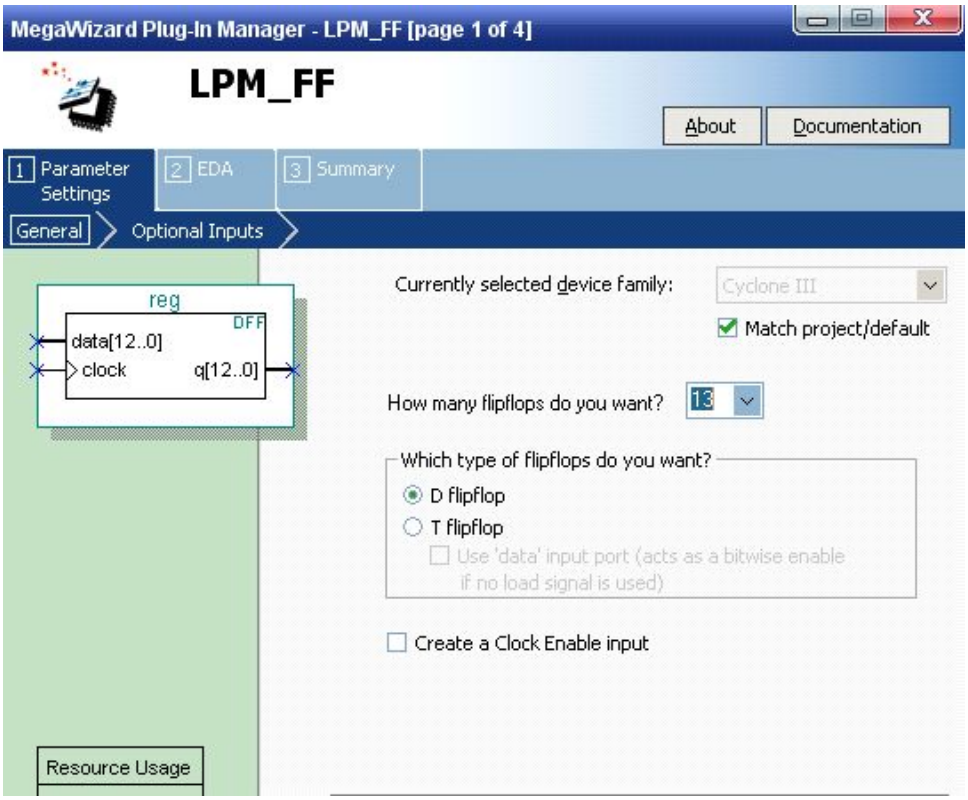


Рис. 54. Окно MegaWizard Plug-In Manager

Схема подключения регистра указана на рисунке 55.

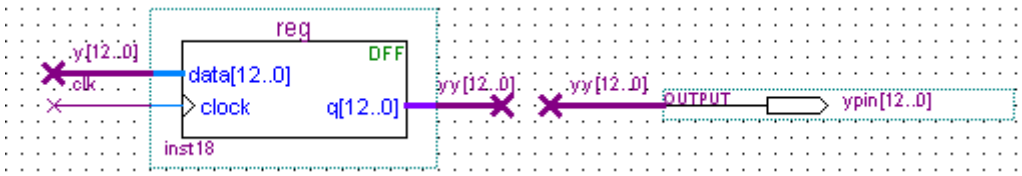


Рис. 55. Дополнительный регистр в схеме

При проектировании цифровых устройств может потребоваться элемент, отсутствующий в библиотеке моделей базовых элементов системы. В этом случае требуемый элемент можно создать на основе базовых элементов системы, а затем оформить его в виде подсхемы (макроэлемента) с прорисовкой условного графического обозначения (УГО). Для решения этой задачи необходим *Символьный редактор*, который позволяет создавать и редактировать УГО (символ). Символьный файл имеет расширение *.bsf.

Под макроэлементом понимается определенным образом оформленная и записанная в рабочую библиотеку проекта схема, которой поставлен в соответствие рисунок УГО. Макроэлементы целесообразно использовать для оформления функционально законченных частей сложного проекта. Само собой разумеется, что оформление схемы в форме макроэлемента выполняется после ее отладки.

Создадим для примера два таких элемента. Первый элемент представлен на рисунке 55.

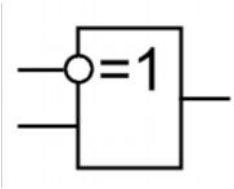


Рис. 55. Макроэлемент 1

Выберите **File>New**. В появившемся окне укажите **Block Diagram/Schematic File**. Сохраните схему, задав ей содержательное имя, отражающее назначение данного макроэлемента.

Разместите на схеме следующие элементы (рисунок 56):
2 элемента input и 1 элемент output из библиотеки libraries/primitives/ pin; элементы not и xor из libraries/primitives/logic.

