Лабораторная работа №7

Тема: Моделирование ассоциативного процессора с применением последовательных (рекуррентных) алгоритмов

Цель работы: освоение навыков построения и верификации (проверки) моделей ассоциативного процессора с применением рекуррентных алгоритмов.

Краткие теоретические сведения

Теоретические сведения, необходимые для выполнения лабораторной работы №7, включающие сведения об особенностях построения и функционирования ассоциативных запоминающих устройств параллельного действия с поиском, параллельным по словам и последовательным по разрядам, выполнении поисковых операций с применением последовательных (рекуррентных) алгоритмов, приведены в теоретической части данного ЭРУД (тема 14, подразделы 14.1.1-14.1.1).

Напомним основные положения из этих сведений.

Рассмотрим ассоциативные процессоры с пословной организацией, т.е. с параллелизмом на уровне слов и с обработкой их последовательно по разрядам.

Множество слов образует ассоциативный массив или АЗУ ассоциативно организованного процессора, структура которого приведена на рисунке 1. Соответственно имеются логические цепи на каждое слово, т.к. весь разрядный срез АЗУ может обрабатываться параллельно.

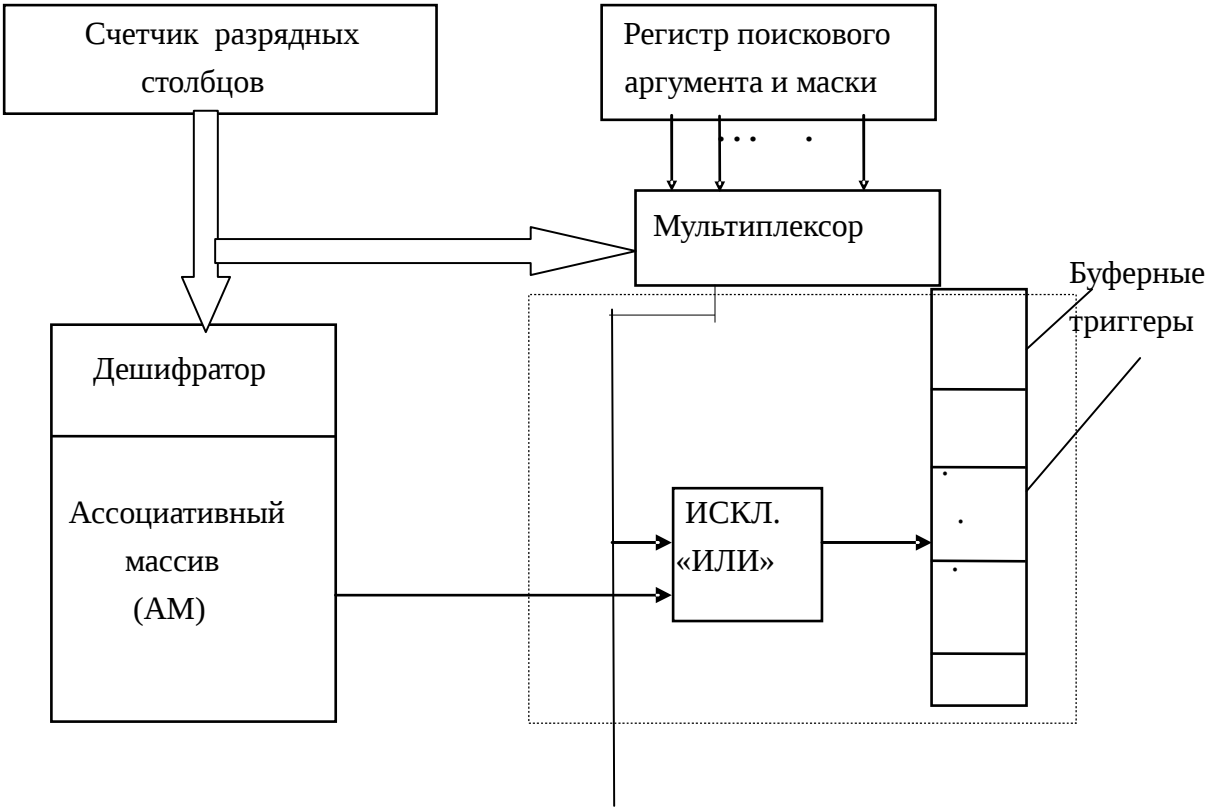

Рисунок 1. Структура ассоциативного процессора

В АЗУ, параллельных по словам и последовательных по разрядам, для обработки данных используются последовательные (рекуррентные) алгоритмы. Это позволяет наряду с элементарными операциями сравнения на равенство применять и более сложные логические и вычислительные алгоритмы.

