УДК 681.5.62–5+681.5.681.3

АКСЕЛЕРАТОР ДЛЯ БЫСТРОГО ПРЕОБРАЗОВАНИЯ КОНСТРУКТИВНЫХ ПОДМНОЖЕСТВ ВЕРШИН ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ

Э.И. Ватутин, И.В. Зотов В.С. Титов
Юго-западный государственный университет
Россия, 305040, Курск, ул. 50 лет Октября, 94
E-mail: evatutin@rambler.ru, zotovigor@yandex.ru, titov-kstu@rambler.ru

Ключевые слова: система логического управления, параллельный алгоритм, проектирование логических мультиконтроллеров, разбиение, комбинаторная оптимизация, R-выражение, ориентированные деревья, комбинаторно-логический акселератор

В работе приведено детальное описание структурно-функциональной организации комбинаторно-логического акселератора для быстрой обработки конструктивных подмножеств вершин граф-схем параллельных алгоритмов. Структура акселератора базируется на использовании многопортовой ассоциативной памяти, принципах векторного параллелизма и специализированных комбинационных схемах, учитывающих специфику выполняемых операций. Приведены оценки аппаратной сложности и быстродействия акселератора, показана возможность его реализации на современной элементной базе.

ACCELERATOR FOR FAST TRANSFORMATIONS OF SUBSETS OF VERTICES OF PARALLEL ALGORITHMS / E.I. Vatutin, I.V. Zotov, V.S. Titov (South West State University, 94 50 Let Oktyabrya, Kursk 305040, Russia).

Key words: logic control system, parallel algorithm, logic multicontroller design, separation, combinatorial optimization, R-expression, oriented trees, combinatorial logic accelerator.

The work presents detailed description of structural and functional organization of accelerator for fast transformations of subsets of vertices of parallel algorithms. Accelerator structure is based on using multiport associative memory, principles of vector parallelism and a number of special combinatorial circuits which takes into account special features of performed transformations. Estimates of the hardware complexity and speed of the accelerator are given. Ability of accelerator implementation using modern element base is showed.

1. Введение

Одной из основных тенденций развития современных систем логического управления (СЛУ) является широкое внедрение принципов параллельности и модульности. Параллельные модульные СЛУ, часто называемые логическими мультиконтроллерами (ЛМК), способны выполнять комплексные управляющие алгоритмы теоретически неограниченной сложности за счет их декомпозиции на компоненты, распределяемые между модулями. Особый интерес представляют ЛМК, формируемые из однотипных

микропрограммных модулей (контроллеров) и позволяющие достичь сверхвысокой производительности, обеспечивая при этом достаточную гибкость и хорошую контролепригодность.

Построение ЛМК сопряжено с необходимостью решения ряда оптимизационных задач на дискретных структурах. Особое место среди них занимает задача выбора разбиения алгоритмов управления, решаемая в процессе их декомпозиции. Качество ее решения напрямую влияет на аппаратную сложность мультиконтроллера и определяет время выполнения алгоритма управления.

В работах [1–10] предложен и развит параллельно-последовательный метод решения указанной задачи, основанный на серии эквивалентных редукционных преобразований конструктивных подмножеств вершин (*R*-выражений) и применимый к достаточно широкому классу управляющих алгоритмов. В монографии [2] показано применение данного метода при построении системы логического управления роботизированной сборочной ячейкой. В [11] описана визуальная система декомпозиции алгоритмов логического управления PAE [12], содержащая его программную реализацию [6]. В работах [13, 14] в ходе ряда вычислительных экспериментов показано превосходство метода над известными аналогами в условиях наличия технологических ограничений. В то же время вычислительные эксперименты показали, что время построения разбиения алгоритма управления с несколькими тысячами вершин может достигать нескольких месяцев [14], что сужает сферу применения параллельно-последовательного метода.

В настоящей статье приведено описание устройства-акселератора, позволяющего произвести перенос операций преобразования *R*-выражений с программного уровня на аппаратный. Указанные операции являются одними из наиболее трудоемких в составе параллельно-последовательного метода как в плане реализации, так и в плане вычислительных затрат. Предлагаемый акселератор может быть использован для операций над конструктивными подмножествами вершин произвольных корректных граф-схем параллельных алгоритмов (не обязательно алгоритмов управления).

2. Табличное представление *R*-выражения в виде дерева

Следуя работам [3–5, 9, 10], в качестве структуры данных, используемой для реализации операций преобразования *R*-выражений, возьмем табличное представление дерева. Узлами подобного дерева являются вершины, хранящие информацию о том, в каком отношении (параллельность или альтернатива) находятся их потомки. Листьями дерева являются вершины ациклической граф-схемы алгоритма управления G^+, входящие в запись *R*-выражения. Узлы дерева характеризуются следующими параметрами: тип узла (параллельный или альтернативный); ссылка на предка – указывает на узел, являющийся предком текущего; ссылки на потомков – указывает на узлы и наборы листьев, являющиеся потомками текущего. Листья дерева характеризуются номером вершины, образующей лист, и ссылкой на узел-предок. Если элемент дерева (под элементом дерева в дальнейшем будем понимать узел или лист) является корневым, у него отсутствует ссылка на предка (точнее, ее значение должно указывать на заведомо несуществующий элемент по аналогии с никуда не указывающими указателями NULL или nil в языках программирования C и Pascal/Delphi соответственно). Аналогично, у листьев по определению нет потомков.

Число листьев, являющихся потомками одного и того же узла, может быть достаточно велико (в первом приближении оно определяется числом параллельных или альтернативных ветвей в составе соответствующих фрагментов алгоритма управления), поэтому из соображений экономии памяти будем рассматривать листья дерева не в виде отдельных вершин, а в виде наборов листьев, составляющие каждого из которых являются потомками одного узла дерева. Экономия памяти при этом достигается за счет уменьшения числа раз,

когда требуется хранить информацию об узле-предке (для каждого узла в отдельности или однократно для набора листьев), а также за счет уменьшения числа элементов дерева (строк таблицы). Кроме того, наборы листьев в такой трактовке фактически представляют собой множества и, как следствие, многие операции с ними могут быть достаточно просто и эффективно реализованы как операции над множествами.

Будем обозначать узлы дерева как T_i^X , а наборы листьев как L_i^X , где i – номер рассматриваемого узла или набора листьев, а X – принадлежность рассматриваемого элемента к дереву X (при дальнейшем описании в операциях, подразумевающих обработку только одного дерева, верхние индексы в некоторых случаях могут быть опущены).

Каждый элемент дерева можно представить в виде совокупности значений следующих полей:

- ТУ (тип узла) – поле, присутствующее у элементов дерева и определяющее тип узла (в случае, если элемент является узлом), или то, что элемент является набором листьев (сокращенно $t\left(T_i^X\right) \in \{\omega, \psi, l\}$, где ω – параллельный тип узла, ψ – альтернативный тип узла, l – узел является набором листьев);
- МВ (множество вершин) – битовый вектор, присутствующий у наборов листьев и определяющий состав вершин, входящих в набор (i-й бит вектора равен 1, если вершина граф-схемы алгоритма a_i входит в состав набора листьев, и 0 в противном случае), в используемых в дальнейшем обозначениях отождествляется с обозначением набора листьев L_i^X ;
- СП (ссылка на предка) – элемент, присутствующий у всех элементов дерева и хранящий номер узла-предка (сокращенно $u\left(T_i^X\right)$ или $u\left(L_i^X\right)$);
- МС (множество ссылок) – битовый вектор, присутствующий у узлов и определяющий состав элементов дерева, являющихся потомками текущего узла.

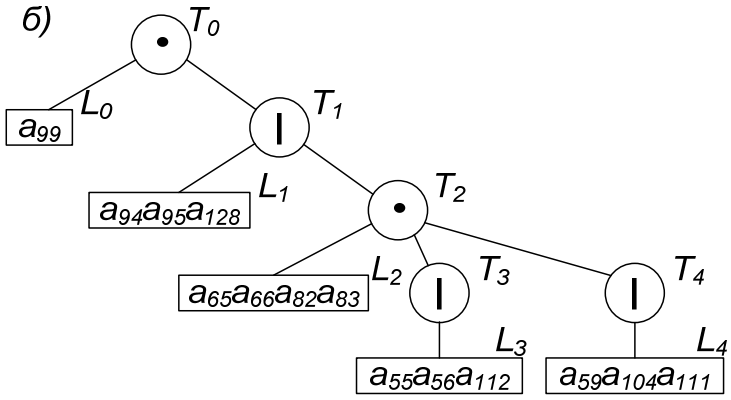
Можно заметить, что поля МВ (у наборов листьев) и МС (у узлов) обладают наибольшим размером, поэтому при хранении узлов и наборов листьев их можно совместить в виде поля МВС (множество вершин/ссылок), трактуемого как поле МВ для наборов листьев и МС для узлов соответственно. Перечисленные выше поля являются основными и используются для хранения исходной структуры дерева. Во время реализации элементарных операций обработки дерева [10] возникает необходимость в наборе дополнительных полей:

- МУ (мощность узла) – поле текущей мощности узла дерева, присутствующее у всех узлов дерева и используемое при вычислении ω-мощности дерева (сокращенно $\omega\left(T_i^X\right)$);
- ТС (тип соответствия) – поле, кодирующее тип соответствия (полное – значение «11», неполное (частичное) – значение «10», соответствия нет – значение «00» или «01», обозначаемое в дальнейшем «0*») элемента дерева и используемое в процессе нахождения r-изоморфного поддерева (сокращенно $\Delta\left(L_i^X\right)$ или $\Delta\left(T_i^X\right)$);
- НС (номер соответствия) – поле, присутствующее у всех элементов дерева и используемое для хранения элемента подстановки изоморфизма (сокращенно $n_r\left(L_i^X\right)$ или $n_r\left(T_i^X\right)$);
- УЭ (удаляемый элемент) – поле-признак того, что данный узел помечен как удаленный из дерева (значение «1») или является действующим компонентом дерева (значение «0») (сокращенно $\chi\left(L_i^X\right)$ или $\chi\left(T_i^X\right)$).

Элементы дерева удобно представлять заданными в табличном виде, i-я строка которой соответствует значениям полей i-го элемента дерева. Пример R-выражения, его представление в виде дерева и таблицы приведен на рис. 1.

а) $a_{99}\bullet(a_{94}|a_{95}|a_{128}|(a_{65}\bullet a_{66}\bullet a_{82}\bullet a_{83}\bullet(a_{55}|a_{56}|a_{112})\bullet(a_{59}|a_{104}|a_{111})))$

б)



в)

№	Тип узла	Ссылка на предка	Множество вершин/ ссылок
0 (T_0)	•	–	1, 2
1 (L_0)	L	0	99
2 (T_1)	|	0	3, 4
3 (L_1)	L	2	94, 95, 128
4 (T_2)	•	2	5, 6, 7
5 (L_2)	L	4	65, 66, 82, 83
6 (T_3)	|	4	8
7 (T_4)	|	4	9
8 (L_3)	L	6	55, 56, 112
9 (L_4)	L	7	59, 104, 111

г)

Узлы

№	Тип узла	Ссылка на предка
0 (T_0)	•	–
1 (T_1)	|	0
2 (T_2)	•	1
3 (T_3)	|	2
4 (T_4)	|	3

Наборы листьев

№	Множество вершин	Ссылка на предка
0 (T_0)	99	0
1 (T_1)	94, 95, 128	1
2 (T_2)	65, 66, 82, 83	2
3 (T_3)	55, 56, 112	3
4 (T_4)	59, 104, 111	4

Рис. 1. Представление *R*-выражения (*а*) в виде дерева (*б*) и различные способы табличного представления (*в*, *г*)

Табличное представление (рис. 1, *в*) было использовано в ранней реализации операций над *R*-выражениями [3, 4]. В нем не делалось различия между узлами и наборами листьев: все они хранились и обрабатывались в виде одной таблицы, а наличие полей МС у узлов и СП у всех элементов дерева позволяло использовать два направления обхода дерева (снизу вверх, т.е. от листьев к корню, и сверху вниз, т.е. от корня к листьям), при этом алгоритмы обработки также использовали оба направления обхода. В более поздней реализации операций [10] было использовано табличное представление (рис. 1, *г*), особенностями которого является раздельное хранение узлов и наборов листьев в виде двух взаимосвязанных таблиц и отсутствие полей МС у узлов дерева, позволяющее существенно экономить память, но в то же время ограничивающее только одним возможным направлением обхода дерева (снизу вверх). Кроме того, модель (см. рис. 1, *г*) позволяет использовать для узлов и наборов листьев различные наборы дополнительных полей при реализации отдельных операций по обработке *R*-выражений, что также ведет к экономии памяти. При работе акселератора используется позднее табличное представление.

Число узлов и наборов листьев в дереве X будем обозначать как $N_T(X)$ и $N_L(X)$ соответственно. Введем также величину числа элементов дерева $N_R(X)=N_T(X)+N_L(X)$, используемую в дальнейшем в некоторых соотношениях, характеризующих

асимптотическую сложность алгоритмов обработки *R*-выражений и аппаратную сложность схем в составе акселератора.

R-выражение будем считать корректно заданным в табличной форме в том случае, если соблюдаются следующие условия.

1) Корень дерева хранится в позиции с номером 0. Значение поля СП корня указывает на заведомо несуществующий элемент дерева $\left(-1_{10} = 11...1_2\right)$.
2) Для каждого узла дерева его потомки хранятся в позициях с номерами, превосходящими номер позиции самого узла (при этом порядок хранения потомков не важен).
3) Все узлы и наборы листьев дерева хранятся в смежных позициях (без «пропусков»).
4) Каждому узлу дерева соответствует не более одного дочернего набора листьев, в дереве не может быть «пустых» наборов листьев, не содержащих вершин.
5) Если дерево представлено единственным набором листьев (без узлов), то в составе этого набора может быть всего один элемент (одна вершина алгоритма управления).
6) В составе корректного дерева не может быть совпадения типа у любой пары смежных узлов.
7) Дерево включает в своем составе, по меньшей мере, один набор листьев. Число узлов дерева может быть нулевым.

Соблюдение перечисленных выше условий позволяет максимально уменьшить время выполнения операций, обеспечить безотказную обработку деревьев любой конфигурации и максимально эффективно использовать память, затрачиваемую на хранение элементов дерева в табличном представлении.

Требование 1 защищает от возможных ложных срабатываний в ситуации, когда *r*-изоморфный эквивалент рассматриваемого узла [9] является корнем дерева и у него по определению отсутствует предок. Требование 2 гарантирует, что при рассмотрении узлов дерева в его табличном представлении в порядке от узлов с большим номером позиции к узлам с меньшим номером будет обеспечен обход дерева по ярусам в направлении снизу вверх. Требование 3, с одной стороны, гарантирует экономное использование позиций табличного представления дерева, а с другой – защищает от ложных срабатываний при попытке обращения к неиспользуемым (пустым) позициям, которые могут быть образованы в ходе операции удаления поддерева и в общем случае могут содержать произвольные двоичные значения. Требование 4 обеспечивает группировку листьев дерева с общим предком в один набор, что упрощает алгоритмы их обработки и повышает быстродействие. Требование 5 продиктовано невозможностью присутствия в наборе двух и более вершин без установления отношения параллельности или альтернативы между ними (отношение между вершинами в таком случае устанавливается типом узла – предка набора листьев). Несоблюдение требования 6 может привести к невозможности нахождение множества сечений для некоторых корректных алгоритмов управления. В случае наличия подобной пары узлов с совпадающим типом требуется раскрытие скобок (например, $a_1 \mid \left(a_2 \bullet \left(a_3 \bullet a_4 \bullet a_5\right)\right) \Rightarrow a_1 \mid \left(a_2 \bullet a_3 \bullet a_4 \bullet a_5\right)$), в противном случае в некоторых случаях невозможно нахождение базового сечения для корректных алгоритмов управления. Требование 7 рассматривает некоторые частные случаи деревьев, которые находят применение в некоторых алгоритмах обработки.

