УДК 681.324-192

Средства моделирования и анализа параллельных процессов в рекуррентном операционном устройстве

Морозов Н.В., Степченков Ю.А.

1. ВВЕДЕНИЕ

Идеи и принципы рекуррентного операционного устройства, см. работы [1 и 2], после стадии разработки оптимальной структуры и моделирования на бумаге, потребовали более точного и удобного инструмента исследования для дальнейшего развития. Естественный выбор - программное моделирование. Первоначально исследовали операционный уровень (ОУ) рекуррентного операционного устройства. Рассматривалось несколько разных архитектур ОУ, для каждого варианта ОУ была разработана своя модель. Длительное и тщательное исследование свойств и сравнение полученных характеристик, позволило, на основе данных моделей, принять решение по наиболее оптимальной структуре ОУ, основным составляющим элементам и их функциям. Подробно сей процесс, и полученные результаты описаны в работе [3]. Централизованная структура ОУ (ЦСОУ), рожденная таким способом, является наиболее эффективной и использовалась для дальнейших исследований.

Настоящая работа описывает возможности средств моделирования и анализа параллельных процессов ЦСОУ. Данные функции выполняет программа ОПЕРА.

Главной для модели ОУ являлась задача обоснованного выбора архитектуры. При разработке ОПЕРЫ преследовались также следующие цели:

- изучение альтернативных вариантов структур ОУ;
- определение достаточного набора модулей (обрабатывающих устройств) и оптимизация их функций;
- выбор наилучшего способа связи секционных обрабатывающих устройств в единое обрабатывающее поле;
- установление комплекта операндов и их функциональных полей;
- определение размера тегов (функциональных полей) и их соотношение с размером других полей в операндах;
- оценка объема памяти, требующейся для каждой секции;
- определение требований ко всем средствам связи и устройствам.

Кроме того, к модели операционного уровня архитектуры РОУ был предъявлен ряд специфических требований:

- обладать определённой степенью гибкости;
- допускать варьирование степенью параллелизма (числом секций) в широких пределах;
- наращивать число функций преобразователей и арифметико-логических устройств;
- собирать данные для производства количественных оценок основных показателей качества структур РОУ: производительности, пропускной способности и других.

Одним словом быть податливым инструментом в руках пытливого исследователя.

Еще один аспект при моделировании ОУ программой ОПЕРА - это программирование алгоритмов для разрабатываемого ОУ, т.е. составление капсул с операндами, задающих конкретный алгоритм, и определение последовательности преобразований функциональных полей для всех операндов. Трудностей здесь несколько, а именно:

- надо отобразить распараллеливание алгоритма на структуру ОУ,
- отладить прохождение капсулы,
- настроить рекуррентные преобразователи.

В рамках данной статьи кроме описания программы ОПЕРА и ее основных функций мы попробует дать представление о ее возможностях в плане помощи в подготовке и отладке капсул-алгоритмов.

2. СТРУКТУРА ПРОГРАММЫ

Структура программы ОПЕРА соответствует ЦСОУ подробно описанной в [3]. Структура моделируемого ОУ (см. рис. 1) содержит следующий набор модулей функциональных устройств и каналов связей:

- *распределитель* операндов (входной FIFO-буфер);
- *итератор*, активный буфер заменяемых операндов (имитирует кольцевой буфер-счетчик FIFO);
- *экспликатор*, устройство формирования горстей операндов;
- *ключ* или блок мультиплексоров (на входах ПАП);
- *память* адресной проверки (ПАП) - секционная память совпадений;
- *жонглёр*, блок "жонглирования" операндами;
- *память подкачки тегов* (ППТ);
- *вычислитель* (обрабатывающее устройство);
- *аккумулятор*, регистр содержащий предыдущий результат вычислителя;
- *тасовщик*, блок тасования операндов;
- *импликатор* (формирователь выходных горстей операндов);
- *сборщик* наборов данных (выходной RAM-буфер);
- *входной* и *выходной* (рекуррентные) *преобразователи* тегов;
- а также шины и каналы связи, объединяющие структурные компоненты в единый операционный уровень.

Программа ОПЕРА моделирует работу ЦСОУ с помощью пошагового моделирования работы каждого составляющего элемента. Последовательность работы задает и контролирует модуль Диспетчер. Прохождение операндов от модуля к модулю обеспечивает общая шина. Схема работы программы изображена на рис. 2. Каждый модуль модели связан с другими модулями через шину общих данных: выбирая операнды, предназначенные для него и посылая результирующие операнды к другим. Модуль диспетчера общается с внешним миром и отслеживает внутреннюю логику моделирования ОУ, принимая на вход капсулу из файла и формируя и записывая в файл результирующий набор данных.