Отметим, что АЗУ может функционировать в двух основных режимах: поисковом и вычислительном.

В режиме поиска обычно требуется локализовать и считать из памяти слова, подчиняющиеся определенным условиям. Например, это могут быть:

– слова, равные, большие или меньшие некоторого аргумента;

– слова, лежащие в заданном диапазоне чисел;

– слова, обладающие максимальной или минимальной величиной среди чисел, хранящихся в памяти, и др.

В процессе поиска содержимое памяти, как правило, не меняется.

В режиме ассоциативных вычислений слова, наоборот, подвергаются обработке, а полученные результаты вновь поступают в массив АЗУ, причем

часто они заменяют ранее хранившиеся там слова (или определенные фрагменты этих слов).

Возможность работы АЗУ в различных режимах обеспечивается благодаря специальной конструкции памяти результатов, которая состоит из логических цепей последовательного типа (по одной на каждое слово АЗУ).

Рассмотрим работу памяти результатов для проведения операций сравнения.

Алгоритмы для варианта с началом сравнения, начиная со старших разрядов, описывается следующими рекуррентными соотношениями[9]:

$$g_{ji}=g_{j,i+1}\vee(\bar{a}_i\wedge S_{ji}\wedge\bar{l}_{j,i+1}),$$

$$l_{ji}=l_{j,i+1}\vee(a_i\wedge\bar{S}_{ji}\wedge\bar{g}_{j,i+1}), \qquad (1)$$

где g_{ji}, $l_{j,i}$ – двоичные переменные, вычисляемые в процессе выполнения поразрядного сравнения;

a_i – i-й разряд аргумента поиска;

$S_{j\,i}$ – i-й разряд j-го слова, хранящегося в массиве памяти.

Эти алгоритмы являются удобными для реализации при помощи цепей последовательного типа.

Будем рассматривать алгоритм, выполняющий сравнение разрядов, начиная со старшего разряда n. В качестве начальных значений задаются

$$g_{j,\,n+1}=l_{j,\,n+1}=0.$$

Последовательные цепи памяти результатов могут строиться либо по асинхронному, либо по синхронному принципу. Лучше применять синхронные схемы, так как они более просты в управлении, имеют более высокую помехозащищенность. В качестве элементов памяти используем наиболее простые триггеры с синхронизацией – D-типа или другого типа (рисунок 2).