Правила *u*- и *d*-поглощения [2–4, 10], используемые при поиске базового сечения, в совокупности с действиями, применяемыми при последующем построении множества смежных сечений, допускают разбиение на ряд более простых (элементарных) операций над *R*-выражениями. К их числу относятся:

- проверка принадлежности вершины a_i *R*-выражению R_j;
- вычисление ω-мощности *R*-выражения;
- проверка отношения нестрогого включения $R_i \left[\subseteq\right] R_j$ (поиск *r*-изоморфного поддерева и подстановки изоморфизма);

- удаление поддерева;
- вставка поддерева;
- раскрытие скобок в *R*-выражениях;
- удаление «лишних» элементов из дерева (пропусков).

Каждой элементарной операции соответствует отдельная схема в составе акселератора, для хранения значений перечисленных полей (СП, ТУ и т.д.) элементов дерева используются наборы триггеров.

Как уже было отмечено выше, аппаратно-ориентированные алгоритмы обработки *R*-выражений синтезированы таким образом, что позволяют обойтись лишь одним направлением обхода дерева (снизу вверх), в то время как при ранней программной реализации были использованы оба направления (обход дерева сверху вниз может быть использован, например, при рекуррентном сравнении деревьев). Это обуславливает основное отличие аппаратно-ориентированного табличного представления *R*-выражения, построенного в тесной взаимосвязи с аппаратно-ориентированными алгоритмами, от ориентированного на программную реализацию. Оно заключается в разделении всех узлов дерева по типу на две группы, хранимые отдельно и кодируемые различными подмножествами полей. В *первую группу* входят параллельные и альтернативные узлы (далее просто узлы), во *вторую* – наборы листьев. Подобное разделение, с одной стороны, обусловлено спецификой операций, выполняемых над узлами различного типа, а с другой позволяет обойтись без поля со множеством ссылок на потомков у узлов, что ведет к существенной экономии количества триггеров (поле со ссылками на потомков имеет существенно больший размер по сравнению с остальными полями). Совместное хранение листьев в виде наборов (битовых векторов) с одной стороны обеспечивает уменьшение аппаратных затрат (хранится весь набор, а не его компоненты по отдельности) за счет уменьшения дублирования полей (например, СП), а с другой – ведет к увеличению быстродействия устройства, т.к. компоненты битовых векторов могут быть обработаны параллельно во времени, а отдельные элементы дерева – последовательно.

Совокупность $N_T(X)$ узлов и $N_L(X)$ наборов листьев образуют дерево, при этом для обозначения текущего количества узлов и наборов листьев в состав дерева X также входят регистры текущего количества узлов (ТКУ) и текущего количества листьев (ТКЛ).

Разрядности полей приведены на рис. 2.

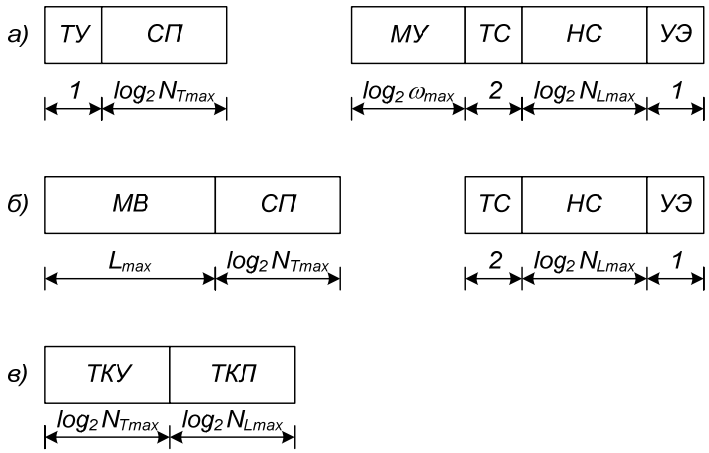


Рис. 2. Состав и размеры полей при кодировании одного узла дерева (а), одного набора листьев (б) и полей ТКУ и ТКЛ дерева в целом (в)

Здесь $N_{T_{\max}}$ – максимальное количество узлов в дереве, $N_{L_{\max}}$ – максимальное количество наборов листьев в дереве, $L_{\max}$ – максимальное количество листьев в наборе (фактически, максимальное количество вершин в обрабатываемой граф-схеме алгоритма G, т.е. $L_{\max} = N$),

ω_{max} – степень параллелизма граф-схемы алгоритма. На рис. 2 слева представлены поля, обязательные для хранения дерева, а справа – вспомогательные поля, используемые отдельными операциями во время преобразований.

3. Акселератор для быстрой аппаратно-ориентированной обработки *R*-выражений

Структурно-функциональная организация комбинаторно-логического акселератора, синтезированного для выполнения аппаратно-ориентированной обработки *R*-выражений, приведена на рис. 3.

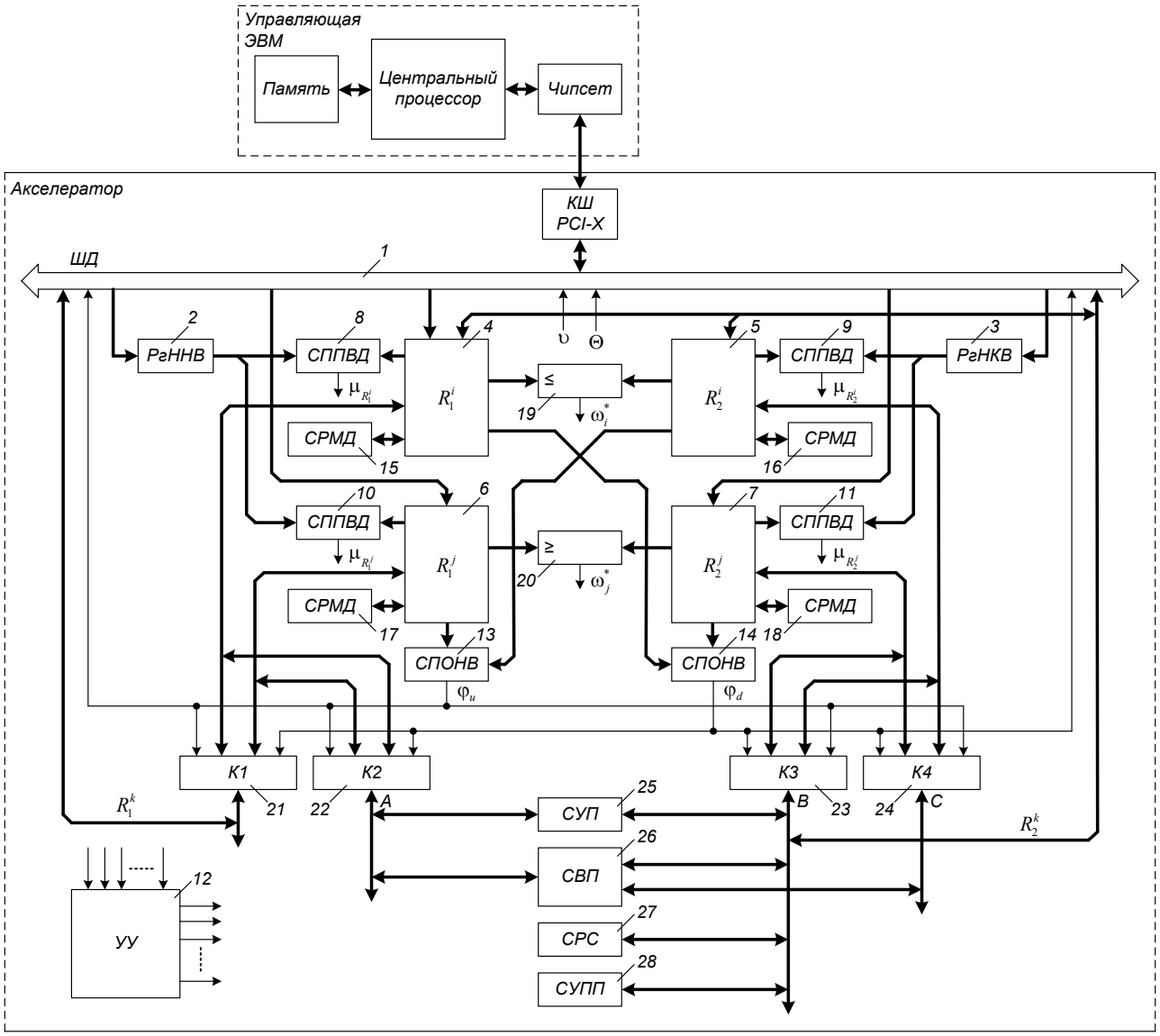


Рис. 3. Структурная схема комбинаторно-логического акселератора для аппаратно-ориентированной обработки *R*-выражений

Рассмотрим сперва работу акселератора в режиме нахождения базового сечения. Обмен информацией между акселератором и управляющей ЭВМ верхнего уровня производится с использованием шины данных 1 и контроллера шины PCI Express. На этапе инициализации акселератора в регистры номера начальной вершины (РгННВ) 2 и номера конечной вершины (РгНКВ) 3 производится загрузка номеров начальной и конечной вершин граф-схемы

алгоритма управления, разбиение которого необходимо синтезировать. Далее в элементы 4–7 производится загрузка обрабатываемой пары выражений системы Ξ

$$\begin{cases} S_i\colon\ R_1^i \to R_2^i \\ S_j\colon\ R_1^j \to R_2^j \end{cases}$$

в виде набора полей, кодирующих соответствующие *R*-выражения. Непосредственно после завершения загрузки начинают свою работу схемы проверки принадлежности вершины дереву (СППВД) 8–11 (являющиеся по своему принципу работы комбинационными), в результате чего оказываются сформированными двоичные признаки $\mu_{R_1^i}$, $\mu_{R_2^i}$, $\mu_{R_1^j}$ и $\mu_{R_2^j}$ принадлежности начальной или конечной вершин к обрабатываемым *R*-выражениям. На основании полученных значений двоичных признаков устройство управления (УУ) 12 либо разрешает дальнейшие операции над загруженной в акселератор парой выражений, либо выдает нулевое значение признака $\upsilon = 0$, свидетельствующего о невозможности выполнения операции подстановки, и единичное значение признака $\Theta = 1$ окончания обработки (с целью упрощения линии связи устройства управления со схемами отдельных преобразований на рис. 3 не показаны). На следующем этапе работы акселератора активируются схемы проверки отношения нестрогого включения (СПОНВ) 13 и 14, проверяющие наличие *r*-изоморфизма у обрабатываемых пар *R*-выражений. Результатом проверки являются значения признаков φ_u и φ_d возможности проведения *u*- или *d*-подстановки. Параллельно с выяснением *r*-изоморфизма с использованием схем расчета мощности дерева (СРМД) 15–18 производится расчет ω-мощностей обрабатываемых *R*-выражений. Рассчитанные значения ω-мощностей *R*-выражений подаются на входы схем сравнения 19 и 20, на выходах которых формируются двоичные признаки ω_i^* и ω_j^* соблюдения ограничений на ω-мощность обрабатываемых выражений. По полученным значениям признаков φ_u, φ_d, ω_i^* и ω_j^* устройство управления либо выдает признаки $\upsilon = 0$ и $\Theta = 1$, свидетельствующие о невозможности проведения операции подстановки (обработка пары выражений S_i и S_j на этом прекращается), в противном случае обработка *R*-выражений продолжается. При этом на выходе коммутатора К1 формируется значение R_1^k левой части результирующего выражения S_k:

$$R_1^k = R_1^i \varphi_u \vee R_1^j \varphi_d$$

(R_1^i в случае активации выполнения *u*-подстановки и R_1^j – в случае *d*-подстановки). На выходах коммутаторов К2–К4 формируются значения полей удаляемого $\left(A^R\right)$, объемлющего $\left(B^R\right)$ и подставляемого $\left(C^R\right)$ деревьев соответственно в зависимости от типа подстановки:

$$A = R_1^j \varphi_u \vee R_1^i \varphi_d,$$
$$B = R_2^i \varphi_u \vee R_2^j \varphi_d,$$
$$C = R_2^j \varphi_u \vee R_2^i \varphi_d.$$

Далее устройством управления поочередно активируются схемы удаления поддерева (СУП) 25, вставки поддерева (СВП) 26, раскрытия скобок (СРС) 27 и удаления пустых позиций (СУПП) 28, в результате чего оказывается выполненной операция подстановки и дерево B^R теперь является частью результирующего выражения S_k.

По завершении обработки устройство управления устанавливает признаки $\upsilon = 1$ и $\Theta = 1$, затем производится выдача на шину 1 значений полей сформированных в ходе подстановки *R*-выражений R_1^k и R_2^k.

Режим построения множества смежных сечений состоит из двух фаз. В первой фазе осуществляется поиск множества выражений $\{S_{j_1}, S_{j_2}, ..., S_{j_m}\}$ для осуществления подстановки в текущее сечение Ω_i. Для этого в элементы 4 и 5 акселератора загружается текущее сечение Ω_i, а в элементы 6 и 7 – левая R_1^j и правая R_2^j части обрабатываемого выражения S_j соответственно. Далее активируются схемы СПОНВ 13 и 14, на выходах которых формируются значения признаков φ_u и φ_d возможности проведения *u*- или *d*-подстановки. Сформированные значения признаков выдаются на шину 1, признак завершения обработки $\Theta = 1$ устанавливается в единицу. В результате подобной обработки всех выражений S_j системы Ξ управляющая машина накапливает информацию о том, какие выражения системы Ξ необходимо подставить в текущее сечение Ω_i с целью получения *u*- или *d*-сечения. На этом первая фаза обработки завершается.

Во второй фазе в элементы 6 и 7 осуществляется загрузка выражения S_{j_1}, отобранного в предыдущей фазе как подходящего для подстановки (в элементах 4 и 5 загружено текущее сечение Ω_i). Далее активируются схемы СПОНВ 13 и 14, целью работы которых являются не столько значения признаков φ_u и φ_d, сколько подстановки изоморфизма, формируемой в полях НС элементов подставляемого дерева. Далее, аналогично рассмотренному выше, на выходах коммутаторов К2–К4 формируются значения деревьев A^R, B^R и C^R, используемые последовательно активируемыми схемами 25–28, в результате чего дерево B^R представляет собой результат операции подстановки. По сигналу готовности $\Theta = 1$ дерево B^R передается в элементы 4 и 5, элементы 6 и 7 принимают от управляющей машины следующее подходящее выражение S_{j_k} и описанная выше вторая фаза работы акселератора повторяется. В результате обработки всего множества выражений $\{S_{j_1}, S_{j_2}, ..., S_{j_m}\}$ на выходе коммутатора К3 формируется искомое смежное сечение, которое в виде набора полей выдается на шину 1.

Наиболее простым способом использования предлагаемого акселератора является последовательное выполнение операций подстановки при поиске базового сечения и затем построении множества смежных сечений. При необходимости возможно объединение акселераторов в линейку или кольцо для организации параллельной обработки *R*-выражений (например, при параллельном выяснении *r*-изоморфизма *i*-го выражения системы Ξ со всеми остальными, используемом как при поиске базового сечения, так и при построении множества смежных сечений). Также возможно объединение акселераторов в матрицу для организации попарного параллельного выяснения отношения *r*-изоморфизма между *R*-выражениями системы Ξ, имеющего место при отыскании базового сечения.