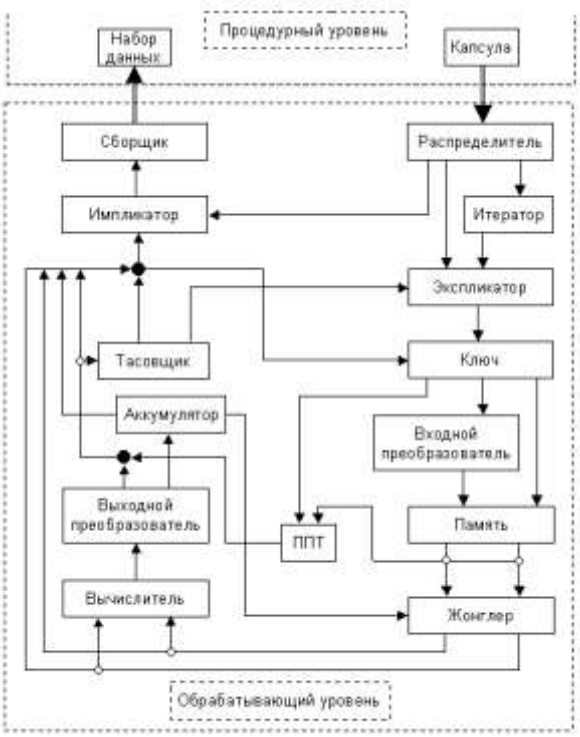

Рис. 1. Обобщенная схема РОУ: состав и связи

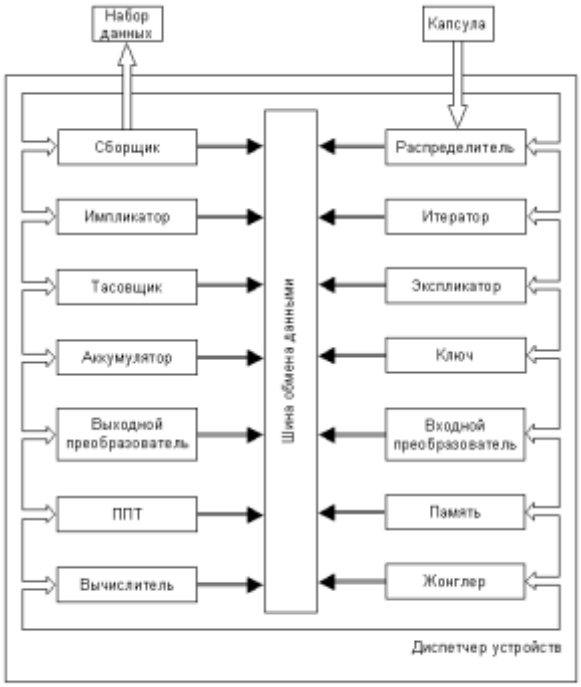

Рис. 2. Программная модель РОУ

3. ОБЩИЕ ВОЗМОЖНОСТИ

Программа ОПЕРА предназначена для моделирования работы операционного уровня архитектуры РОУ путем имитации выполнения параллельных алгоритмов.

На вход программы поступают капсулы (последовательный набор операндов) с данными и тегами - функциональными полями (ФП), задающими алгоритм их обработки. В результате моделирования функционирования ОУ формируются выходные наборы данных. Вход и выход, капсулы и наборы данных (НД), могут поступать и возвращаться в вышестоящий уровень архитектуры и, в этом случае, программа ОПЕРА может быть полезна в моделировании всей архитектуры РОУ.

Процесс моделирования идет по шагам. Один шаг соответствует одному полному циклу срабатывания каждого составляющего ОУ под управлением Диспетчера. Сам Диспетчер тоже выполняет полный цикл своей работы. Все происходящее в ОУ, продвижение операндов, изменение ФП и данных, отображается программой в состояниях всех элементов ОУ.

Основным методом моделирования является пошаговое выполнение капсулы. После каждого шага можно подробно изучить состояние ОУ и каждой ее секции. Для достаточно больших капсул имеется возможность установить точку останова. После запуска моделирования останов произойдет перед выполнением заданного шага или обработкой заданного операнда.