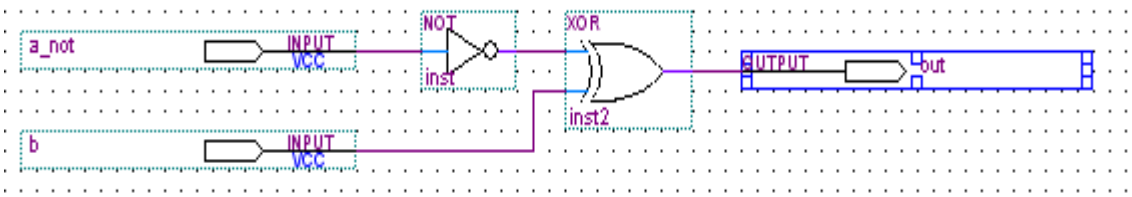


Рис. 56. Схема макроэлемента 1

Скомпилируйте проект, нажав кнопку **Start Compilation**, или выбрав пункт **File>Processing>Start Compilation** в главном меню. Выполните команду **File>Create/Update>Create Symbol Files from Current File** из главного меню. Сохраните файл <имя_Вашего_макроэлемента>.bsf в папке проекта.

Перейдите на вкладку с основной схемой Вашего проекта и щелкните левой кнопкой мыши дважды по пустому месту. Появится окно библиотеки символов. Перейдите в папку Project и выберите только что созданный макроэлемент (рисунок 57).

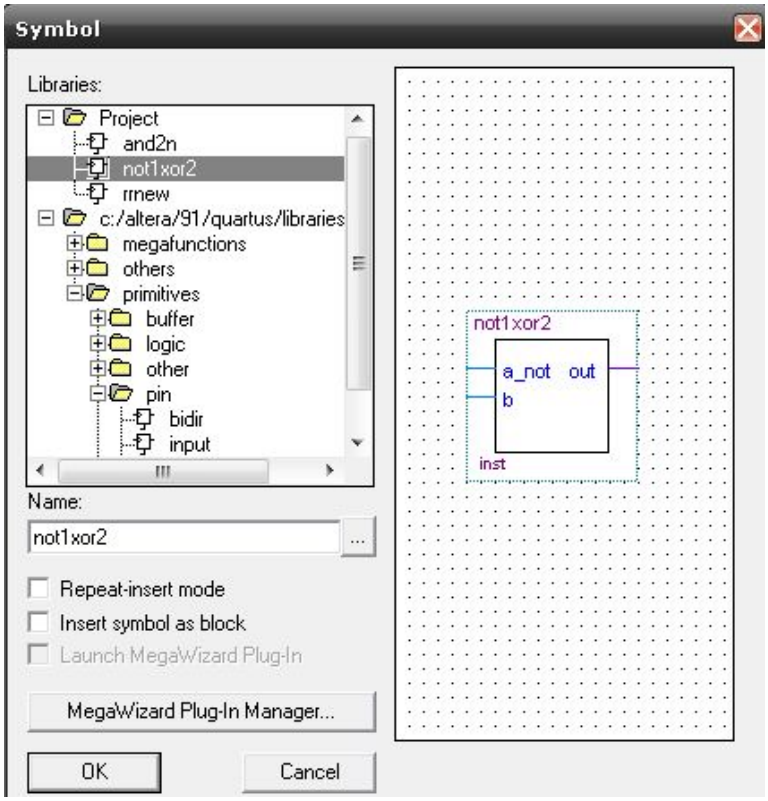


Рис. 57. УГО макроэлемента

Теперь элемент можно разместить на схеме.

Создадим следующий элемент, представленный на рисунке 58.

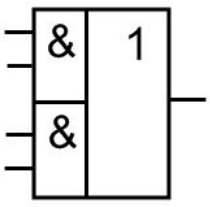


Рис. 58. Макроэлемент 2

Выберите пункт **File>New>Block Diagram/Schematic File**. Создайте схему, приведенную на рисунке 59.

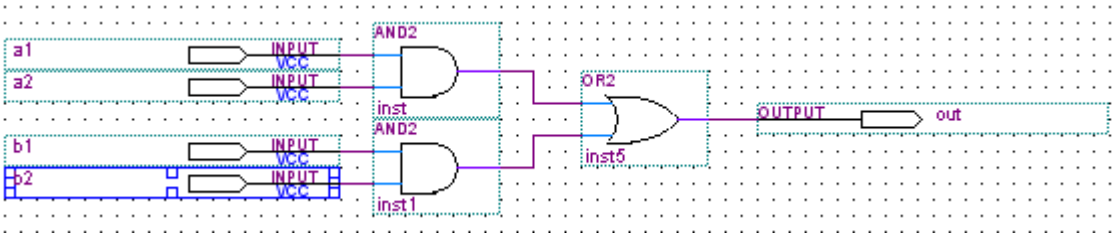


Рис. 59. Схема макроэлемента 2

Выберите **File>Create/Update>Create Symbol Files from Current File**. Внешний вид макроэлемента можно изменить. Для этого выберите **File>Open**. Укажите файл <имя_Вашего_макроэлемента>.bsf. В появившемся окне Вы можете редактировать изображение элемента по своему усмотрению или как представлено на рисунке 60.