4. Однородная среда для хранения табличного представления *R*-выражения в виде дерева

Для хранения перечисленных выше полей элементов табличного представления дерева будем использовать однородную среду со следующей структурой ячейки [15]:

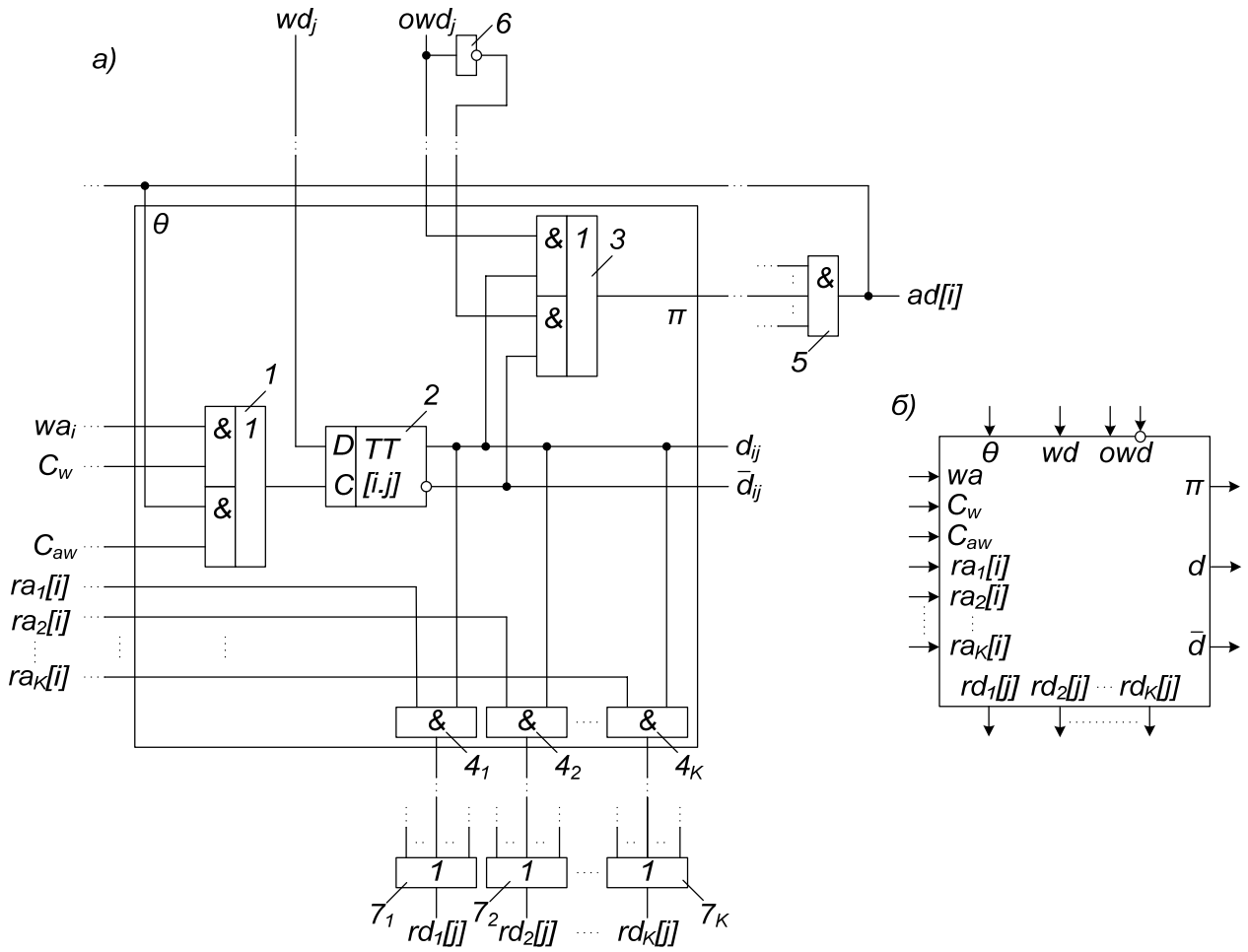


Рис. 4. Ячейка однородной среды электронной модели дерева (ЭМД) (а) и ее схематичное обозначение (б)

Однородная среда, состоящая из $N \times M$ ячеек одинаковой структуры, обеспечивает следующие возможности: хранение информации, параллельное чтение данных из нескольких портов, запись данных с использованием порта записи, ассоциативный поиск информации, ассоциативную запись данных одновременно в несколько строк. Рассмотрим возможности среды более подробно.

Чтение данных из ячейки осуществляется следующим образом. На входы ячеек $ra_1[i], i = \overline{1, M}$ (подразумевается чтение из первого порта, *ra* – сокращение от Read Address, адрес чтения) подается битовый вектор «00…0100…0», в котором единица в *i*-й позиции обозначает необходимую для считывания *i*-ю строку. На выходах элементов И 4_k, $k = \overline{1, K}$, *K* – число портов чтения, *j*-го столбца $\left(j = \overline{1, N}\right)$ во всех ячейках, за исключением *i*-ой, будет сигнал логического нуля, а на выходе элемента И 4_1 *i*-ой строки (в рассматриваемом случае) – требуемое значение *j*-го бита *i*-ой строки, которое появится на выходе *j*-го элемента ИЛИ 7_1, коммутирующего столбец ячеек.

Несложно заметить, что одновременно возможно чтение нескольких строк однородной среды. Для этого необходимо подать битовые вектора «00…0100…0» с единицами в требуемых позициях на входы различных портов $ra_k, k = \overline{1, K}$ и получить требуемые данные на выходах портов rd_k (*rd* – сокращение от Read Data, данные чтения).

В случае, если на входе ra_k вектор будет содержать несколько единиц, на выходе rd_k будет сформирована дизъюнкция данных строк, отмеченных единицами. Подобная особенность используется, например, при нахождении множества всех листьев дерева $\bigcup_i L_i^X$.

Также предусмотрена возможность непосредственного чтения данных всего массива ячеек сразу (либо некоторой его части) благодаря наличию «прямых» (без какой-либо коммутирующей логики) выходов d_{ij} и $\overline{d}_{ij}$ от триггеров ячеек.

Запись данных в ячейку производится путем подачи битового вектора «00…0100…0» на входы $wa[i]$ ячеек среды (сокращение от Write Address, адрес записи), причем единица в *i*-ой позиции, также как и при операции чтения, выбирает строку, в которую производится запись. На вход $wd[j]$ (сокращение от Write Data, данные записи) подаются данные, которые требуется сохранить в ячейках среды. Запись производится в момент появления высокого уровня синхросигнала C_w, причем синхросигнал проходит на синхровходы *C* триггеров 2 ячеек только для выбранной *i*-ой строки благодаря коммутатору 1.

Во время выполнения операций преобразования *R*-выражений возможна ситуация, когда входные данные *wd* являются логической функцией выходных данных среды *rd*. Во избежание временных гонок в ячейках среды необходимо использовать синхронные двухступенчатые *D*-триггеры 2, причем запись в первую ступень триггера производится по высокому уровню синхросигнала *C*, а во вторую – по низкому.

Теоретически возможна организация одновременной параллельной записи в разные строки среды путем создания нескольких портов записи (по аналогии с портами чтения), однако в разрабатываемом устройстве такая функциональность не требуется, что позволяет снизить аппаратную сложность ячейки.

Запись данных может быть совмещена во времени с чтением, но только в том случае, если читаются и записываются различные строки среды. Контроль подобных коллизий средой не осуществляется.

При *ассоциативном поиске информации* на входы $owd[j]$ (сокращение от Old Written Data, предыдущие хранимые данные) ячеек подаются искомые данные. При этом совпадение поданных данных с хранимыми в ячейке идентифицируется единичным значением признаки совпадения π на выходе коммутатора 3. В случае совпадения данных по всей строке на выходе элемента И 5, коммутирующего сигналы π ячеек строки, формируется единичное значение признака θ, которое поступает на выход $ad[i]$ среды (сокращение от Associative Data, данные ассоциативного поиска) и идентифицирует совпадение. Очевидно, что подобный поиск проводится параллельно по всем строкам среды.

При *ассоциативной записи данных* производится сохранение информации, подаваемой на входы $wd[j]$, одновременно в нескольких строках среды. Строки, в которые осуществляется запись, определяются по совпадению хранящейся в них информации с данными, подаваемыми на вход $owd[j]$. В случае подобного совпадения, аналогично рассмотренному выше ассоциативному поиску, на выходах элементов И 5 строки формируется единичное значения признака θ, разрешающего запись данных входа *wd* по синхросигналу C_{aw} (*aw* – сокращение от Associative Write, ассоциативная запись), успешно проходящему в таком случае через коммутатор 1. Если хотя бы в одной ячейке строки признак π будет иметь нулевое значение (сохраненное в ячейке значение не совпадает с поданным на вход *owd*), признак θ также получит нулевое значение и синхросигнал C_{aw} не пройдет на синхровход *C* триггера 2 ячейки через коммутатор 1: запись данных в строку не произойдет.

Ячейки объединяются в среду, как показано на рис. 5.

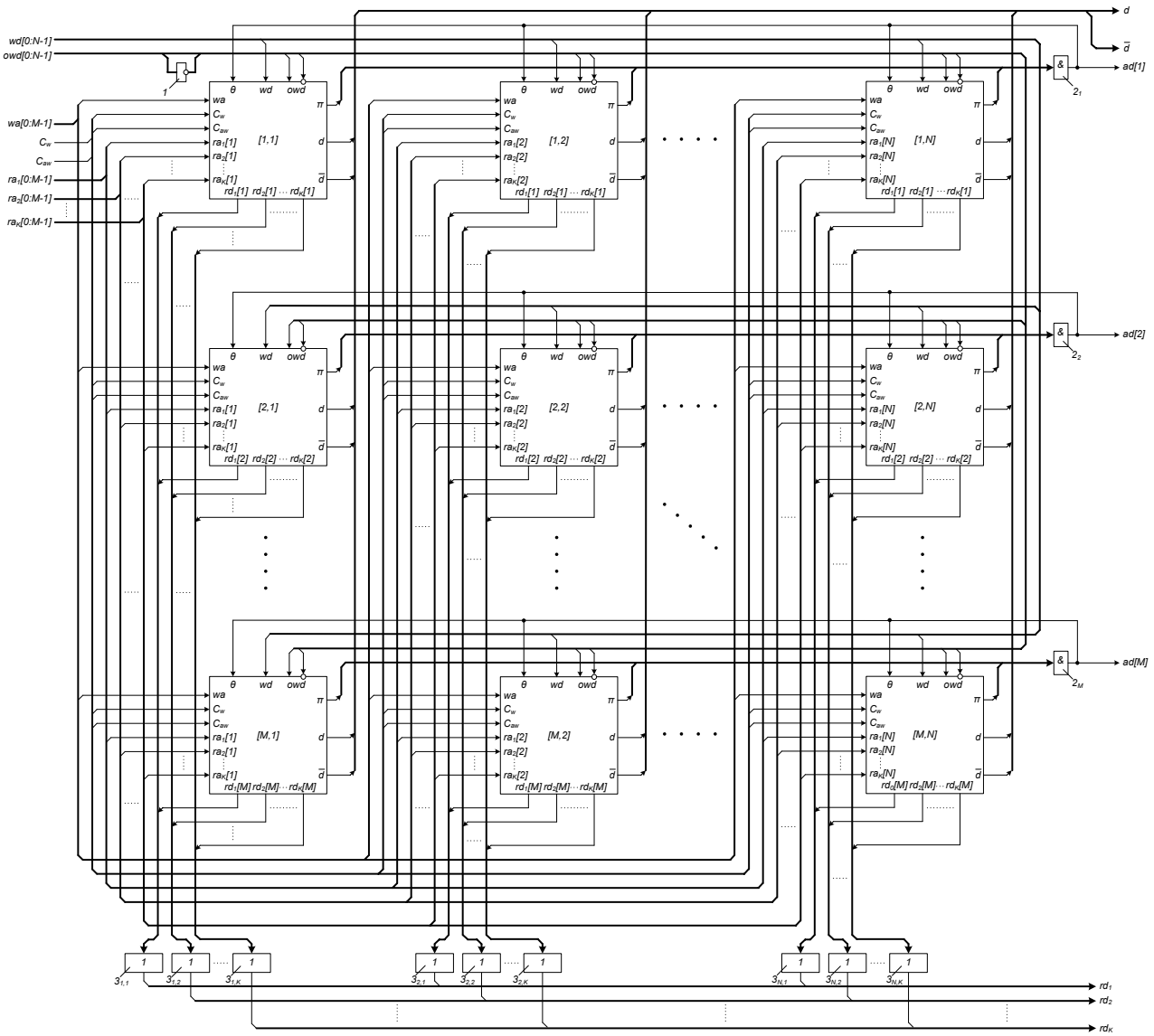


Рис. 5. Однородная среда ЭМД (ОСЭМД)

Сокращенное обозначение среды, используемое в рассматриваемых ниже схемах, приведено на рис. 6.

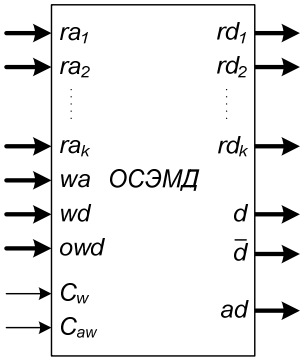


Рис. 6. Условное обозначение ОСЭМД

Каждая строка однородной среды электронной модели дерева призвана хранить поля одного элемента дерева, однако в большинстве случаев более целесообразно рассматривать несколько совместно работающих ОСЭМД, каждая из которых хранит одно из полей узла (ТУ, МВ и т.д.). Совместная работа ОСЭМД обеспечивает возможность совместного

параллельного чтения *i*-х строк сред (фактически, значений полей *i*-го элемента дерева), при этом сохраняется возможность раздельной записи измененных значений полей и достигается упрощение структуры ячеек некоторых ОСЭМД (например, не для всех полей требуется возможность ассоциативной записи, многопортового чтения или необходимо применение двухступенчатых триггеров). Таким образом, представленную на рис. 4 схему ячейки ОСЭМД можно считать наиболее общей, причем в некоторых случаях она допускает упрощение.

5. Загрузка данных в ОСЭМД

Первым этапом функционирования акселератора является получение исходных данных в виде пары выражений системы Ξ (от управляющей ЭВМ, как показано на рис. 3, или другого блока устройства). Система выражений Ξ фактически представляет собой набор *R*-выражений, которые можно загружать последовательно (передача от управляющей ЭВМ), параллельно (передача от другого блока, хранящего электронную модель графа алгоритма управления, подлежащего разбиению) либо параллельно-последовательно (компромиссный вариант). Рассмотрим загрузку данных одного из деревьев в ОСЭМД более подробно.

Выполнение этой операции целиком в параллельной форме затруднительно, т.к. для этого потребовалось бы очень большое количество связей, определяемое суммарным размером всех полей в битах:

$$n = O\left(N_T\left(1+\log_2 N_T+1\right)+N_L\left(L_{\max}+\log_2 N_T+1\right)\right)\simeq$$
$$\simeq O\left(N_T \log N_T + N_L L_{\max}\right). \qquad (\Phi 4)$$

Для алгоритмов размером $100-1000$ вершин число связей составило бы величину порядка 10^4-10^5, что неприемлемо. Поэтому выполнение загрузки данных будем осуществлять построчно (поэлементно с точки зрения компонентов дерева). При этом полагается, что данные одной строки ОСЭМД сформированы на входящей шине данных (ШД) одним из перечисленных выше способов (рис. 7). Здесь и далее с целью упрощения структуры схем неиспользуемые входы ОСЭМД не показаны.

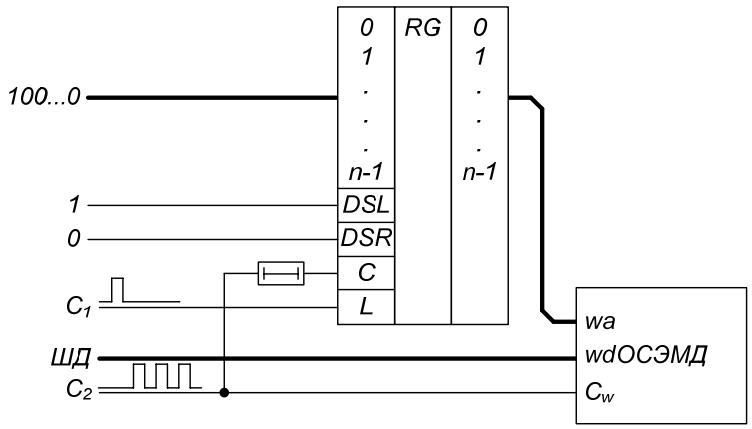


Рис. 7. Схема загрузки данных в ОСЭМД

В процессе инициализации по синхросигналу C_1 производится запись единицы в младшую позицию сдвигового регистра. Выход регистра соединен со входом адреса записи ОСЭМД, что обеспечивает поочередную выборку строк среды с целью записи информации в процессе продвижения единицы по регистру. При появлении синхросигнала C_2 производится запись данных шины ШД в ОСЭМД и, спустя интервал времени, генерируемый элементом задержки, сдвиг содержимого регистра на одну позицию влево, т.е.