Работа программы ОПЕРА возможна в нескольких режимах.

В ручном режиме моделирования работа преобразователей и вычислителя имитируется, т.е. задаются, пользователем. Программа их запоминает и в последующем, если они не требуют изменения, ими можно пользоваться, проводя моделирование в полуавтоматическом режиме.

Контроль за ходом процесса моделирования в этих режимах осуществляется по содержимому основного поля (окна) программы ОПЕРА, отображающему состояние текущей секции ОУ. На нем схематично отображены: прямоугольными блоками все основные компоненты ОУ, и линиями их взаимосвязи по основным шинам. Всего отображаются следующие блоки:

- распределитель и экспликатор,
- ключ и память,
- жонглер портов,
- вычислитель,
- тасовщик и кольцо,
- импликатор и сборщик.

Полностью автоматический режим моделирования возможен для числовой капсулы, которая содержит в себе все необходимые данные для моделирования.

4. ФУНКЦИОНИРОВАНИЕ

Изначально параллельные алгоритмы могут быть эффективно реализованы в рамках модели рассматриваемого ОУ. Для алгоритмов с менее выраженной параллельностью эта задача более нетривиальна. Комбинаторные и другие алгоритмы общего назначения потребуют очень тонких решений, большого опыта и хорошей ориентации в непростой структуре ОУ. Только точные знания всех многочисленных возможностей и каждого блока в отдельности помогут в разработке требуемой капсулы. Перечислим основные функции каждого блока:

Распределитель	принимает операнды из капсулы и направляет их по месту назначения в импликатор, итератор или экспликатор.
Итератор	принимает управляющие заменяемые операнды из капсулы от распределителя во внутренний буфер, который по запросу экспликатора продвигается и выталкивает верхний операнд (если он есть) в экспликатор. Операнд многократного действия после выталкивания снова попадает в буфер.
Экспликатор	основная его функция - из поступающих операндов от тасощика, итератора и распределителя сформировать горсть, т.е. набор операндов, направляемых в каждую рабочую секцию ОУ на текущем шагу. Горсть может состоять из двух полугорстей - двух операндов направляемых в одну секцию. Сформированная горсть поступает в ключ.
Ключ	процесс формирования горсти начинается в ключе из уже ранее поступивших операндов и 'работающих' в ОУ и недостающие добираются экспликатором. В ключе формируется две полугорсти: одна непосредственно направляется в память, а другая - сначала во входной преобразователь.
Входной преобразователь	выполняет две функции. Каждый поступивший операнд может быть: а) задержан на несколько шагов

	и б) его ФП могут быть преобразованы в новый набор ФП.
Память	хранит поступившие операнды и выдает жонглеру 'вспыхнувшую' пару операндов (последний пришедший операнд попал в уже занятую ячейку).
Жонглер	определяет код операции и ФП для поступившей пары операндов из памяти или аккумулятора и направляет данные и код операции в вычислитель, а ФП в выходной преобразователь.
Вычислитель	непосредственно выполняет требуемую операцию над данными. Результат всегда отправляется в аккумулятор.
Выходной преобразователь	выполняет преобразование ФП и формирует новый операнд, добавляя к результату вычислителя новые ФП. Полученный операнд отправляется в тасовщик.
Тасовщик	определяет секцию назначения нового операнда. Операнд может, как остаться в своей секции, так и быть направленным в другую секцию или же размноженным по нескольким секциям.
Импликатор	производит 'отлов' готовых операндов и помещает их в сборщик.
Сборщик	позволяет формировать два набора данных и отправлять их на верхний уровень по мере готовности.

5. ПОСТРОЕНИЕ И ОТЛАДКА КАПСУЛЫ

Помимо моделирования ОУ главной целью программы ОПЕРА является облегчение процесса построения и отладки функционирования капсул. В это понятие входит - определение последовательности операндов в капсуле, набора требуемых ФП, последовательности их рекуррентных преобразований и установка необходимых режимов функционирования отдельных составляющих элементов для выполнения разрабатываемых алгоритмов на моделируемом ОУ. Само же устройство, режимы его работы и взаимосвязи многофункциональны и сложны. Как следствие процесс построения капсул и выработка алгоритма обработки данных долог и труден.