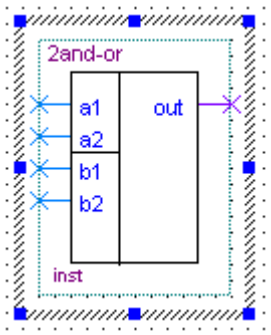


Рис. 60. УГО макроэлемента 2

Для проверки работы УА необходимо создать специальную временную диаграмму. Для этого выберите в меню `File>New`. Далее укажите `Vector Waveform File`. Появится новая вкладка. Слева находится таблица, в которой столбец Name отвечает за названия входных и выходных шин (портов), а столбец Value отображает значения входов/выходов. Чтобы все порты из Вашего проекта оказались в этом списке сделайте следующее. Щелкните правой кнопкой в пустом месте таблицы. В появившемся контекстном меню выберите пункт `Insert>Insert Node or Bus`. Далее нажмите Node Finder. В графе Filter

укажите «Pins: all» и нажмите кнопку List (рисунок 61). Слева появятся все наименования портов из Вашей схемы. Далее нажмите кнопку «>>» (Copy all to Selected Nodes List – Скопировать все в список выбранных узлов). Нажмите ОК дважды.

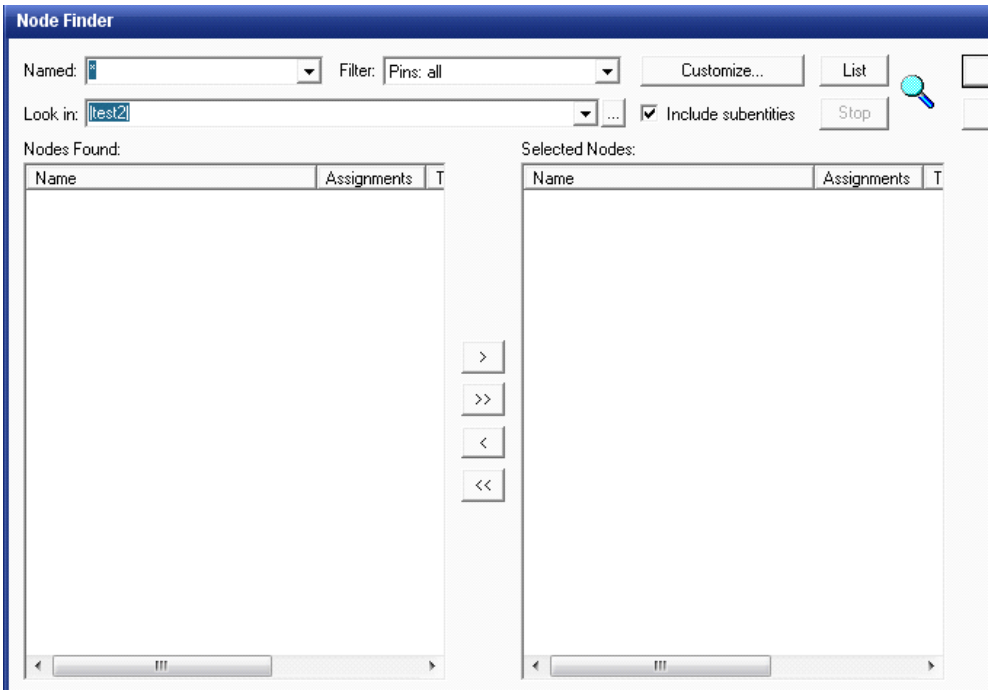


Рис. 61. Окно Node Finder

Найдите в таблице сигнал *CLK*. Щелкните по нему правой кнопкой мыши. Выберите **`Value>Clock`**...(или нажмите Ctrl+Alt+K) В **Time Period** необходимо указать параметры стробирующего сигнала, а именно период (Period), значение которого можно оставить по умолчанию 10 нс, и длительность сигнала высокого уровня в течение этого периода (Duty cycle –оставьте 50%). Нажмите ОК.

Найдите в таблице сигнал *Reset*. Щелкните по нему правой кнопкой мыши. В контекстном меню выберите пункт **`Value>Arbitrary Value`** (Ctrl+Alt+B). Задайте параметры как на рисунке 62.