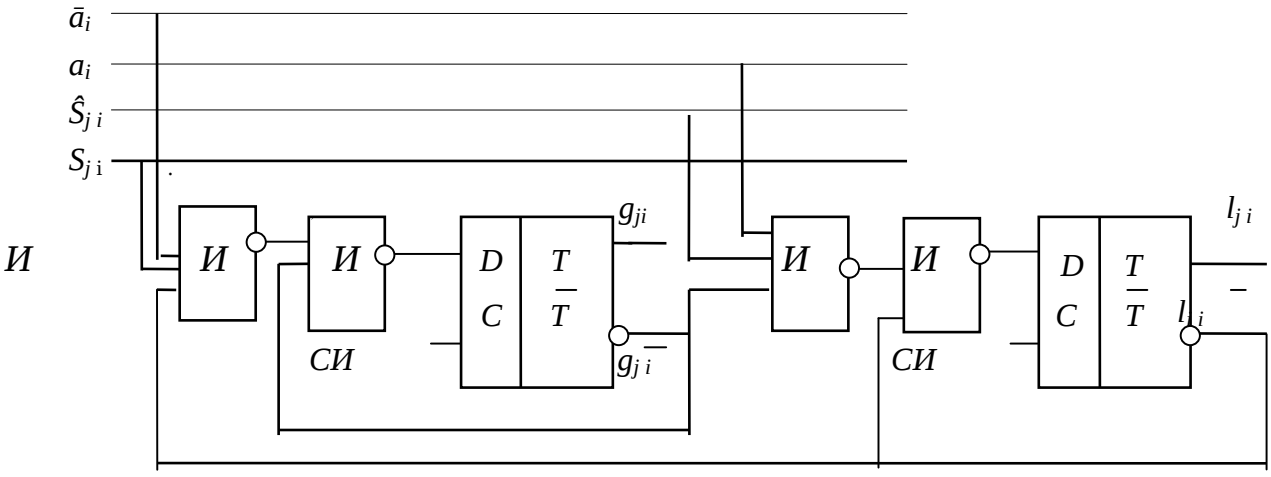

Рисунок 2. Ячейки промежуточных результатов, используемые при сравнении величин

На схеме, обслуживающей слово j, имеются два триггера – g_j и l_j, которые будем называть ячейками промежуточных результатов. Логические значения выходов этих ячеек, как и переменные в выражении (1), обозначим g_{ji} и l_{ji} соответственно (на схеме имеются также вспомогательные вентили (*И*)).

При сравнении содержимого заданного слова памяти с аргументом поиска значения их одноименных разрядов a_i и $S_{j\,i}$ последовательно и синхронно передаются на входы ячеек промежуточных результатов. В каждом такте *D*-триггеры переходят в новые состояния, которые соответствуют значениям логических переменных g_{ji} и l_{ji} в выражении (1).

Окончательный результат сравнения определяется совокупностью логических значений, зафиксированных на выходах триггеров после завершения поразрядного анализа содержимого слова памяти и поискового аргумента.

Если $g_{j\,0} = l_{j0} = 0$, то $S_j = A$,

$g_{j\,0} = 1$, *а* $l_{j\,0} = 0$, то $S_j > A$,

$g_{j0} = 0$, *а* $l_{j\,0} = 1$, то $S_j < A$.

Отметим, что состояние $g_{j\,0} = l_{j\,0} = 1$ принципиально невозможно.

Сигналы с выходов памяти результатов подаются на приоритетный анализатор, при помощи которого осуществляется считывание слов, удовлетворяющих заданным соотношениям.

Основные типы операций поиска

Выше были кратко перечислены основные типы операций поиска. Рассмотрим их несколько подробнее.

Поиск величин, заключенных в заданном интервале

Для реализации операций граничного поиска в каждой логической цепи памяти результатов, кроме триггеров *g* и *l*, необходимо иметь еще один вспомогательный триггер. При помощи этих триггеров, называемых далее флажками результата, фиксируются слова, входящие одновременно в два

набора, каждый из которых составлен из элементов, удовлетворяющих одному частному ограничению.

Перед началом поиска все флажки результата устанавливаются в «1». На каждом промежуточном этапе флажки, соответствующие словам, для которых частное условие поиска не выполняется, сбрасываются в «0». Таким образом, по завершении операции поиска единицы остаются только в тех флажковых триггерах, которые связаны со словами, удовлетворяющими всем ограничениям.

Поиск величин, заключенных в заданных границах, проводится в 2 этапа:

1) отыскиваются все числа, которые меньше верхней границы;

2) отыскиваются те числа, которые больше нижней границы.

После окончания поиска флажки результата отмечают только те величины, которые находятся в требуемом интервале.

Поиск максимального (минимального) значения

Для этого поиска внешний аргумент поиска не нужен. Вместо него разряды регистра аргумента, начиная со старшего, последовательно устанавливаются в значение «1». После установки в «1» очередного разряда производится операция маскированного поиска, в ходе которого отбираются все слова, содержащие "1" в соответствующей позиции.

Если на каком-то шаге ни одного такого слова не обнаружено (т.е. во всех словах памяти в этом разряде стоят «0»), содержимое памяти результатов не меняется. После каждого шага количество слов, отмеченных единичными флажками результатов, уменьшается. Процесс продолжается до тех пор, пока не будут опрошены все разряды массива АЗУ. Если в памяти записано несколько максимальных чисел, равных друг другу, все они будут зафиксированы в памяти результатов.