С целью облегчения этого процесса разработка алгоритма начинается с построения символьной капсулы. В символьных капсулах данные и поля, требующие преобразований, представляются символьными кодами (по аналогии с вычислениями - запись в виде формулы). ФП операнда определяют в каждом элементе ОУ его судьбу - что с ним делать и куда направлять далее. Попадая в преобразователь, операнд получает новый набор ФП. В процессе моделирования преобразование ФП задается пользователем, в соответствии с алгоритмом, и запоминается программой. Будем называть далее это 'данными о преобразованиях'. Разрабатываемый алгоритм полностью задается символьной капсулой вместе с данными о преобразованиях. Капсула задает операндам начальный набор ФП и определяет последовательность их появления в ОУ. Данные о преобразованиях

содержат информацию о трансформации ФП в течение моделирования алгоритма от шага к шагу.

Элементом ОУ, позволяющим делать алгоритмы и, следовательно, капсулы независимыми от данных, является вычислитель. Задавая данные символьными именами, одну и туже капсулу можно использовать для расчета разных данных.

В символьную капсулу вместо символьных данных, т.е. обозначений, можно подставить конкретные числовые величины. Моделирование символьной капсулы с конкретными данными производится в режиме 'Вычисление Символьной Капсулы'. Полученный в этом случае набор данных будет содержать числовой результат вычислений, по которому может быть проверена правильность функционирования капсулы.

Следующий этап - переход к полностью числовому виду капсулы. В числовых капсулах поля, требующие преобразований, представляются числовыми кодами. В этом режиме данных о преобразованиях для моделирования ФП не требуется - они уже содержатся в исходных значения ФП. Окончательно правильность подготовки и функционирования данной капсулы подтверждается моделированием в числовом режиме.

6. ПРИМЕР

Проиллюстрируем все сказанное на конкретном примере. Пример взят из стандарта кодирования голоса GSM [4]. Требуется составить капсулу для алгоритма автокорреляции. Исходная формула следующая:

$$ACF(k) = \sum_{i=k}^{N-1} S(i) * S(i-k),\ k = 0, 1, .., 8,\ N = 1,2, \ldots ,160.$$

где: ACF(k) - массив коэффициентов корреляции, S(N) - массив исходных отсчетов сигнала.

Данный алгоритм 'изначально параллелен' и очевидно может быть эффективно реализован. Процедура вычисления автокорреляции требует 10 параллельных секций и описывается уравнениями:

```
Для k-й секции (0 < k < 8)        :
s (i) = s (i-1)                   ; задержка в 1 отсчет
p(i-k,i) = s(i-k) * s(i)          ; результат произведения
c(i-k,i) = p(i-k,i)+ c(i-k-1,i-1); сумма произведений
                                  ; для i ≥ 2
r = c(i-k,i) | @T                 ; вывод результата
```

Для простоты возьмем N=13 и составим следующую капсулу:

```
Ac: Cx=3 Cp=9 Cc=h_f Cr=0001 Ce=s Cs=ei_x Ca=! ;
Ai: I0n=ACF I0i=s I0s=9 ;
Atm: Tc= Tm= Tu= Tr= Ts= Te= Ta= Tn=0 To=0 [ Sm=2 Dr=1111 ];
Cacn: [ Oc=+ Ou=l Sm=1 Dr=1111 Ds=on De=n Da=n ];
Cc: Cj=f Cd=1 Cl= Ce= Cr=[Oc=nop Ou=k Sm=1 Dr=1111 Ds=nn De= Da=];
Di V=0  [ Oc=+ Ou=l Sm=2 Dr=1111 Ds=on De=n Da=n ];
Di: V=s0 [ Oc=+ Ou=e Sm=0 Dr=0000 Ds=on De=n Da=n ];
Di: V=s1 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s2 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s3 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s4 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s5 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s6 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
```

```
Di: V=s7 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s8 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s9 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s10 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s11 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Di: V=s12 [ Oc=* Ou=e Sm=0 Dr=0000 Ds=nn De=n Da=n ];
Cacn: [ Oc=-> Ou=l Sm=1 Dr=1111 Ds=tn De=n Da=n ];
Cadm: [ Oc=nop Ou=e Sm=0 Dr=1111 Ds=nn De=n Da=n ];
Cacd:
```

Капсула содержит несколько установочных операндов, за ними расположены основные операнды с данными, Di:, и завершают капсулу операнды заканчивающие работу и очищающие устройство. Для реального примера (N=160) достаточно 9 вспомогательных операндов (не основных обрабатываемых операндов, Di) в капсуле - 9 операндов из 169. Т.е. накладные расходы на организацию вычислительного процесса (ВП) - вычисления девяти коэффициентов корреляции - составляют менее 6%, которые включают в себя передачу в ОУ:

- установочных операндов (аналог исполнительной программы в фон-неймановских и потоковых архитектурах);

- собственно обрабатываемых данных;

- очищающих операндов, полностью подготавливающих ОУ для приема и обработки новых капсул (таким образом, решающих и проблему "уборки" мусора).