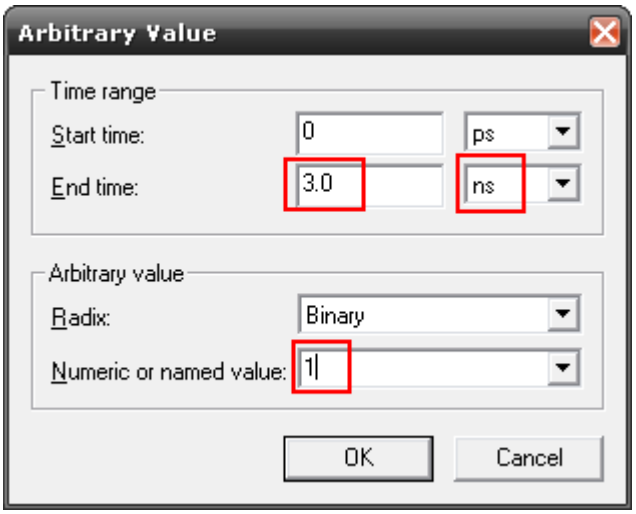


Рис. 62. Окно «Arbitrary Value»

Зайдите в меню **Processing>Simulator Tool** (рисунок 63). В Simulation Mode выберите Functional. В Simulation Input укажите файл <имя_Вашего_проекта>.vwf. После этого нажмите кнопку **Generate Functional Simulation Netlist** (процедуру Generate Functional Simulation Netlist необходимо проводить после каждого изменения проекта).

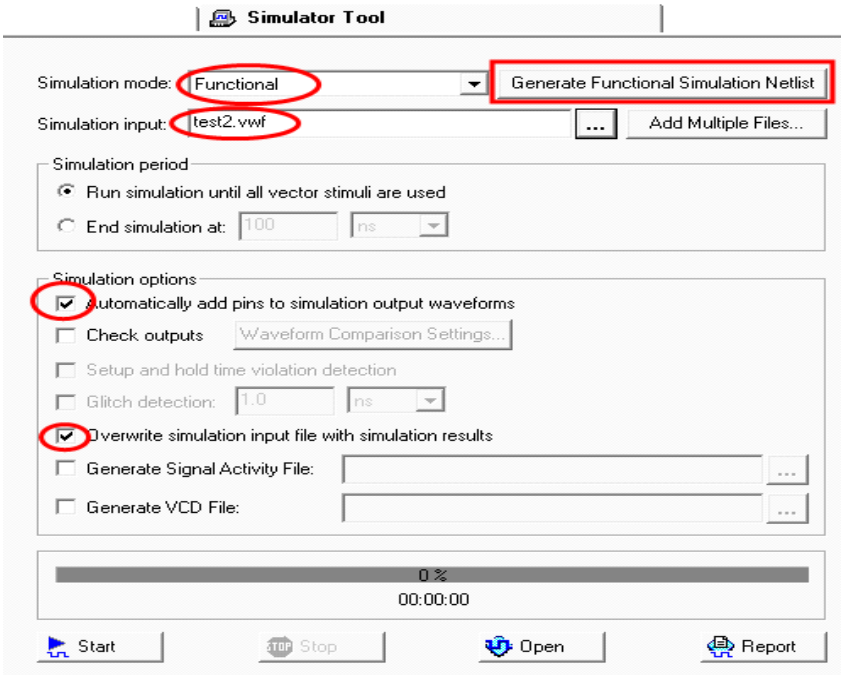


Рис. 63. Диаграмма

Если возникли ошибки, необходимо их исправить. Обязательно поставьте галочку в пункте *Overwrite simulation input file with simulation results*.

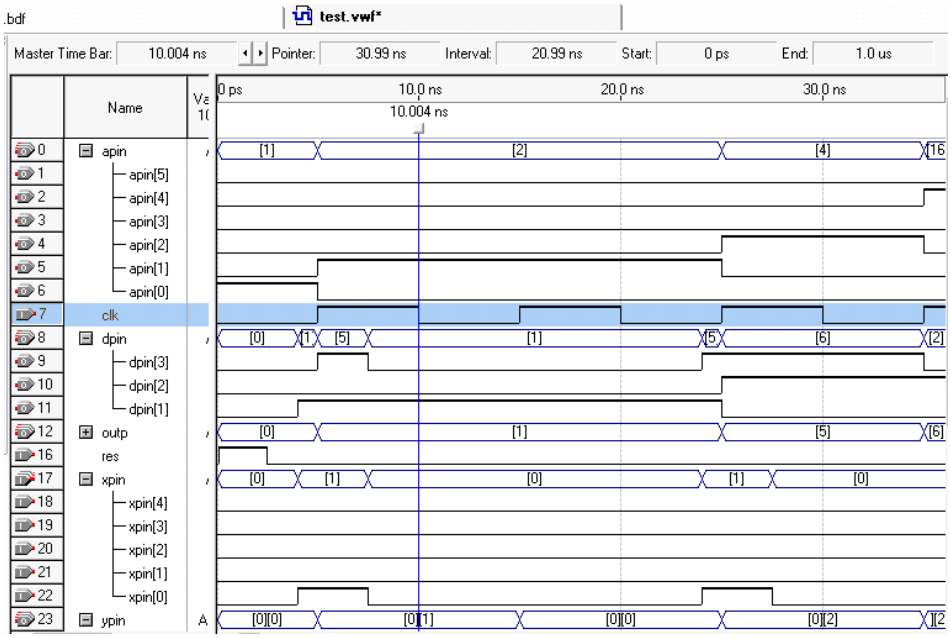


Рис. 64. Окно Simulation Tool

Далее перейдите на вкладку с файлом .vwf попробуйте указать некоторые входные сигналы *xpin*, аналогично работе с сигналом *Reset* выбирая курсором нужный временной диапазон, и задавая ему значения. Срабатывание идет по фронту синхроимпульса, поэтому изменять значения линий входной шины нужно **до начала такта** и **после сигнала общего сброса Reset**.