Очевидно, что поиск минимального значения в целом выполняется аналогично. Он также начинается со старших разрядов, но на каждом шаге выявляются слова, содержащие в соответствующих позициях не «1», а «0».

Поиск ближайшего снизу (сверху) значения

Эта операция состоит из 2-х этапов:

1) выявляются все числа, меньше указанной величины;

2) среди выявленных на 1-м этапе чисел находится максимальное число.

Если необходимо отыскать числа, ближайшие сверху к заданной величине, производится поиск чисел, превосходящих последнюю, и среди них отбираются минимальные.

Упорядоченная выборка (сортировка)

Эта операция заключается в пошаговом поиске максимального или минимального значения в наборе чисел, отсеянных после прохождения предыдущего этапа сортировки. Эту операцию можно реализовать в виде последовательности рассмотренных выше операций.

Поиск по соответствию

В некоторых случаях требуется найти записи, двоичные представления которых содержат максимальное количество разрядов, совпадающих с разрядами поискового аргумента. Для решения этой задачи в памяти результатов АЗУ необходимо иметь набор счетчиков, при помощи которых осуществляется подсчет совпадающих значений одноименных разрядов в каждом слове памяти. Если указанные счетчики, в свою очередь, имеют средства адресации по содержанию, в них можно привести поиск максимального значения и определить таким образом слова, наиболее схожие с поисковым аргументом.

Поиск на основе булевых функций

Введя в состав АЗУ соответствующие логические цепи, можно организовать поиск, критерием в котором будет выступать некоторая булева функция. Поиск такого типа применяется при локализации слов, для которых логическая функция, определенная на части разрядов, при подстановке в нее соответствующих битов аргумента принимает заданные значения. Чаще используются такие функции, как «Исключающее ИЛИ», «Логическая эквивалентность», ИЛИ, И.

Вычислительные операции ассоциативного процессора

К вычислительным операциям ассоциативного процессора можно отнести арифметические и логические операции над разрядными срезами, а также над байтами, полями, слова, которые будут рассмотрены в работе №8.

Контрольные вопросы:

1. Какая логическая операция является базовой в АЗУ?
2. Какие логические функции используются для выполнения базовой логической операции?
3. Приведите логические формулы для определения совпадения и несовпадения аргумента поиска A и слова памяти S_j с учетом маскирования (c).
4. Назовите способы реализации сравнения аргумента поиска A и слова памяти S_j?
5. Какие алгоритмы чаще используются при выполнении поисковых операций в АЗУ?
6. Назовите основные типы поисковых операций в ассоциативных процессах?
7. Определите соотношение между A и S_j, если q_{ji} =1, l_{ji}=0

Задание

Построить и проверить программную модель ассоциативного процессора с применением последовательных (рекуррентных) алгоритмов.

Модель должна обеспечивать выполнение поисковых операций ассоциативного процессора. Варианты операций приведены в таблице 1.

Таблица 1

Вариант	Поисковые операции	Примечания
1	Поиск ближайшего сверху (снизу) значения	Оба варианта
2	Поиск величин, заключенных в заданном интервале	
3	Упорядоченная выборка (сортировка)	По возрастанию и убыванию
4	Поиск по соответствию	
5	Поиск по основе булевых функций	

Примечание: Вариант, выполняемый каждым студентом, определяется преподавателем.

Требования к программе:

Разработанная программа должна уметь выполнять следующие функции:

-вычислять значения логических переменных q_{ji} и l_{ji}

-формировать двоичный массив размером m слов по n разрядов;

-выполнять поисковые операции, приведенные в данном методическом материале (таблица 1);

-выводить результаты выполнения заданных поисковых операций.

Методика выполнения:

1. Строится программа вычисления значений логических переменных q_{ji} и l_{ji}.
2. Производится формирование двоичного массива размером m слов по n разрядов с использованием программы генерации псевдослучайных чисел.
3. Строится программа выполнения поисковых операций, приведенных в таблице 1 (не менее 2-х).
4. Производится поиск и вывод на экран результатов выполнения заданных поисковых операций.

Стандартная процедура проверки разработанной программы заключается в анализе результатов выполнения предъявленных программе требований.