В начале капсулы идут операнды, устанавливающие требуемые режимы работы для всех блоков ОУ, а именно операнд глобальной конфигурации Ac:, операнд инициализации наборов данных Ai:, операнд шаблона Atm:, заменяемый операнд Cacn:, операнд конфигурации секции Cc:. Они задают следующие основные параметры:

- секции - задействовано 9 секций;
- память - сдержит по 3 ячейки;
- режимы экспликатора - горстевой сдвиг с размножением первого, начальная маска косвенной репликации - 0001;
- режим ключа - расширенное слияние E- и I-горстей;
- задержка во входном преобразователе равна 1;
- режим жонглера - F-операнд на правую шину;
- режим кольца - связи нет, кольцо отключено;
- импликатор содержит один шаблон, отлавливающий операнды направляемые во вторую ячейку памяти;
- сборщик - один НД с именем ACF на 9 элементов;

Кроме того в 1 ячейку памяти каждой секции заносится заменяемый операнд Cacn:. Он является основным рабочим операндом алгоритма и встречает все приходящие по этому адресу действительные операнды.

Обработка всех этих настроечных операндов заканчивается за один шаг. Второй шаг предназначен для организации быстрого вхождения в цикл - во вторую ячейку памяти всех секций заносится нулевое значение. При этом поведение ОУ при получении первых элементов суммы `c(i-k,i) не будет отличаться от последующих.`

Далее начинается выборка из капсулы действительных операндов и работа собственно алгоритма корреляции. Экспликатор совместно с ключом, в соответствии с установленным режимом, формирует первую горсть операндов,

которая поступает в память. Коэффициенты корреляции ACF(k) содержат разное число элементов суммы – 13 для секции 0 (для рассматриваемого примера N=13), 12 – для секции 1 и 5 для секции 8. Поэтому в соответствии с выбранным режимом репликации секции вовлекаются в обработку постепенно (см. ниже). Так как оба операнда в горсти направляются в память по одному адресу, то происходит 'вспышка' и они оба поступают в жонглер и далее в вычислитель.

Попадая в вычислитель, данные из каждой пары операндов перемножаются и на выходе секций формируются операнды с новыми наборами ФП, получаемыми из выходного преобразователя. Схема выполнения шага 3 представлена следующей таблицей (в обработку вовлечена только секция 0):

секция	0	1	2	3	4	5	6	7	8
1-я полугорсть	s0								
2-я полугорсть	s0								
операция	*								
результат	p00								

Результат выполнения шага 3, операнд {p00}, получая новый набор ФП, направляется на повторную обработку по E-шине в эту же секцию в ячейку 2. Операнд на E-шине имеет самый высокий приоритет, что запрещает считывание операндов из Распределителя (входной капсулы). Поэтому в секции 0 происходит "вспышка" и инициация "пустой" операции сложения – своеобразная плата за идентичность данного и последующих шагов ВП при минимуме настроечных параметров и длины капсулы. Результат выполнения шага 4, операнд {c00}, получая новый набор ФП, направляется на повторную обработку по E-шине в эту же секцию во входной преобразователь, где им определена задержка в 1 шаг - он сработает - выйдет из входного преобразователя, только на шаге 6. Схема выполнения шага 4 представлена следующей таблицей.