подготовку к записи следующей строки. После появления последовательности синхросигналов C_2 ОСЭМД содержит информацию обо всех элементах дерева.

Процесс загрузки данных на рис. 7 представлен в обобщенной форме. С учетом того, что *R*-выражение представляется в виде двух отдельных компонент (узлов и наборов листьев), возможна организация как последовательного, так и параллельного заполнения этих компонент во времени.

6. Комбинационные схемы формирования и преобразования битовых последовательностей

На некоторых этапах функционирования ускорителя возникает необходимость в исключении из рассмотрения части элементов дерева (например, тех, которые являются в данный момент свободными), выборе первой свободной или последней занятой позиции таблицы и т.д. Подобные действия вполне можно организовать алгоритмически (как это реализовано при программной реализации) путем сканирования элементов таблицы в прямом или обратном порядке, однако также возможна схемотехническая реализация в виде комбинационных схем [16]. Второй подход при аппаратной реализации представляется более интересным, т.к. допускает более быстрое получение результата. Принципы построения предлагаемых схемы рассмотрены ниже.

Отметим, что построение любой из приведенных ниже схем возможно несколькими способами: непосредственно по картам Карно (наивысшее быстродействие и аппаратная сложность), с использованием ПЛМ, с использованием алгоритмических особенностей выполняемых действий (например, последовательное распространение признаков переноса, обеспечивающее сравнительно низкое быстродействие в совокупности с низкой аппаратной сложностью) или компромиссный между рассмотренными выше вариант (например, последовательное распространение признаков переноса между группами разрядов, обрабатываемыми параллельно, обеспечивающее компромиссный вариант по быстродействию и аппаратной сложности).

6.1. Схема подсчета бит (СПБ)

Исходными данными для схемы выступает двоичный вектор *X*. В результате обработки на выходе схемы должно быть сформировано двоичное значение, равное числу бит вектора *X*, установленных в единицу. На рис. 8–10 приведены таблицы истинности и карты Карно для искомых схем с различной разрядностью вектора *X*. Видно, что с ростом разрядности вектора *X* сложность схемы растет, однако время задержки получения результата остается постоянным и составляет $3t_0$, где t_0 – элементарное время задержки логического вентиля. Подобным образом теоретически можно синтезировать схему для любой разрядности вектора *X*.

x_1	x_2	f_2	f_1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$f_2 = x_1 x_2$ $\quad f_1 = x_1 \overline{x}_2 \vee \overline{x}_1 x_2 = x_1 \oplus x_2$

x_1 \ x_2	0	1
0	0	0
1	0	1

x_1 \ x_2	0	1
0	0	1
1	1	0

Рис. 8. Таблица истинности, карты Карно и формулы выходов схемы СПБ при количестве входов $n = 2$

x_1	x_2	x_3	f_2	f_1
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$f_2 = x_1x_3 \vee x_2x_3 \vee x_1x_2$$

x_1 \ x_2x_3	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$f_1 = x_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2x_3 \vee x_1x_2x_3 \vee \overline{x}_1x_2\overline{x}_3 = x_1 \oplus x_2 \oplus x_3$$

x_1 \ x_2x_3	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Рис. 9. Таблица истинности, карты Карно и формулы выходов схемы СПБ при количестве входов $n = 3$

x_1	x_2	x_3	x_4	f_3	f_2	f_1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

$$f_3 = x_1x_2x_3x_4$$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

$$f_2 = x_1x_2\overline{x}_3 \vee x_1\overline{x}_2x_3 \vee x_1\overline{x}_2x_4 \vee x_1x_3\overline{x}_4 \vee \overline{x}_1x_3x_4 \vee x_2\overline{x}_3x_4$$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	0	1
10	0	1	1	1

$$f_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Рис. 10. Таблица истинности, карты Карно и формулы выходов схемы СПБ при количестве входов $n = 4$

Из общих закономерностей, подмеченных в ходе анализа рис. 8–10, можно отметить лишь формулу для младшего разряда результата:

$$f_1 = x_1 \oplus x_2 \oplus ... \oplus x_n .$$

Формулы остальных разрядов, по видимому, выводятся для каждого случая индивидуально.

В [17] синтезируемая схема подсчета бит именуется «цифровым компрессором», а в [18] предлагается ее пирамидальный вариант (рис. 11).

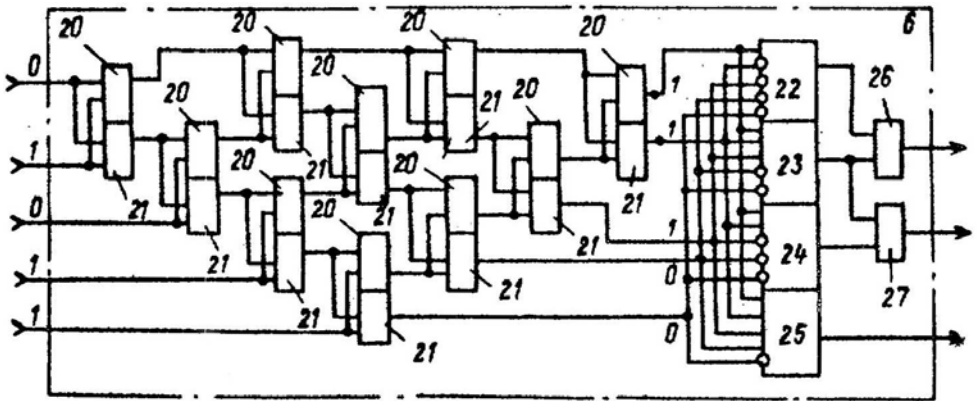


Рис. 11. Пирамидальный вариант схемы подсчета бит (под номером 20 – элементы ИЛИ, 21 – И, 22–25 – элементы запрета, 26 и 27 – ИЛИ)

Принцип действия схемы достаточно прост. Пирамидальная часть схемы, именуемая в дальнейшем схемой упорядочения двоичного вектора (СУДВ), образована элементами ИЛИ 20 и И 21. Она производит упорядочение («компандирование») исходного вектора таким образом, что в одной его части оказываются единичные значения, а в другой – нулевые (например, двоичный вектор «01101011011» будет преобразован в вектор «11111110000»). Это обеспечивается благодаря парам элементов ИЛИ и И (рис. 12), фактически производящим обмен соседних разрядов вектора местами в случае, если $x_1 < x_2$. Указанные пары элементов ИЛИ и И образуют пирамидальную структуру схемы, которая обеспечивает попарное сравнение и, при необходимости, обмен всех бит исходного вектора. В результате происходит перемещение единичных значений в один конец вектора, а нулевых – в другой. Похожий принцип попарных сравнений соседних элементов используется в широко известном алгоритме пузырьковой сортировки массивов [19, 20].

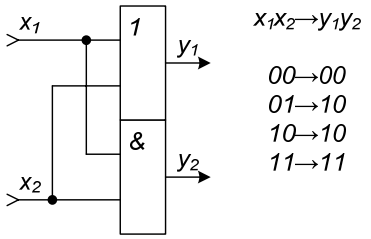


Рис. 12. Элементы ИЛИ и И, производящие упорядочение пары битовых значений

Далее обработкой полученного вектора занимается комбинационная часть схемы, образованная элементами 22–27. За счет того, что в таблицах истинности синтезируемых функций результата встречается большое количество запрещенных комбинаций (см. рис. 13–15), которые не могут появиться на входах комбинационной части схемы благодаря схеме СУДВ, на картах Карно присутствует большое количество звездочек, что уменьшает число термов в синтезируемых функциях и, соответственно, аппаратную сложность комбинационной части схемы по сравнению с приведенным выше вариантом реализации «в лоб» (рис. 8–10).

x_1	x_2	f_2	f_1
0	0	0	0
0	1	*	*
1	0	0	1
1	1	1	0

$f_2 = x_2$

x_1 \ x_2	0	1
0	0	*
1	0	1

$f_1 = x_1\overline{x}_2$

x_1 \ x_2	0	1
0	0	*
1	1	0

Рис. 13. Таблица истинности, карты Карно и формулы выходов комбинационной части пирамидальной схемы СПБ при количестве входов $n = 2$

x_1	x_2	x_3	f_2	f_1
0	0	0	0	0
0	0	1	*	*
0	1	0	*	*
0	1	1	*	*
1	0	0	0	1
1	0	1	*	*
1	1	0	1	0
1	1	1	1	1

$f_2 = x_2$

x_1 \ x_2x_3	00	01	11	10
0	0	*	*	*
1	0	*	1	1

$f_1 = x_3 \vee x_1\overline{x}_2 = x_1 \oplus x_2 \oplus x_3$

x_1 \ x_2x_3	00	01	11	10
0	0	*	*	*
1	1	*	1	0

Рис. 14. Таблица истинности, карты Карно и формулы выходов комбинационной части пирамидальной схемы СПБ при количестве входов $n = 3$

x_1	x_2	x_3	x_4	f_3	f_2	f_1
0	0	0	0	0	0	0
0	0	0	1	*	*	*
0	0	1	0	*	*	*
0	0	1	1	*	*	*
0	1	0	0	*	*	*
0	1	0	1	*	*	*
0	1	1	0	*	*	*
0	1	1	1	*	*	*
1	0	0	0	0	0	1
1	0	0	1	*	*	*
1	0	1	0	*	*	*
1	0	1	1	*	*	*
1	1	0	0	0	1	0
1	1	0	1	*	*	*
1	1	1	0	0	1	1
1	1	1	1	1	0	0

$f_3 = x_4$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	*	*	*
01	*	*	*	*
11	0	*	1	0
10	0	*	*	*

$f_2 = x_2\overline{x}_4$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	*	*	*
01	*	*	*	*
11	1	*	0	1
10	0	*	*	*

$f_1 = x_1\overline{x}_2 \vee x_3\overline{x}_4 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	*	*	*
01	*	*	*	*
11	0	*	0	1
10	1	*	*	*

Рис. 15. Таблица истинности, карты Карно и формулы выходов комбинационной части пирамидальной схемы СПБ при количестве входов $n = 4$

Время задержки получения результата пирамидального варианта схемы подсчета бит зависит от размерности входного вектора и составляет величину $(n+2)t_0$.

Для каждого значения *n* результирующие формулы выходов комбинационной части схемы, по видимому, индивидуальны. Схема СУДВ включает в себя $2\cdot((n-1)+(n-2)+...+1) = 2\cdot\frac{n-1+1}{2}\cdot(n-1) = n(n-1)$ двухвходовых элементов И и ИЛИ.

Таким образом, пирамидальный вариант схемы подсчета бит [18] характеризуется большим временем задержки получения результата при меньшей аппаратной сложности по сравнению с реализацией «в лоб».

Модификацией предложенного выше способа реализации схемы подсчета бит является использование вместо комбинационной части схемы совокупности схемы выделения старшей единицы (СВСЕ, см. ниже) и шифратора (в составе схемы СВСЕ можно обойтись без цепочки элементов ИЛИ, т.к. двоичный вектор на ее входе уже представлен в виде «11…100…0» и сочетание входных сигналов «10» на входах элементов запрета может встретиться только однократно). На выходе схемы СВСЕ в унитарном коде формируется вектор из нулей и единицы в *i*-ой позиции, где *i* – искомое число единичных бит в исходном векторе. Шифратор переводит полученное значение в двоичный код.

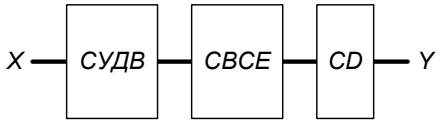


Рис. 16. Вариант схемы подсчета бит с использованием схемы выделения старшей единицы

Также возможен вариант построения схемы с использованием пирамиды сумматоров (рис. 17). Похожим способом реализуется подсчет бит двоичного вектора при программной реализации указанной операции с использованием векторных расширений современных процессоров [21].

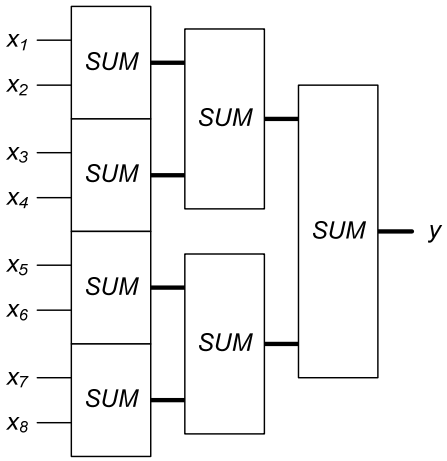


Рис. 17. Вариант схемы подсчета бит с использованием пирамиды сумматоров для случая n=8

Сумматоры первого яруса (на рис. 17 расположены слева) имеют разрядность входов 1 бит, второго яруса – два бита, третьего – три бита и т.д. Время задержки получения результата и аппаратная сложность схемы целиком и полностью определяются параметрами используемых при этом сумматоров.

6.2. Схема маскировки неиспользуемых позиций (СМНП)

Исходными данными для схемы СМНП является число *X*, представленное в двоичном коде и определяющее количество единичных бит в результирующем двоичном векторе на выходе схемы, причем единичные значения группируются в начальных (младших) позициях результирующего вектора. Например, если на вход схемы подан вектор «000101» (5 в десятичной форме), то на ее выходе должен быть сформирован вектор «1111100…0». Разрядность результирующего вектора *n* определяется потребностями более сложных схем, использующих схему СМНП в качестве составной части. Свое название схема получила из-за специфики применения: при необходимости можно «маскировать» (сбросить в ноль) часть бит двоичного вектора *Y* путем поразрядной конъюнкции его элементов y_i с выходами

схемы СМНП f_i . Подобным образом, например, во избежание ложных срабатываний исключаются из рассмотрения неиспользуемые наборы листьев и узлы дерева в его табличном представлении.

Обобщенный вариант реализации схемы, в полной мере отражающий логику ее работы, приведен на рис. 18.

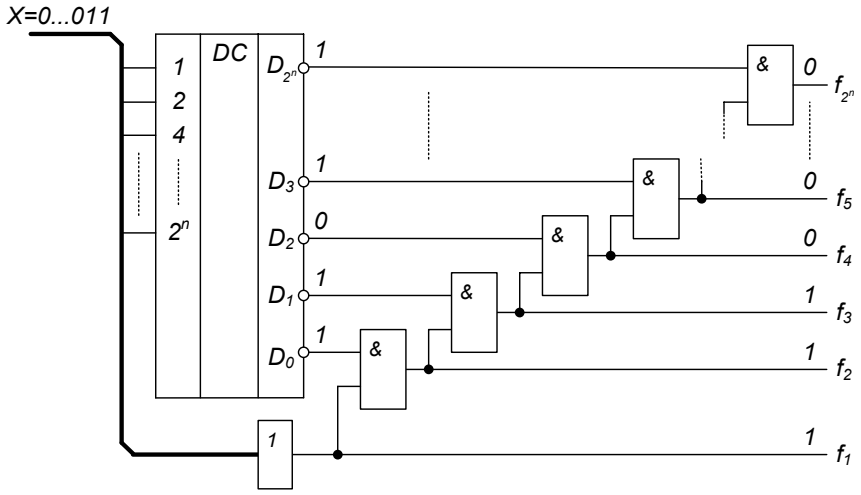


Рис. 18. Обобщенный вариант реализации схемы СМНП

Принцип работы схемы заключается в следующем. Соединенные в цепочку элементы И обеспечивают на выходе *i*-го элемента единичное значение только в том случае, если значение текущего (*i*-го) и всех предыдущих (младших) выходов дешифратора равны единице. Нулевое значение на *i*-м выходе дешифратора обеспечивает нулевое значение на выходе *i*-го и всех последующих элементов И. Для того, чтобы число единичных бит было равно десятичному представлению m двоичного вектора X, а не $m-1$, в схему введен элемент ИЛИ, выход которого является младшим разрядом выходного вектора. Нулевое значение на выходе элемента ИЛИ возможно только в том случае, если все разряды вектора X равны нулю, т.е. $n=0$. Во всех остальных случаях значение на выходе элемента ИЛИ равно единице.