Нажмите Ctrl+I или в меню **`Processing>Start Simulation`** и запустите процесс модуляции. После того как он закончится, Вы можете снова выбрать вкладку с файлом .vwf и подтвердить изменения. По результатам диаграммы при необходимости нужно подогнать длительность тактового сигнала так, чтобы функционирование УА было стабильным.

В полученном файле диаграмм обнулите все сигналы кроме *CLK* и *Reset*. Запустите процесс модуляции (**`Processing>Start Simulation`**). Сохраните данный файл как <имя_Вашего_проекта>_.vwf.

5. ПОСТРОЕНИЕ ОБЪЕДИНЁННОЙ СХЕМЫ ОПЕРАЦИОННОГО И УПРАВЛЯЮЩЕГО АВТОМАТА

Для того, чтобы объединить синтезированные на ранних этапах ОА и УА, необходимо проделать следующие действия.

Откройте в Quartus проект с ОА. Откройте схему (**не проект, а только схему!**) *.bdf из папки проекта с УА. Сохраните данную схему как *.bdf **в папке проекта с ОА!** Далее, выберите пункт меню **`Project>Add/Revove Files in Project`**… В появившемся окне (рисунок 65) добавьте в проект с ОА все те файлы из проекта с УА, которые Вы использовали для построения регистров, дешифраторов, счетчиков, а также тех элементов, которые Вы непосредственно синтезировали сами.

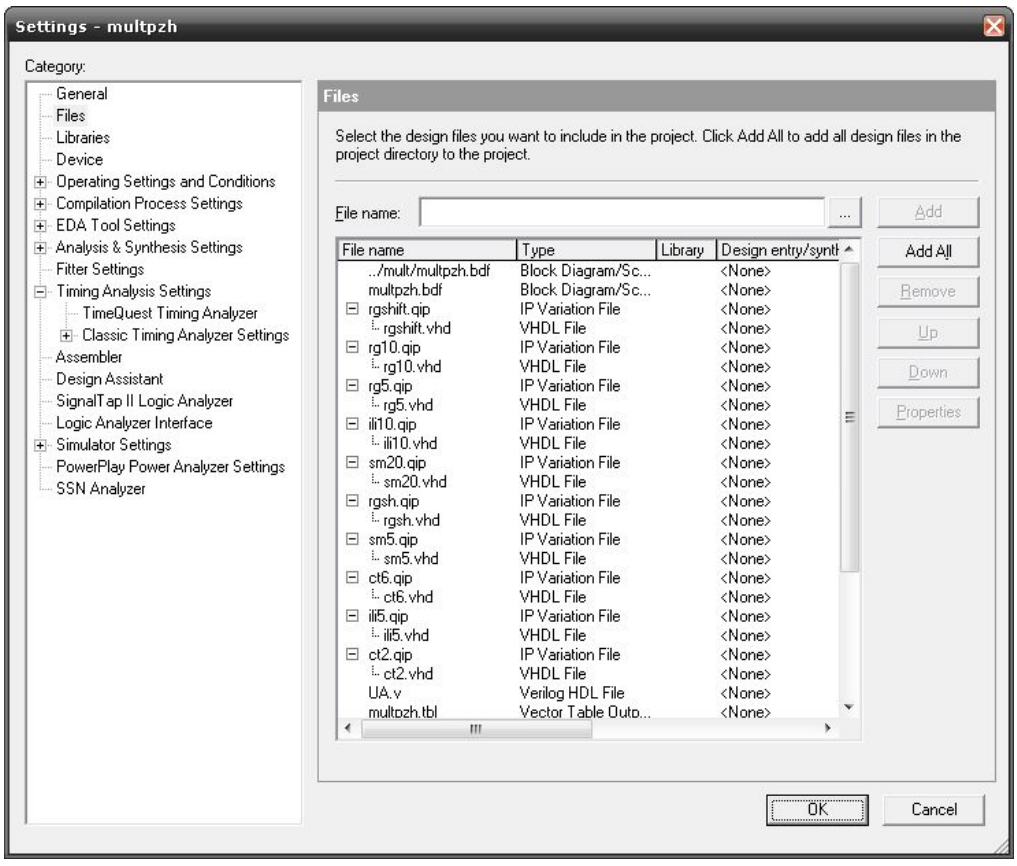


Рис. 65. Добавление файлов к проекту с ОА

Перейдите на вкладку со схемой УА. Уберите в ней лишние выводы (такие как «*out*», и выводы со входов элементов памяти). Добавьте еще два вывода (разместите на схеме элементы *output* и назовите их «Z» и «clkout»). Подключите вывод *Z* к *x[8]*. Из таблицы 5.2 видно, что x9 является интерпретацией

осведомительного сигнала Z, а в схеме УА осведомительные сигналы формируются с нуля. Проинвертируйте тактовый сигнал *clk* и соедините его инверсию с выводом *clkout* (рисунок 66).