секция	0	1	2	3	4	5	6	7	8
1-й операнд пары	p00								
2-й операнд пары	0								
операция	+								
результат	c00								

Шаг 5 повторит действия шага 3 для секции 0 и вовлечет в обработку секцию 1. За исключением того, что добавится операнд s1 на входе. На выходе секции 0 результат нет необходимости куда-либо направлять, т.к. он будет использован на следующем шагу прямо из аккумулятора (действие определяют функциональные поля операнда s1). На выходе секции 1 результат направляется на повторную обработку по E-шине в эту же секцию в ячейку 2 по аналогии с шагом 3 в секции 0 (действие определяют функциональные поля операнда s0). Схема выполнения шага 5 представлена следующей таблицей:

секция	0	1	2	3	4	5	6	7	8
1-я полугорсть	s1	s0							
2-я полугорсть	s1	s1							
операция	*	*							
результат	p11	p01							

Шаг 6 тот самый шаг, на котором срабатывает положенный в самом начале

во 1 ячейку памяти заменяемый операнд. На этом шаге из входного преобразователя выходит (а по мере вовлечения секций в работу –выходят) операнд (операнды) лежащие там, {c00}, и увлекают за собой заменяемый операнд, вместо которого жонглер подставляет значение из аккумулятора. Схема выполнения шага 6 представлена следующей таблицей:

секция	0	1	2	3	4	5	6	7	8
1-й операнд пары	p11	p01							
2-й операнд пары	c00	0							
операция	+	+							
результат	c11	c01							

С каждым следующим нечетным шагом в ВП будет вовлекаться новая секция до тех пор, пока не будет вовлечена в обработку последняя, восьмая секция на шаге 19, схема выполнения которого представлена следующей таблицей.

секция	0	1	2	3	4	5	6	7	8
1-я полугорсть	s8	s7	s6	s5	s4	s3	s2	s1	s0
2-я полугорсть	s8	s8	s8	s8	s8	s8	s8	s8	s8
операция	*	*	*	*	*	*	*	*	*
результат	p88	p78	p68	p58	p48	p38	p28	p18	p08

Шаг 20 – последний шаг, где имеет место некоторое различие в работе секций, которое привносится операндом s0. Схема выполнения шага 20 представлена следующей таблицей.

секция	0	1	2	3	4	5	6	7	8
1-й операнд пары	p88	p78	p68	p58	p48	p38	p28	p18	p08
2-й операнд пары	c77	c67	c57	c47	c37	c27	c17	c07	0
операция	*	*	*	*	*	*	*	*	*
результат	c88	c78	c68	c58	c48	c38	c28	c18	c08

Начиная с 21 шага, все секции работают идентично (обработка операнда s0 полностью завершена). Каждый следующий нечетный шаг характеризуется завершением обработки очередного действительного операнда – s1, s2, … s159.

Становится очевидным механизм работы - чередование умножения и сложения нужных элементов данных прямо в соответствии с алгоритмом, который мы описывали в начале.

Завершение фазы вычислений - формирование результатов в Импликаторе и выдача их ОУ происходит на шагах 29 и 30. После того как все действительные операнды в капсуле исчерпаны, на шаге 29 на обработку поступает вспомогательный операнд Cacn, который направляется во все секции одновременно и замещает там ранее хранившийся операнд. Именно эта замена и обеспечивает на шаге 30 изменение хода ВП – инициация не операции сложения, а выдача результата в Импликатор, который производит формирование набора данных.

В заключение заметим, что цикл вычислений данного алгоритма состоит из 2 шагов сразу в 9 секциях. Для расчета нашего конкретного примера понадобится всего 31 шаг: 25 шагов на выполнение собственно алгоритма, 1 шаг на инициализацию (шаг 1), 2 шага на вхождение в цикл (шаги 2 и 4), 2 шага на выдачу

результата (шаги 29 и 30) и 1 шаг на очистку ОУ (шаг 31). На современных DSP процессорах только на вычисление алгоритма понадобится 115 тактов вычислений.

7. ВЫВОДЫ

Разработана программа ОПЕРА для моделирования работы операционного уровня архитектуры РОУ путем имитации выполнения параллельных алгоритмов. Разработаны средства и методика построения и отладки капсул. С помощью программы реализовано несколько тестовых алгоритмов, которые подтвердили правильность выбранного направления в построении архитектуры РОУ.

ЛИТЕРАТУРА

1. Филин А.В. Особенности обработки сигналов на процессоре с рекуррентно-динамической парадигмой вычислений. // Системы и средства информатики: Вып. 11. - М.: Наука, 2001.

2. Филин А.В. Динамический подход к выбору архитектуры вычислительных устройств обработки сигналов. // Системы и средства информатики: Вып. 11. - М.: Наука, 2001.

3. Степченков Ю.А., Петрухин В.С., Филин А.В. Рекуррентное операционное устройство для процессоров обработки сигналов. // Системы и средства информатики: Вып. 11. - М.: Наука, 2001.

4. Recommendation GSM 06.10, February 1992, page 19.