Рассмотренный вариант схемы СМНП является общим. В каждом конкретном случае для заданного значения n схема допускает упрощение, которое можно синтезировать с использованием карт Карно. Пример подобного упрощения для случая $n=8$ приведен на рис. 19.

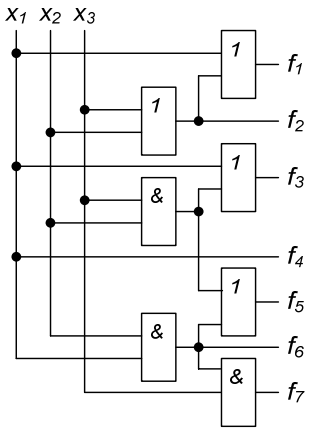


Рис. 19. Пример реализации схемы СМНП в виде МДНФ значений выходного вектора

Приведенная на рис. 19 схема обладает гораздо меньшей аппаратной сложностью по сравнению со схемой на рис. 18 и характеризуется временем задержки получения результата $2t_0$, не зависящим от n.

В случае необходимости синтеза вектора, в котором набор из m единиц должен быть расположен в конечных (старших) позициях, схемы, приведенные на рис. 18 и 19, остаются прежними, производится лишь перенумерация выходов: первый выход становится последним, второй – предпоследним и т.д.

По сути схема СМНП реализует функцию, обратную по отношению к преобразованию, выполняемому совокупностью схемы выделения старшей единицы (СВСЕ, см. ниже) и шифратора.

6.3. Схема сравнения битовых векторов (ССБВ)

Исходными данными для схемы является пара битовых векторов X и Y, выступающих в роли битовых представлений двух множеств. Результатом работы схемы должны быть двоичные значения признаков полного δ^+ и частичного δ^- совпадения векторов. Очевидно, что признак δ^+ должен получить единичное значение только в случае побитового совпадения векторов:

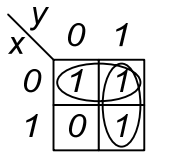
$$\delta^+ = \mathop{\&}_{i=\overline{1,n}} (x_i \sim y_i) = \mathop{\&}_{i=\overline{1,n}} (x_i y_i \vee \overline{x}_i \overline{y}_i).$$

Согласно предназначению схемы признак δ^- должен быть установлен в случае, если $X \subset Y$. Другими словами, значение признака δ^- должно быть нулевым только если в составе векторов на некоторой позиции i $x_i = 1$, а $y_i = 0$ (i-й элемент присутствует в X и не присутствует в Y). Следовательно

$$\delta^- = \mathop{\&}_{i=\overline{1,n}} f(x_i, y_i) = \mathop{\&}_{i=\overline{1,n}} \overline{x_i \overline{y}_i}.$$

Использованная в приведенной формуле функция δ^- синтезирована на основе приведенных выше рассуждений как показано на рис. 20.

x	y	f
0	0	1
0	1	1
1	0	0
1	1	1



$$f = \overline{x} \vee y = \overline{x\overline{y}}$$

Рис. 20. Таблица истинности и карта Карно для функции δ^-

Формулы, подобные приведенным выше, могут быть использованы и при других операциях над множествами, представленными в виде битовых векторов [22].

На основании полученных формул можно синтезировать искомую схему (рис. 21).

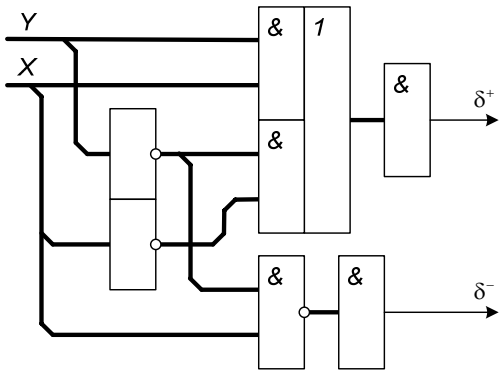


Рис. 21. Схема сравнения битовых векторов

Ее аппаратная сложность составляет $8n-2$ эквивалентных вентилей (n в данном случае – разрядность входных векторов), а время задержки появления результата составляет $\left(3+\lceil \log_2 n \rceil\right)t_0$, где $\lceil x \rceil$ – операция округления вверх до ближайшего целого.

6.4. Схемы выделения заданных граничных значений

При выполнении некоторых действий возникает необходимости в нахождении в заданном двоичном векторе самого младшего или самого старшего бита, значение которого равно заданному. Примеры подобных действий приведены в таблице 1.

Таблица 1. Примеры выделения граничных значений

Действие	**Исходный вектор *X***	**Результат *Y***
Выделение младшего нуля $g_0^{\downarrow}(X)$	11**0**0101011101010	0010 (2)
Выделение старшего нуля $g_0^{\uparrow}(X)$	00110111110**1**1111	1010 (10)
Выделение младшей единицы $g_1^{\downarrow}(X)$	000**1**001101111010	0011 (3)
Выделение старшей единицы $g_1^{\uparrow}(X)$	10111011110**1**0000	1011 (11)

Комбинационные схемы, выполняющие подобные действия, далее именуются как схема выделения младшего нуля (СВМН), схема выделения старшего нуля (СВСН), схема выделения младшей единицы (СВМЕ) и схема выделения старшей единицы (СВСЕ).

Любая из перечисленных схем может быть синтезирована в МДНФ форме, однако с учетом большого числа входов минимизация получаемых в ходе анализа таблиц истинности СДНФ форм с использованием карт Карно напрямую затруднительна, а получаемые при этом формулы и схемы достаточно громоздки. Исходя из этого (не исключая возможности синтеза МДФН форм выходов схемы) рассмотрим обобщенный способ синтеза схем на примере схемы СВМН (рис. 22).

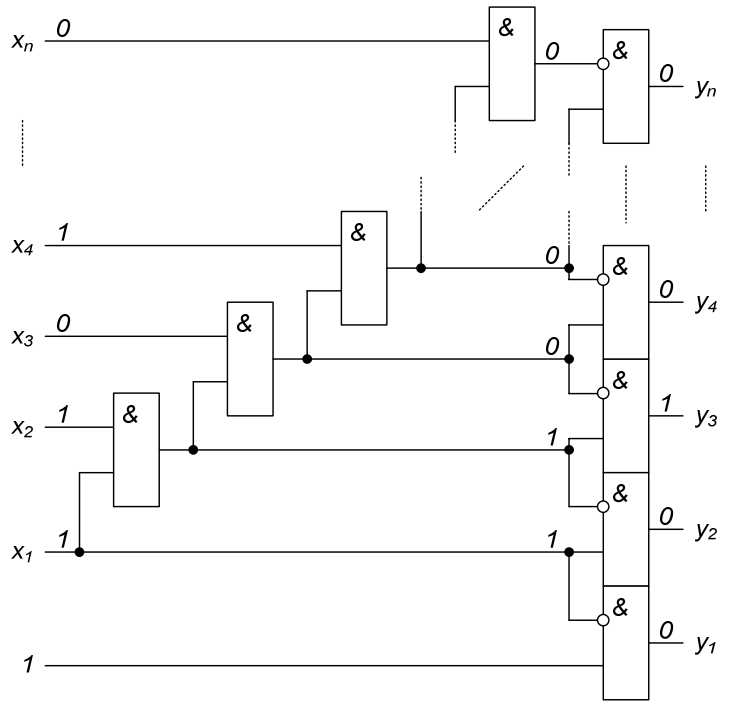


Рис. 22. Обобщенная схема выделения младшего нуля

Элементы И, объединенные в цепочку, преобразуют исходный двоичный вектор в вектор вида «11…100…0», а стоящие следом на ними элементы запрета обеспечивают единицу на выходе того элемента, на входе которого присутствует комбинация значений «10». Т.к. в получаемом таким способом выходном векторе присутствует на один бит меньше, чем в исходном, в качестве младшего разряда (x_0) к исходному вектору добавляется единичный бит. При этом следует отметить, что элемент запрета, формирующий значение младшего разряда выхода, изображен на рис. 22 исключительно из соображений наглядности и общности: при практической реализации схемы он может быть заменен на инвертор младшего разряда, т.к. $\overline{x}_1 \wedge 1 = \overline{x}_1$.

Схема, представленная на рис. 22, характеризуется аппаратной сложностью $\underbrace{n-1}_{И} + \underbrace{n-1}_{запрет} + \underbrace{1}_{НЕ} = 2n-1$ вентилей, а время задержки получения результата, составляющее nt_0, линейно зависит от разрядности входного вектора.

Схема СВСН легко получается из рассмотренной выше схемы СВМН путем перенумерации входов: первый выход становится последним, второй – предпоследним и т.д.

Схема СВМЕ может быть синтезирована двумя способами. Первый из них заключается в инвертировании при помощи блока инверторов исходного вектора (включая добавленный младший разряд x_0 в случае использования схемы, представленной на рис. 22) с последующим использованием схемы СВМН. Основным недостатком этого способа является увеличение времени формирования результирующего вектора на величину t_0. Второй способ сводится к синтезу самостоятельной схемы (рис. 23), принцип работы которой в целом напоминает принцип работы схемы СВМН (рис. 22).

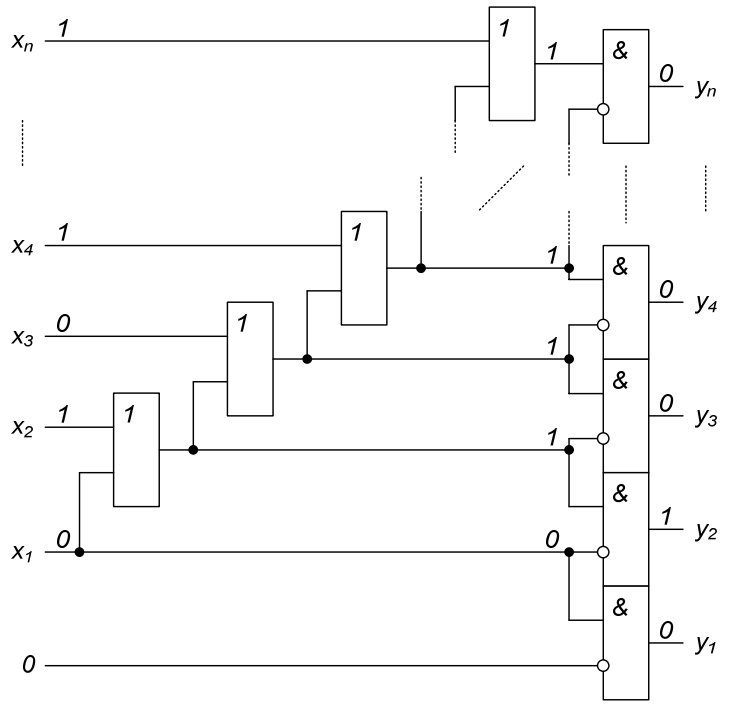


Рис. 23. Схема выделения младшей единицы

На выходах элементов ИЛИ, объединенных в цепочку, исходный вектор преобразуется в вектор вида «00…011…0». Далее на выходе элемента запрета, на вход которого поданы значения «01», формируется единичное значение выходного вектора. На выходах всех остальных элементов запрета формируются нулевые значения. Как и в случае схемы на рис. 22, элемент запрета, на выходе которого формируется значение младшего бита результата, приведен на рис. 23 из соображений наглядности и общности изображения схемы, т.к. он эквивалентен значению младшего бита входного вектора $y_1 = x_1 \wedge \overline{0} = x_1$.

Подобная схема, например, используется при реализации приоритетных шифраторов [23].

Схема СВСЕ, аналогично схеме СВСН, может быть получена из приведенной на рис. 23 схемы путем перенумерации входов. Аппаратная сложность схемы на рис. 23 составляет $\underbrace{n-1}_{ИЛИ} + \underbrace{n-1}_{запрет} = 2n-2$ вентилей, а ее время задержки формирования выходного вектора – nt_0.

6.5. Схема определения типа соответствия предка (СОТСП)

Схема СОТСП используется в составе схемы определения отношения нестрогого включения в процессе отыскания подстановки *r*-изоморфизма при корректировке значения поля ТС узла-предка при рассмотрении пары узлов предок-потомок [9, 10]. С учетом того, что рассмотрение поля ТС возможно как вектора из признаков δ^+ и δ^-, таблица истинности, карты Карно и результирующие функции выходов схемы СОТСП приведены на рис. 24.

СП↓		СП↑		СП'	
x_1	x_2	x_3	x_4	δ^-	δ^+
0	*	0	*	0	*
0	*	1	0	0	*
0	*	1	1	0	*
1	0	0	*	0	*
1	0	1	0	0	*
1	0	1	1	0	*
1	1	0	*	0	*
1	1	1	0	1	0
1	1	1	1	1	1

$\delta^{-\prime} = x_1 x_2 x_3 = \delta^{-\downarrow}\delta^{+\downarrow}\delta^{-\uparrow}$ $\quad$ $\delta^{+\prime} = x_4 = \delta^{+\uparrow}$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	0	0	0	0

x_1x_2 \ x_3x_4	00	01	11	10
00	*	*	*	*
01	*	*	*	*
11	*	*	1	0
10	*	*	*	*

Рис. 24. Синтез функций схемы определения типа соответствия предка

Выражения обновленных значений признаков $\delta^{+\prime}$ и $\delta^{-\prime}$ синтезированы для случая, когда у рассматриваемого узла в дереве A^R и его *r*-изоморфного эквивалента в дереве B^R имеются узлы-предки. Случай, когда в дереве B^R у рассматриваемого узла предок отсутствует, идентифицируется единичным значением признака $\tilde{\gamma}$. При этом значение поля ТС должно быть установлено в «соответствие отсутствует» («0*»), т.е. признак $\delta^{-\prime}$ должен иметь нулевое значение. С учетом этой особенности схему СОТСП можно представить в следующем виде (рис. 25).

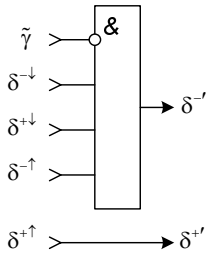


Рис. 25. Схема определения типа соответствия предка

7. Схемотехническая реализация операции определения принадлежности вершины дереву

Потребность в выяснении принадлежности вершины дереву возникает при проведении *u*- и *d*-подстановок, при этом проверяется принадлежность начальной или конечной вершины граф-схемы алгоритма одному из деревьев выражения системы Ξ и производится выдача признака μ, разрешающего выполнение последующих действий.

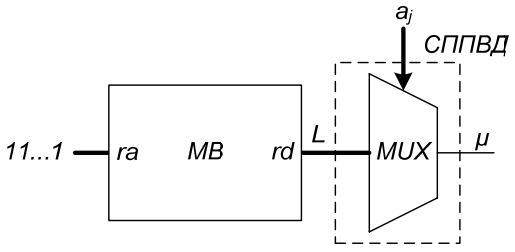


Рис. 26. Схема проверки принадлежности вершины дереву (СППВД)

На вход *ra* элемента МВ обрабатываемого *R*-выражения подается значение «11…1», которое инициирует чтение и поэлементную дизъюнкцию всех наборов листьев дерева, значение которой (вектор *L*) формируется на выходе *rd* (для корректной работы схемы поля

МВ неиспользованных наборов листьев должны содержать нулевые значения «00…0»). Сформированный вектор L подается на информационный вход мультиплексора, а на его адресный вход – номер интересующей вершины, в результате чего на выходе мультиплексора появляется значение признака μ присутствия искомой j-ой вершины в дереве.

По сравнению с программной реализацией выигрыш во времени достигается, во-первых, за счет параллельной векторной обработки всех бит векторов МВ в составе ОСЭМД, а во-вторых за счет параллельной обработки самих векторов (при программной реализации обе операции выполняются последовательно).