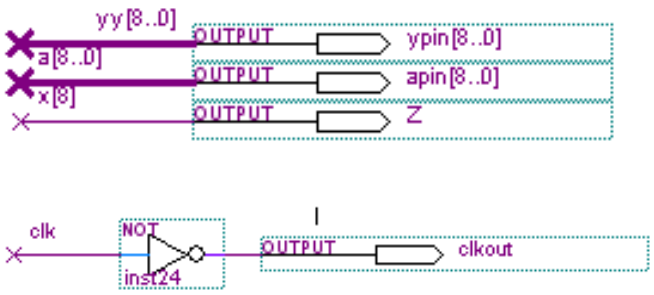


Рис. 66. Выводы УА

Выберите пункт меню **File>Create/Update>Create Symbol Files for Current File.** Теперь УА может быть представлен в виде блока и доступ к нему можно осуществить из библиотеки проекта (Project\<имя_файла_со_схемой_УА>) как показано на рисунке 67.

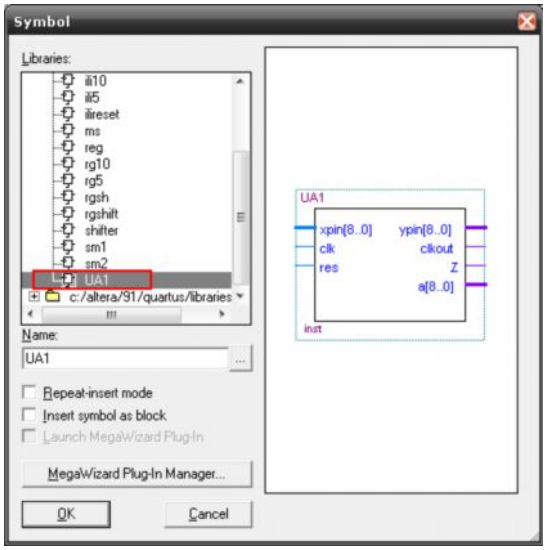


Рис. 67. Выбор схемы с УА

Далее удалите из схемы с ОА функциональный блок, который был интерпретацией УА, и добавьте вывод *Reset* (элемент *input* из стандартной библиотеки Quartus), необходимый для сброса УА в исходное состояние.

На входы *xpin* схемы УА нужно подавать осведомительные сигналы *p*. Поскольку осведомительный сигнал *xpin[0]* (x1 в таблице 5) в первом случае

реализации УА не использовался, то здесь его нужно сформировать. Это можно сделать, используя внутренние состояния УА. Анализируя граф на рисунке 22, можно сделать вывод, что состояния *a[0]* и *a[2]* можно использовать для формирования данного осведомительного сигнала. Аналогично с сигналом *xpin[8]* (рисунок 68).

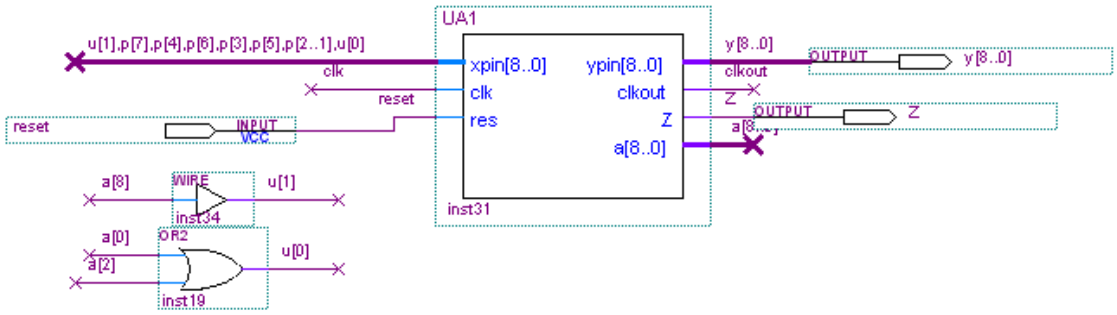


Рис. 68. Формирование осведомительных сигналов *xpin[8]* и *xpin[0]*

При компиляции проекта может возникнуть конфликт из-за того, что на входе *xpin* схемы УА используются имена осведомительных сигналов как с шины *p[7..1]*, так и с выходов других элементов, которые (выходы) также обозначены, как и некоторые осведомительные сигналы шины *p[7..1]* (рисунок 69).