8. Схемотехническая реализация операции определения ω-мощности дерева

Алгоритм определения ω-мощности дерева [10] состоит из трех этапов. На первом этапе производится инициализация начальных значений полей МУ. На втором этапе выполняется присваивание начальных значений полей МУ узлам-предкам наборов листьев (у каждого узла дерева может быть не более одного набора листьев, поэтому коллизий с повторным присваиванием значения от другого набора листьев быть не может). На третьем этапе производится просмотр узлов дерева снизу вверх и итеративная корректировка значений полей МУ, в результате которой поле МУ корня дерева будет содержать искомую ω-мощность дерева в целом. Выполнения указанных действий реализуется схемой, изображенной на рис. 27.

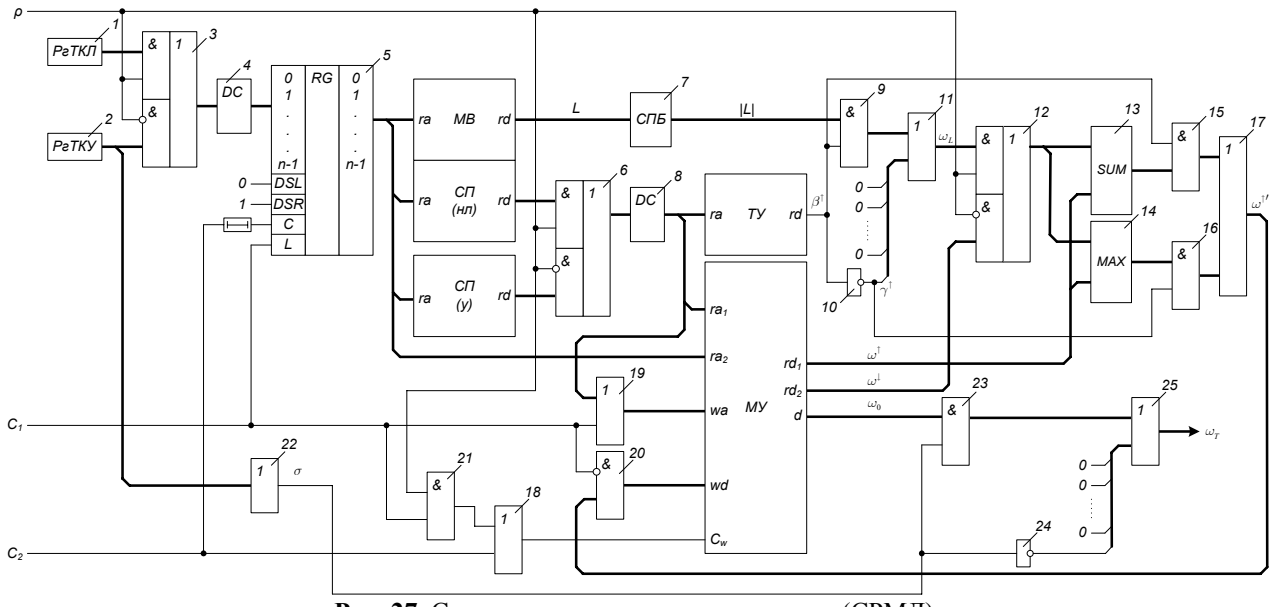


Рис. 27. Схема расчета ω-мощности дерева (СРМД)

Регистры 1 и 2 используются для хранения текущего количества листьев и текущего количества узлов соответственно. Через коммутатор 3 и дешифратор 4 значение одного из регистров в зависимости от этапа алгоритма вычисления ω-мощности сохраняется в сдвиговом регистре 5. Значение, хранящееся в регистре 5, является адресом для чтения значений полей МВ, СП набора листьев, СП узла и МУ элементов дерева. Коммутатор 6 и дешифратор 8 формируют адрес для чтения значений полей ТУ и МУ элементов дерева в зависимости от этапа алгоритма вычисления ω-мощности. Схема подсчета бит 7 формирует значение количества единичных бит в битовом векторе на ее входе. Элементы 9–11 используются для формирования значения ω-мощности набора листьев в зависимости от

значения поля ТУ предка. Коммутатор 12 используется для выбора текущего значения ω-мощности в зависимости от этапа алгоритма вычисления ω-мощности. Элементы 13–17 используются для формирования обновленного значения ω-мощности предка текущего узла $\omega^{\uparrow\prime}$. Элементы 18 и 21 предназначены для управления сигналом записи обновленных значений ω-мощности узлов в зависимости от синхросигналов C_1 и C_2. В совокупности с ними, элементы 19 и 20 формируют на входах адреса и данных записи элемента МУ необходимые значения в зависимости от этапа алгоритма вычисления ω-мощности. Элемент 22 служит для выработки признака σ, сигнализирующего о том, что дерево представлено единственным набором листьев. В случае, если $\sigma = 0$, элементы 23–25 используются для формирования единичного значения на выходе схемы. В противном случае на выходе схемы присутствует искомое значение ω-мощности дерева, синтезированное в ходе предыдущих итераций работы устройства.

На этапе инициализации на управляющий вход коммутатора 3 подается сигнал $\rho = 1$, который обеспечивает подачу значения текущего количества наборов листьев из регистра РгТКЛ на вход дешифратора 4. С выхода дешифратора 4 на вход регистра 5 поступает двоичный код из всех нулей и единицы в N_L-м разряде, который записывается в регистр при подаче синхросигнала C_1. Также при появлении синхросигнала C_1 на входе *wa* элемента ОСЭМД, хранящего поле МУ, благодаря элементу ИЛИ 19 генерируется вектор «11…1»; на входе *wd* – вектор благодаря элементу И 20 «00…0»; на входе C_w – сигнал логической единицы с выхода элемента ИЛИ 18, прошедший через элемент И 21 благодаря единичному значению сигнала ρ, что в совокупности обеспечивает инициализацию нулями полей МУ всех узлов дерева.

Далее осуществляется *цикл сканирования наборов листьев*. Для этого устройство управления (на рис. 27 не показано) выдает серию из N_L синхроимпульсов C_2. К моменту появления очередного синхроимпульса состояние комбинационных цепей схемы должно быть установлено следующим образом. Шина данных с выхода регистра 5, подключенная ко входу *ra* элемента МВ, обеспечивает появление на выходе *rd* вектора множества вершин *i*-го набора листьев, который поступает на вход схемы подсчета бит (СПБ) 7 и формирует на выходе схемы значение количества единичных бит в векторе. Также сигналы с выхода регистра 5 подаются на входы *ra* элементов СП наборов узлов (сокращенно «СП (у)») и СП наборов листьев (сокращенно «СП (нл)»), на выходах *rd* которых формируются значения полей СП выбранного *i*-го набора. Сигнал ρ по прежнему имеет единичное значение, поэтому на выходе коммутатора 6 присутствует значение поля СП набора листьев, которое поступает на вход дешифратора 8. С выхода дешифратора 8 битовый вектор из нулей и единицы в разряде, соответствующем значению поля СП выбранного набора листьев, поступает на вход *ra* элемента ТУ, на выходе *rd* которого формируется значение типа узла – предка рассматриваемого набора листьев $\beta^{\uparrow}$. Также выход дешифратора 8 подключен ко входу *ra* первого порта элемента МУ, на выходе *rd* первого порта которого формируется значение мощности узла предка рассматриваемого набора листьев $\omega^{\uparrow}$; и ко входу элемента ИЛИ 19, через который значение с выхода дешифратора 8 проходит на вход *wa* элемента МУ без изменения ввиду нулевого значения сигнала C_1 в данный момент, подготавливая тем самым элемент МУ к записи. Элементы 9, 10 и 11 формируют на выходе элемента 11 значение ω-мощности набора листьев в соответствии с алгоритмом [10] (значение равно 1 в случае альтернативного типа узла предка и количеству единичных элементов $|L|$ в битовом векторе *L* в случае параллельного типа узла предка), которое поступает на вход коммутатора 12. Второй вход коммутатора 12 в данном режиме работы схемы не используется, т.к. $\rho = 1$. На выходе коммутатора 12 формируется значение ω-мощности набора листьев ω_L в

соответствии с типом узла предка. Элементы 13–17 обеспечивают формирование обновленного значения ω -мощности $\omega^{\uparrow\prime}$ узла-предка в зависимости от его типа ($\omega^{\uparrow\prime}=\omega^{\uparrow}+\omega^{\downarrow}$ в случае параллельного типа узла-предка и $\omega^{\uparrow\prime}=\max\left(\omega^{\uparrow},\omega^{\downarrow}\right)$ в случае альтернативного). Несложно показать, что во время этапа обработки листьев всегда выполняется соотношение $\omega^{\uparrow\prime}=\omega_L$ (в случае параллельного типа узла-предка $\omega^{\uparrow\prime}=\omega_L+0=\omega_L$, а в случае альтернативного – $\omega^{\uparrow\prime}=\max\left(\omega_L,0\right)=\omega_L$, нулевое значение $\omega^{\uparrow}$ записано в элемент МУ на предыдущем этапе – этапе инициализации, при этом ненулевого значения $\omega^{\uparrow}$ быть не может, т.к. все узлы имеют не более одного набора листьев согласно требованиям корректности. Полученное обновленное значение $\omega^{\uparrow\prime}$ проходит через элемент И 20 без изменения, т.к. $C_1=0$, и поступает на вход *wd* элемента МУ. Таким образом, на входах *wa* и *wd* сформированы адрес рассматриваемого элемента дерева и обновленное значение его ω -мощности соответственно. При поступлении на вход C_w элемента МУ синхросигнала C_2, проходящего через элемент ИЛИ 18, происходит запись обновленного значения ω-мощности. Спустя некоторый промежуток времени, формируемый элементом задержки, синхросигнал C_2 поступает на синхровход регистра 5, сдвигая единицу в битовом векторе в сторону младших разрядов, т.е. фактически производится переход к новой итерации этапа алгоритма. В результате, по завершении N_L итераций будут сформированы начальные значения ω -мощностей узлов дерева, имеющих наборы листьев в качестве потомков.

Далее осуществляется *цикл сканирования узлов дерева*. Для этого сигнал $\rho=0$ формирует на выходе коммутатора 3 значение текущего количества узлов дерева, которое подается на вход дешифратора 4. На выходе дешифратора 4 формируется битовый вектор из нулей и единицы в N_T-ом разряде. Битовый вектор записывается в регистр 5 при появлении синхросигнала C_1, при этом модификация значений МУ, полученных на предыдущем этапе, не происходит, т.к. синхроимпульс C_1 не проходит через элемент И 21 из-за нулевого значения сигнала ρ . После инициализации значения регистра 5 синхросигналом C_1 устройство управления выдает серию из N_T-1 синхросигналов C_2 (количество синхросигналов на единицу меньше количества узлов дерева, т.к. значение ω-мощности корня дерева формируется в процессе рассмотрения его потомков и не требует отдельной итерации работы схемы). К моменту появления очередного синхросигнала C_2 состояние комбинационной части схемы устанавливается следующим образом. Битовый вектор с выхода регистра 5 подается на вход *ra* второго порта элемента МУ, на выходе *rd* второго порта которого формируется значение ω -мощности текущего *i*-го узла дерева $\omega^{\downarrow}$. Также битовый вектор с выхода регистра 5 подается на входы *ra* элементов СП узлов и наборов листьев. Т.к. сигнал $\rho=0$, то на выходе коммутатора 6 формируется значение поля СП рассматриваемого *i*-го узла дерева, которое подается на вход дешифратора 8. Значение с выхода дешифратора 8 подается на вход *ra* элемента ТУ, на выходе *rd* которого формируется значение типа узла предка $\beta^{\uparrow}$; на вход *ra* первого порта элемента МУ, на выходе *rd* первого порта которой формируется значение ω-мощности предка текущего узла $\omega^{\uparrow}$; и через элемент ИЛИ 19 на вход *wa* элемента МУ. Благодаря нулевому значению сигнала ρ на выходе коммутатора 12 формируется значение ω -мощности текущего *i*-го узла $\omega^{\downarrow}$ (значение, сформированное элементами 7, 9–11, на данном этапе работы схемы не важно), которое подается на входы элементов 13 и 14. Также на входы элементов 13 и 14 подается значение

ω-мощности предка *i*-го узла $\omega^{\uparrow}$, что в совокупности с сигналами $\beta^{\uparrow}$ и $\gamma^{\uparrow}$ обеспечивает формирование на выходе блока элементов ИЛИ 17 обновленного значения ω-мощности предка *i*-го узла $\omega^{\uparrow\prime}$, которое проходит через элемент И 20 и подается на вход *wd* элемента МУ. Таким образом, к моменту появления синхросигнала C_2 на входах элемента МУ сформирован адрес предка текущего *i*-го узла и обновленное значение его ω-мощности, записываемое в элемент МУ по синхросигналу C_2, проходящему через элемент ИЛИ 18 на его вход C_w. Спустя промежуток времени, формируемый элементом задержки, синхросигнал поступает на синхровход регистра 5, сдвигая единицу на одну позицию вправо (в сторону младших разрядов), тем самым подготавливая схему к новой итерации. В результате появления серии из $N_T - 1$ синхроимпульсов C_2 в нулевом элементе МУ будет сформировано искомое значение ω-мощности дерева ω_0, используемое в дальнейших преобразованиях.

Возможна ситуация, когда дерево состоит только из одного набора листьев (примером может служить левая часть *R*-выражения $a_0 \rightarrow a_1 \,|\, a_2$), ω-мощность такого дерева равна единице. Для корректной обработки такой ситуации в схему введены элементы 22–25. На выходе элемента ИЛИ 22 формируется значение признака σ, единичное значение которого сигнализирует о присутствии в дереве узлов. Элементы 23–25 формируют на выходе схемы значение $\omega_T = \omega_0$ в случае присутствия в дереве хотя бы одного узла $(\sigma = 1)$ и значение «00…01» $(\omega_T = 1)$ в противном случае.

Временные диаграммы работы схемы, поясняющие принцип ее работы, приведены на рис. 28.

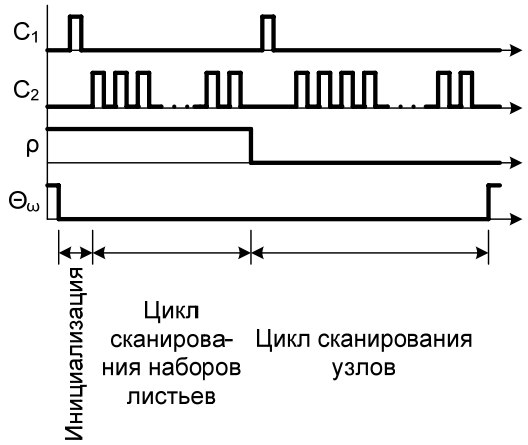


Рис. 28. Временные диаграммы работы схемы определения ω-мощности дерева

В приведенной на рис. 27 схеме, как и во всех последующих, использование конкретных номеров портов не является жестко заданным: в процессе получения электрической принципиальной схемы (или топологии межсоединений элементов на плате или кристалле) возможно подключение шин адресов и данных к другим портам, если это продиктовано какими-либо не рассматриваемыми в данной работе соображениями (нагрузочная способность выводов, паразитные емкости межсоединений, количество слоев межсоединений и т.д.).

Время расчета значения ω-мощности дерева зависит от количества элементов дерева и может сильно отличаться для деревьев различной структуры, поэтому по завершении расчета устройством управления выдается единичное значение признака Θ_ω, являющееся сигналом к дальнейшим действиям.

По сравнению с программной реализацией выигрыш во времени достигается за счет:

- параллельной инициализации значений полей МУ;
- параллельного подсчета количества элементов в векторах листьев схемой подсчета бит;
- параллельного чтения значений МУ $\omega^{\downarrow}$ и $\omega^{\uparrow}$ из разных портов ОСЭМД;
- параллельного чтения значений различных полей (например, МВ и ТУ).

Остальные операции (сканирование наборов листьев и узлов дерева) осуществляются последовательно.