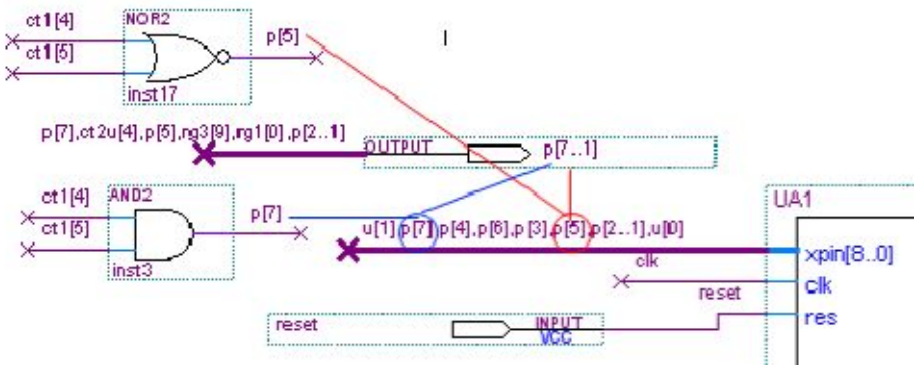


Рис. 69. Возникновение конфликта имен

Для того, чтобы избежать подобной ситуации, необходимо переименовать все выходы элементов в схеме ОА, имена которых совпадают с именами элементов шины осведомительных сигналов (рисунок 70).

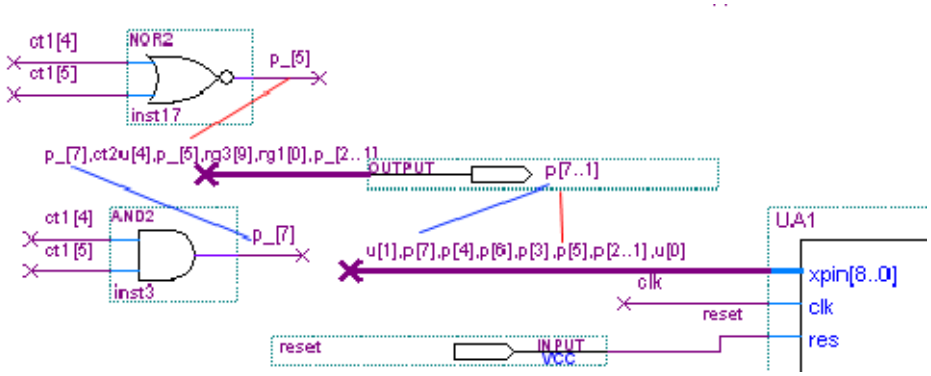


Рис. 70. Переименование выводов

Схему, построенную таким образом, можно исследовать также, как и схему ОА, где в качестве УА использовался функциональный блок с записанной микропрограммой функционирования на Verilog.

Структура ОА, у которого в качестве входных данных используются числа с ПЗ с порядками, имеет некоторые отличия от аналогичной структуры ОА, входными данными которого являются числа с ПЗ с характеристиками.

Характеристики являются смещенными порядками. Значения порядков могут быть как положительными, так и отрицательными. Старший разряд порядка является знаковым.

До начала работы с порядками необходимо обратить внимание на знак порядка:

«0» – порядок неотрицателен, в преобразовании не нуждается;

«1» – порядок отрицателен, необходимо преобразовать в ДК; для этого можно воспользоваться управляемым инвертором (рисунок 71).

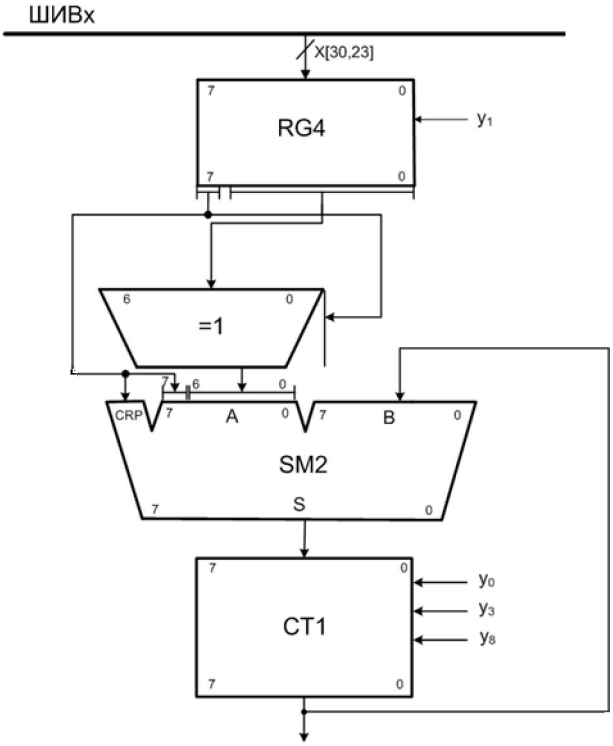


Рис. 71. пример схемы преобразования порядков

Если при сложении порядков одного знака в результате получен порядок, знак которого отличается от знаков операндов, то эта ситуация сигнализирует о возникновении ПРС. Таким образом, для того чтобы выявить ПРС, необходима комбинационная схема (рисунок 42).