При программной реализации алгоритма производится обход всех элементов дерева в направлении сверху-вниз (от корня к листьям), при этом требуется просмотреть N_T узлов дерева и N_L наборов листьев, причем при рассмотрении каждого из наборов листьев необходимо последовательное сканирование всех $L_{\max}$ компонентов вектора МВ. Таким образом, нужно выполнение порядка $O\left(L_{\max} \cdot N_L + N_T\right) \simeq O\left(L_{\max} \cdot N_R\right)$ операций.

По сравнению с программной реализацией предлагаемое аппаратное решение дает выигрыш во времени за счет параллельного подсчета количества элементов в векторах листьев схемой подсчета бит. При этом остальные операции (сканирование наборов листьев и узлов дерева) выполняются последовательно. Аппаратная реализация имеет асимптотическую временную сложность порядка $O\left(N_L + N_T\right) = O\left(N_R\right)$ (при условии, что подсчет числа единичных бит в битовом векторе осуществляется схемой подсчета бит за время, не зависящее от $L_{\max}$), т.е. имеет место выигрыш в скорости обработки в $\frac{L_{\max} N_L + N_T}{N_L + N_T} \approx \frac{N_R\left(L_{\max} + 1\right)}{N_R} = L_{\max} + 1 \simeq O\left(L_{\max}\right)$ раз.

Возможен также вариант схемы, в котором обработка всех наборов листьев будет производится параллельно во времени (ввиду отсутствия зависимостей по данным между ними) и временная сложность составит величину порядка $O\left(N_T\right) \simeq O\left(N_R\right)$. Такая схема будет работать несколько быстрее предложенной на рис. 27 (несмотря на схожие асимптотики выигрыш во времени будет, т.к. $k_1 N_T + k_2 \le k_1 N_T + k_2 N_L$, где k_1 и k_2 – коэффициенты, опускаемые при рассмотрении асимптотических сложностей), однако аппаратная сложность подобной схемы будет как минимум на 1–2 порядка выше из-за дублирования схем подсчета бит (СПБ) и необходимости параллельного N_L -портового чтения поля ТУ и N_L-портовой записи значений поля МУ. С учетом немалой аппаратной сложности предложенной схемы определения ω -мощности дерева (обработка наборов листьев осуществляется последовательно) такой шаг на данном этапе развития микроэлектроники является неоправданным.

Как было отмечено в [24], ввиду отсутствия зависимостей возможна также параллельная обработка непересекающихся поддеревьев рассматриваемого дерева, что теоретически должно позволить сокращение времени обработки в среднем в K раз, где K – среднее количество потомков у узлов дерева, однако схемотехническая реализация данной идеи при сохранении предложенной выше табличной формы представления дерева затруднительна.

9. Схемотехническая реализация операции проверки отношения нестрогого включения

Как уже было отмечено выше, в процессе применения правил *u*- и *d*-подстановок одним из первых выполняется операция проверки отношения нестрогого включения, за которой (в случае положительного исхода данной и ряда дополнительных проверок) следует операция подстановки. В результате выполнения проверки устанавливается как сам факт

эквивалентности (*R*-выражения A^R и B^R находятся $A^R[\subseteq]B^R$ или не находятся $A^R[\not\subseteq]B^R$ в отношении нестрогого включения), так и тип соответствия (полное или частичное). В процессе выяснения отношения также производится выяснение подстановки изоморфизма [9, 25].

На рис. 29 приведена схема, синтезированная для установления *r*-изоморфизма деревьев в соответствии с предложенным в [25] алгоритмом.

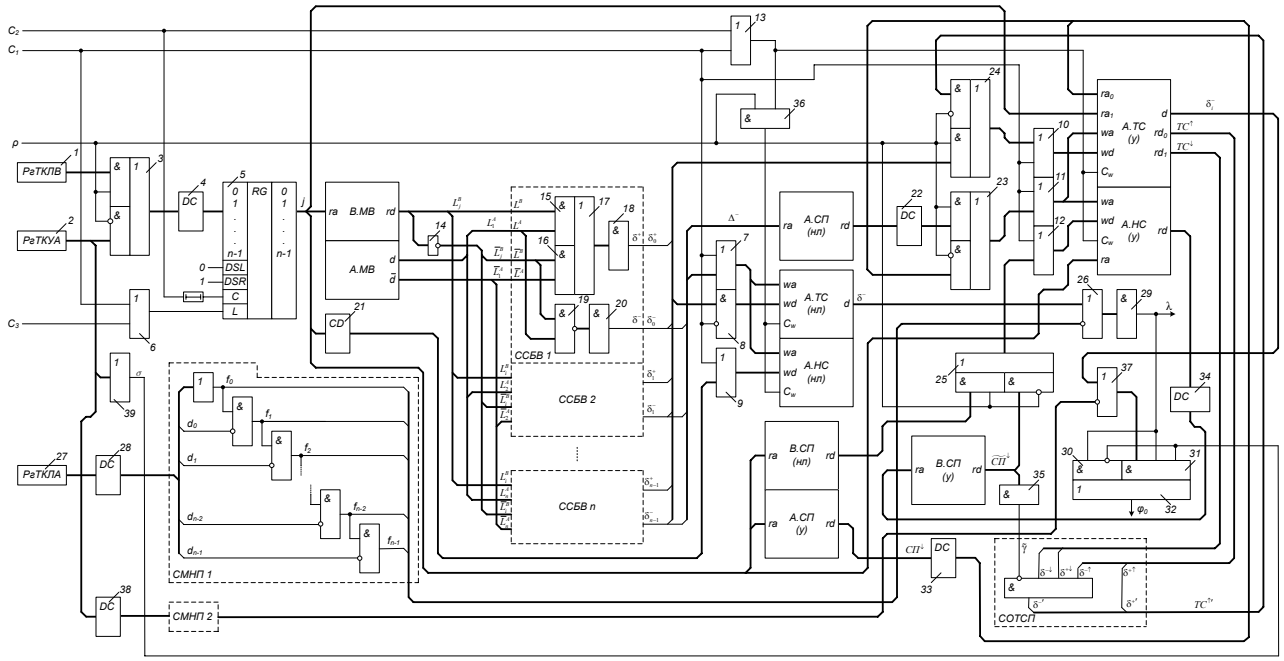


Рис. 29. Схема проверки отношения нестрогого включения (СПОНВ)

Регистры 1, 2 и 27 предназначены для хранения значений $N_L(B)$, $N_T(A)$ и $N_L(A)$ соответственно, при этом значения полей обрабатываемых деревьев хранятся в элементах ОСЭМД B.MB, A.MB, A.СП(нл), A.ТС(нл), A.НС(нл), B.СП(нл), A.СП(у), B.СП(у), A.ТС(у) и A.НС(у). Коммутаторы 3, 23–25 управляются сигналом ρ и предназначены для коммутации значений на разных этапах работы алгоритма. Дешифраторы 4, 22, 28, 33, 34, 38 используются для преобразования номеров позиций элементов дерева в табличном представлении из двоичного кода в унитарный, используемый на входах *ra* элементов ОСЭМД. Шифратор 21 используется для обратного преобразования. Сдвиговый регистр 5 используется для поочередного выбора элементов обрабатываемых деревьев. Блоки элементов ИЛИ 7, запрета 8, ИЛИ 9–12 используются для формирования начальных значений полей на этапе инициализации. Элементы ИЛИ 6, ИЛИ 13 и И 36 используются для коммутации синхросигналов. Блок элементов запрета 26 в совокупности с элементом И 29 и схемой СМНП 1 используются для формирования значения признака λ отсутствия *r*-изоморфизма. Схема СМНП 2 в совокупности с блоком элементов запрета 37, элементом ИЛИ 39, запрета 30, блоком элементов И 31 и элементом ИЛИ 32 используются для формирования результирующего признака *r*-изоморфизма φ_0, при этом на выходе элемента ИЛИ 39 формируется значение признака σ. Элемент И 35, на выходе которого формируется значение признака $\tilde{\gamma}$ в совокупности со схемой СОТСП отвечают за формирование обновленного значения поля типа соответствия узла-предка. Блок элементов НЕ 14 в совокупности со схемами ССБВ 1–*n*, образованными блоками элементов И 15 и 16, ИЛИ 17, И-НЕ 19 и элементами И 18 и 20 используются для формирования признаков δ_i^+ и δ_i^-. Элемент задержки предназначен для задержки синхросигнала сдвига содержимого регистра

5 на время, достаточное для формирования и записи обновленных значений полей на предыдущей итерации алгоритма.

Предлагаемая схема работает следующим образом. Перед началом работы в регистр РгТКЛВ загружается значение $N_L(B)$, в регистр РгТКУА – $N_T(A)$, а в регистр РгТКЛА – $N_L(A)$. В элементы В.МВ, А.МВ, А.СП(нл), А.СП(у), В.СП(нл), В.СП(у), хранящие одноименные значения полей деревьев A^R и B^R, загружается необходимая информация о конфигурации деревьев.

На *этапе инициализации* сигнал $\rho = 1$ обеспечивает передачу значения $N_L(B)$ из регистра РгТКЛ1 через коммутатор 3 на вход дешифратора 4 в двоичном коде. С выхода дешифратора 4 значение $N_L(B)$ в унитарном коде поступает на вход сдвигового регистра 5. По синхросигналу C_1, проходящему через элемент ИЛИ 6, происходит запись значения $N_L(B)$ в сдвиговый регистр 5. Появление единичного значения синхросигнала C_1 обеспечивает появление на выходе блока элементов ИЛИ 7 значения «11…1», которое поступает на входы *wa* элементов А.ТС(нл) и А.НС(нл), выбирая тем самым все строки указанных элементов для записи. На выходе блока элементов запрета 8 единичное значение синхросигнала C_1 обеспечивает появление значения «00…0», которое подается на вход *wd* элемента А.ТС(нл). На выходе блока элементов ИЛИ 9 единичное значение синхросигнала C_1 обеспечивает формирование значения «11…1», которое поступает на вход *wd* элемента А.НС(нл). Также синхросигнал C_1 обеспечивает формирование значений «11…1» на выходах блоков элементов ИЛИ 10–12, откуда они поступают на входы *wa* и *wd* элементов А.ТС(у) и А.НС(у). Также синхросигнал C_1 проходит через элементы ИЛИ 13 и И 36 на входы C_w элементов А.ТС(нл), А.НС(нл), А.ТС(у) и А.НС(у) и инициирует процесс записи сформированных начальных значений в указанные элементы.

Далее следует *этап параллельного сравнения каждого из наборов листьев дерева B^R со всеми наборами листьев дерева A^R*. Значение *j* с выхода сдвигового регистра 5 поступает на вход *ra* элемента В.МВ, что обеспечивает появление на его выходе *rd* битового вектора *j*-го набора листьев L_j^B, поступающего на первые входы схем ССБВ 1–*n* ($n = L_{\max}$ в данном случае). Также битовый вектор L_j^B подается на вход блока инверторов 14, на выходе которого формируется инвертированное значение $\overline{L}_j^B$, поступающее на третьи входы схем ССБВ 1–*n*. С выходов *d* и $\overline{d}$ элемента А.МВ множество битовых векторов наборов листьев подается на вторые и четвертые входы схем ССБВ 1–*n* соответственно (значения L_0^A и $\overline{L}_0^A$ подаются на входы ССБВ 1, L_1^A и $\overline{L}_1^A$ – на входы ССБВ 2 и т.д.). На выходах схем ССБВ, образованных элементами $15_i - 20_i$, $i = \overline{1,n}$ формируются значения признаков полного δ_i^+ и частичного δ_i^- соответствия битовых векторов L_j^B и L_i^A. Исходя из особых свойств *R*-выражений [9, 10] возможно не более одного совпадения битовых векторов, т.е. вектор Δ^-, состоящий из значений δ_i^- и проходящий через блок элементов ИЛИ 7 на входы *wa* элементов А.ТС(нл) и А.НС(нл), содержит не более одной единицы: для записи выбирается либо *i*-я строка элементов А.ТС(нл) и А.НС(нл), где *i* – номер набора листьев L_i^A, находящегося в полном или частичном соответствии с набором листьев L_j^B, либо $\Delta^- = 00...0$, т.е. последующая запись обновленных значений в элементы А.ТС(нл) и А.НС(нл) не происходит. Значения признаков δ_i^- и δ_i^+ с выходов схем ССБВ 1–*n* поступают через блок элементов запрета 8 на вход *wd* элемента А.ТС(нл), образуя обновленные значения полей ТС

наборов листьев дерева A^R. Значение номера *j* рассматриваемого набора листьев дерева B^R в унитарном коде поступает на вход шифратора 21, с выхода которого значение *j* в двоичном коде проходит через блок элементов ИЛИ 9 на вход *wd* элемента А.НС(нл). Таким образом, элементы А.ТС(нл) и А.НС(нл) готовы к записи обновленных значений полей ТС и НС *i*-го набора листьев дерева A^R, для которого наблюдается полное или частичное совпадение с *j*-м набором листьев дерева B^R. Вектор признаков Δ^- с выходов схем ССБВ 1–*n* поступает на вход *ra* элемента А.СП(нл), на выходе *rd* которого формируется значение номера предка *i*-го набора листьев в двоичном коде. Значение с выхода *rd* элемента А.СП(нл) поступает на вход дешифратора 22, на выходе которого формируется значение предка *i*-го набора листьев в унитарном коде. Сформированное значение проходит через коммутатор 23 и блок элементов ИЛИ 11 на входы *wa* элементов А.ТС(у) и А.НС(у). Значения признаков δ_i^- и δ_i^+ с выходов схем ССБВ 1–*n* поступают через коммутатор 24 и блок элементов ИЛИ 10 на вход *wd* элемента А.ТС(у). Значение *j* с выхода сдвигового регистра 5 поступает на вход *ra* элемента В.СП(нл), на выходе *rd* которого формируется значение номера узла-предка рассматриваемого *j*-го набора листьев в составе дерева B^R. Сформированное значение проходит через коммутатор 25 и блок элементов ИЛИ 12 на вход *wd* элемента А.НС(у). Таким образом, на входах *wa* и *wd* элементов А.ТС(у) и А.НС(у) сформированы обновленные значения полей ТС и НС узлов дерева, являющихся предками наборов листьев. Появление синхросигнала C_2, проходящего через элемент ИЛИ 13 на входы C_w элементов А.ТС(нл), А.НС(нл), А.ТС(у) и А.НС(у), обеспечивает запись обновленных значений полей в перечисленные элементы. Также синхросигнал C_2 проходит через элемент задержки на синхровход сдвигового регистра 5, сдвигая единицу в хранящемся в нем битовом векторе в сторону младших разрядов: происходит переход к следующей итерации алгоритма. После получения $N_L(B)$ синхросигналов C_2 в элементах А.ТС(нл), А.НС(нл), А.ТС(у) и А.НС(у) сформированы значения полей ТС и НС наборов листьев и узлов, являющихся их предками, в составе дерева A^R.

Сформированные значения признаков δ_i^- с выхода *d* элемента А.ТС(нл) поступают на первый вход блока элементов запрета 26. Значение $N_L(A)$ количества наборов листьев в составе дерева A^R в двоичном коде с выхода регистра РгТКЛА 27 поступает на вход дешифратора 28. С выхода дешифратора 28 значение $N_L(A)$ в унитарном коде поступает на вход схемы СМНП 1, на выходе которой формируется вектор вида $00...0\underbrace{11...1}_{N_L(A)}$. В младших разрядах вектора присутствует $N_L(A)$ единиц, обозначающих используемые для хранения наборов листьев дерева позиции в его табличном представлении, старшие разряды, отвечающие неиспользуемым позициям, содержат нули. Сформированный на выходе схемы СМНП 1 вектор поступает на второй вход блока элементов запрета 26. На выходе блока элементов запрета 26 формируется вектор, единичное значение *i*-го бита в котором говорит либо об установленном соответствии для *i*-го набора листьев дерева A^R, либо о неиспользованной *i*-ой позиции в табличном представлении дерева A^R. Значение с выхода блока элементов запрета 26 поступает на вход элемента И 29, на выходе которого формируется значение признака λ. В случае, если $\lambda = 0$, имеет место отсутствие соответствия для одного или нескольких наборов листьев в составе дерева A^R. При этом сформированное нулевое значение на выходе элемента И 29 поступает на входы элементов запрета 30 и И 31 и формирует на выходе элемента ИЛИ 32 нулевое значение признака φ_0: дерево A^R не является *r*-изоморфным дереву B^R. Работа устройства на этом окончена. В случае, если $\lambda = 1$ работа устройства продолжается.