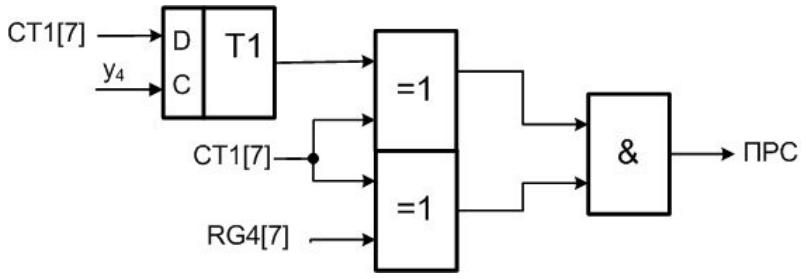


Рис. 72. Пример комбинационной схемы для выявления ПРС

Особого внимания требует ситуация, когда в результате может возникнуть комбинация «100..0», т.е. «1» в знаковом разряде порядка, а во всех остальных – нули. Это может быть как признак временного ПРС (в дальнейшем, если возникает необходимость в нормализации мантиссы результата, устраняется, в

противном случае нужно выдать сигнал «ПРС»), так и ПМР. Поэтому необходимо различать эти ситуации (рисунок 73).

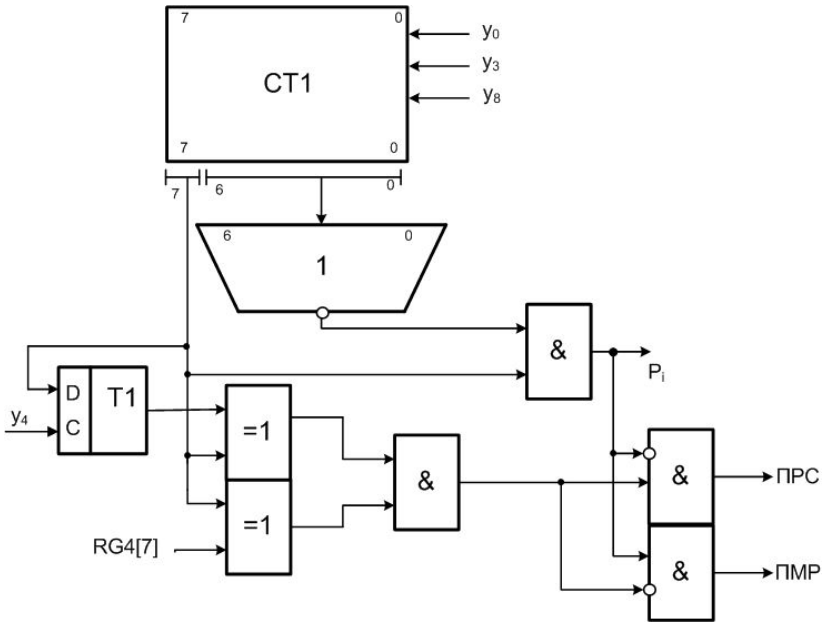


Рис. 73. Пример схемы выявления ПМР и ПРС

Перед тем, как выдать результат умножения на выходную шину, в случае, если порядок отрицателен, необходимо его преобразовать обратно в ПК. Для этого достаточно из текущего значения порядка вычесть «1» и проинвертировать его.

6. ЗАКЛЮЧЕНИЕ

Основной целью учебного пособия является помощь в приобретении практических навыков синтеза операционного и управляющего микропрограммных автоматов для реализации машинных алгоритмов одной из заданных арифметических операций с помощью САПР Quartus.

Для проверки полученных знаний рекомендуется синтезировать вычислительное устройство, исходные данные к которому должны содержать:

- название выполняемой АЛУ операции;
- систему счисления и форматы данных;
- особенности алгоритма выполнения операции;
- логический базис для синтеза комбинационной схемы.

7. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Савельев А.Я. Прикладная теория цифровых автоматов. М.: ВШ, 1987.

2. Каган Б.М. Электронные вычислительные машины и системы. М.: Энергоатомиздат, 1985.

3. Баранов С.И., Скляров В.А. Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986.

4. Майоров С.А., Новиков Г.И. Структура электронных вычислительных машин. М,: Машиностроение, 1979.

5. Лысиков Б.Г. Арифметические и логические основы цифровых автоматов. Минск: ВМ, 1980.

6. ГОСТ 2.708-72 ЕСКД. Правила выполнения электрических схем цифровой вычислительной техники.

7. Кутепова Е.С., Фадеева Т.Р. Дискретная математика и цифровые автоматы. Двоичное умножение и деление. метод. указания к практическим занятиям. Киров: Ротапринт, 1987 - 27с.

8. Мельцов В.Ю., Фадеева Т.Р. Синтез микропрограммных управляющих автоматов: учеб. пособие. –Киров: ВятГТУ, 2000, - 56с.