Перед началом *этапа определения значений полей ТС и НС узлов дерева* A^R производится сброс в ноль сигнала ρ. В результате этого значение $N_T(A)$ текущего количества узлов в составе дерева A^R в двоичном коде с выхода регистра РгТКУА проходит через коммутатор 3 на вход дешифратора 4. Значение $N_T(A)$ в унитарном коде с выхода дешифратора 4 поступает на вход сдвигового регистра 5, где оно фиксируется по синхросигналу C_3, проходящему через элемент ИЛИ 6.

С выхода сдвигового регистра 5 значение текущего узла *j* поступает на вход ra_1 элемента А.ТС(у), на выходе rd_1 которого формируется значение поля ТС *j*-го узла дерева A^R $\left(TC^{\downarrow}\right)$. Сформированное значение с выхода rd_1 элемента А.ТС(у) поступает на первый вход схемы СОТСП. Также значение *j* поступает на вход *ra* элемента А.СП(у), на выходе *rd* которого формируется номер узла-предка рассматриваемого *j*-го узла дерева A^R. Сформированное значение номера узла-предка $СП^{\downarrow}$ в двоичном коде с выхода *rd* элемента А.СП(у) поступает на вход дешифратора 33. С выхода дешифратора 33 значение номера узла-предка в унитарном коде поступает на вход ra_0 элемента А.ТС(у), на выходе rd_0 которого формируется значения поля ТС узла-предка $\left(TC^{\uparrow}\right)$. Сформированное значение с выхода rd_0 поступает на второй вход схемы СОТСП. Также значение *j* поступает на вход *ra* элемента А.НС(у), на выходе *rd* которого формируется значение номера узла дерева B^R, предположительно *r*-изоморфного *j*-му. Значение с выхода *rd* элемента А.НС(у) в двоичном коде поступает на вход дешифратора 34. С выхода дешифратора 34 значение в унитарном коде подается на вход *ra* элемента В.СП(у), на выходе *rd* которого формируется номер узла-предка в дереве B^R $\left(\widetilde{СП}^{\downarrow}\right)$. С выхода *rd* элемента В.СП(у) значение $\widetilde{СП}^{\downarrow}$ поступает на вход элемента И 35, на выходе которого формируется значение признака $\tilde{\gamma}$, поступающее на третий вход схемы СОТСП. На выходе схемы СОТСП, как было рассмотрено выше, формируется обновленное значение $TC^{\uparrow\prime}$ поля ТС предка рассматриваемого узла. Сформированное значение проходит через коммутатор 24 и блок элементов ИЛИ 10 на вход *wd* элемента А.ТС(у). Значение $\widetilde{СП}^{\downarrow}$ с выхода *rd* элемента В.СП(у) проходит через коммутатор 25 и блок элементов ИЛИ 12 на вход *wd* элемента А.НС(у). Значение номера предка рассматриваемого узла $СП^{\downarrow}$ с выхода дешифратора 33 проходит через коммутатор 23 и блок элементов ИЛИ 11 на входы *wa* элементов А.ТС(у) и А.НС(у). Таким образом, на входах элементов А.ТС(у) и А.НС(у) сформированы обновленные значения полей ТС и НС предка текущего *j*-го узла. Поступление синхросигнала C_2, проходящего через элемент ИЛИ 13 на входы C_w элементов А.ТС(у) и А.НС(у), обеспечивает запись обновленных значений полей узлов дерева A^R. При этом синхросигнал C_2 не проходит через элемент И 36, оставляя значения полей ТС и НС наборов листьев дерева A^R неизменными. Также синхросигнал C_2 проходит через элемент задержки на синхровход сдвигового регистра 5, сдвигая единицу в хранящемся в нем битовом векторе в сторону младших разрядов: происходит переход к следующей итерации алгоритма. Спустя $N_T(A)$ синхросигналов C_2 формируются значения полей ТС и НС всех узлов дерева A^R.

Значения признаков δ^- узлов дерева A^R с выхода *d* элемента А.ТС(у) поступают на первые входы блока элементов запрета 37. На вторые входы блока элементов запрета поступает вектор, образованный прохождением значения $N_T(A)$ из регистра РгТКУА через дешифратор 38 и схему СМНП 2 (маскирующий неиспользуемые позиции узлов дерева). На

выходе блока элементов запрета 37 формируется вектор, нулевое значение в *i*-м бите которого говорит об отсутствии соответствия для *i*-го узла дерева, т.е., другими словами, отсутствии *r*-изоморфизма рассматриваемых деревьев. Сформированный вектор поступает на вход элемента И 31, на выходе которого формируется поразрядная конъюнкция элементов вектора. В общем случае дерево содержит как наборы листьев, так и узлы. Наличие узлов идентифицируется значением признака $\sigma = 1$, получаемым на выходе элемента ИЛИ 39. При $\sigma = 1$ результирующее значение φ_0 определяется конъюнкцией бит вектора с выхода блока элементов запрета 37, получаемой благодаря элементу И 31 и проходящей через элемент ИЛИ 32.

Возможен случай, когда дерево A^R состоит из единственного набора листьев. В таком случае $N_T(A) = 0$ и $\sigma = 0$. При этом элемент И 31 закрыт, а значение признака φ_0 определяется значением признака λ, проходящим через элементы запрета 30 и ИЛИ 32.

Временные диаграммы, поясняющие принцип работы схемы, приведены на рис. 30.

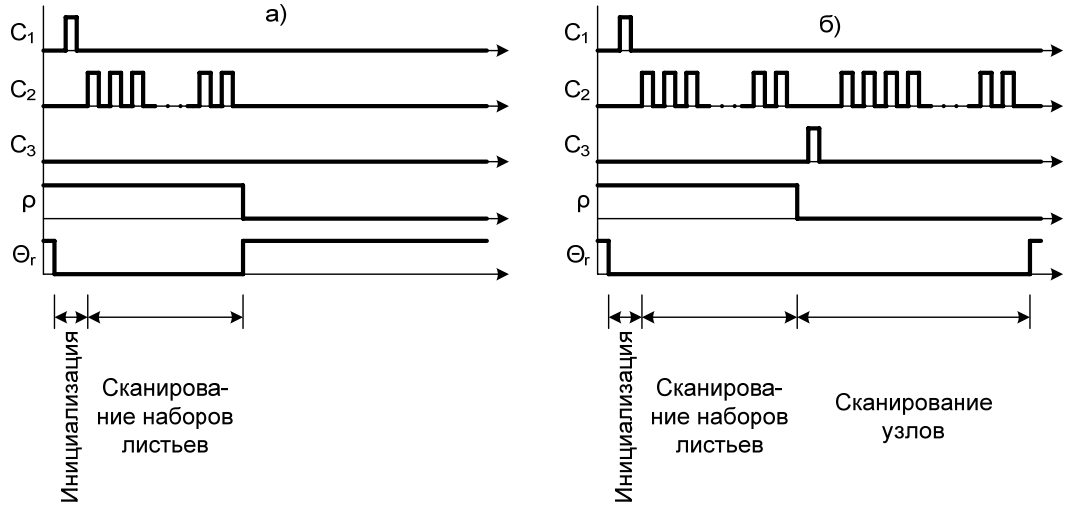


Рис. 30. Временные диаграммы работы схемы определения отношения нестрогого включения (а – досрочное завершение работы по $\lambda = 0$, б – полный цикл)

Время проверки *r*-изоморфизма пары деревьев, также как время выполнения рассмотренной выше операции определения ω-мощности дерева, может быть существенно различно (обработка может быть завершена с отрицательным результатом уже на втором этапе алгоритма за 1 такт работы устройства или же может потребоваться $N_T(B) + N_T(A)$ тактов для сканирования всех наборов листьев и узлов в случае присутствия *r*-изоморфизма), поэтому факт окончания выполнения операции идентифицируется по единичному значению признака готовности Θ_r.

По сравнению с программной реализацией выигрыш в скорости достигается за счет:

- параллельной инициализации значений полей ТС и НС узлов и наборов листьев дерева A^R;
- параллельного сравнения выбранного *j*-го набора листьев дерева B^R со всеми наборами листьев дерева A^R;
- параллельного сравнения компонентов битовых векторов, соответствующих наборам листьев;
- параллельного чтения значений $TC^{\uparrow}$ и $TC^{\downarrow}$ из разных портов ОСЭМД;
- параллельного чтения и записи значений различных полей (А.ТС(нл), А.НС(нл), А.НС(у) и т.д.) из/в разные элементы ОСЭМД.

Программная реализация предложенного алгоритма проверки *r*-изоморфизма предполагает попарное сравнение наборов листьев деревьев A^R и B^R (каждое сравнение наборов листьев требует порядка $O(L_{\max})$ действий) с последующим просмотром узлов дерева A^R снизу вверх. Ее асимптотическая временная сложность составляет величину порядка $O(N_L^2 L_{\max} + N_T) \simeq O(N_L^2 L_{\max})$. Предложенное аппаратное решение обладает асимптотической временной сложностью порядка $O(N_L + N_T) = O(N_R)$ благодаря параллельному сравнению компонент векторов наборов листьев и параллельному сравнению *j*-го набора листьев дерева B^R со всеми наборами листьев дерева A^R , т.е. имеет место выигрыш в скорости обработки в $\frac{N_L^2 L_{\max}}{N_R} \approx N_R L_{\max}$ раз.

Возможен также вариант схемы с матричной структурой расположения схем ССБВ, позволяющий проведение попарного сравнения всех наборов листьев за 1 такт. Дальнейшее сканирование узлов дерева A^R возможно только последовательно ввиду присутствия зависимостей по данным между узлами, принадлежащими к разным ярусам дерева. Подобный матричный вариант реализации характеризуется асимптотической временной сложностью порядка $O(N_T) \simeq O(N_R)$. Несмотря на совпадение асимптотик, подобный вариант будет в несколько раз быстрее рассмотренного выше ввиду того, что при рассмотрении асимптотик не учитываются коэффициенты. Так предлагаемый вариант реализации выдает результат не более чем за $k_1 N_L + k_2 N_T$ тактов, а матричный – за $k_1 + k_2 N_T$, где k_1 и k_2 – коэффициенты, опускаемые при рассмотрении асимптотик. Однако матричный вариант потребует N_L^2 схем ССБВ, что, ввиду немалой аппаратной сложности схемы, на данном этапе развития микроэлектроники является неоправданным.

Распространение типа соответствия от листьев к корню дерева и синтез подстановки изоморфизма возможно осуществлять параллельно для различных поддеревьев в составе обрабатываемого дерева A^R , что может в еще большей мере повысить быстродействие устройства, однако подобная аппаратная реализация потребует существенно больших аппаратных затрат и модификации используемого табличного представления *R*-выражений в виде деревьев.

10. Схемотехническая реализация операции удаления поддерева

Непосредственно после положительного исхода проверки отношения нестрогого включения у пары *R*-выражений (в случае соблюдения соотношения между ω-мощностями *R*-выражений) необходимо произвести операцию *u*- или *d*-подстановки. Первым шагом подстановки является удаление поддерева в составе объемлющего дерева B^R , *r*-изоморфного дереву A^R . Данная операция может быть реализована путем сканирования элементов дерева A^R , выяснения расположения их *r*-изоморфных эквивалентов в составе дерева B^R (используя вычисленные на предыдущем этапе значения полей НС) с последующей пометкой элементов как удаляемых (в случае полного соответствия) с использованием поля УЭ или корректировки значений их полей (в случае частичного соответствия). На следующем шаге выполнения операции подстановки элементы ОСЭМД, помеченные единичным значением поля УЭ, считаются свободными и могут быть использованы для хранения элементов подставляемого дерева.

На рис. 31 приведена схема, производящая пометку удаляемых элементов дерева B^R и модификации полей оставшихся элементов в соответствии с приведенным в [10] алгоритмом.

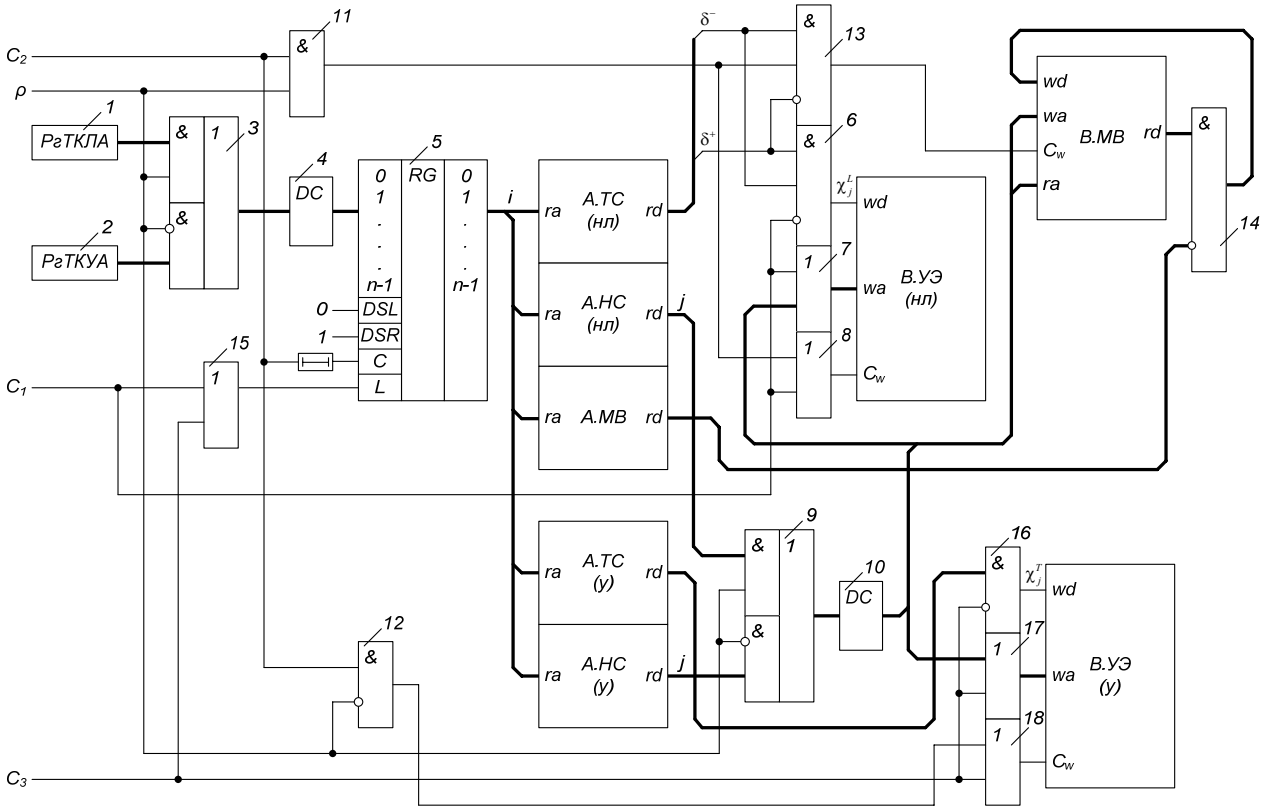


Рис. 31. Схема удаления поддерева (СУП)

Регистры 1 и 2 используются для хранения текущего количества листьев и текущего количества узлов соответственно. Через коммутатор 3 и дешифратор 4 значение одного из регистров в зависимости от выполняемого этапа алгоритма сохраняется в сдвиговом регистре 5. Значение, хранящееся в регистре 5, является адресом для чтения полей ТС, НС, МВ наборов листьев и полей ТС и НС узлов дерева A^R. Элементы 6–8 используются при инициализации значений поля УЭ наборов листьев дерева B^R. Коммутатор 9 используется на разных этапах работы устройства для выявления номера предка текущего набора листьев или узла. Дешифратор 10 используется для преобразования номера узла или набора листьев из двоичного представления в унитарное с целью задания адреса изменяемого элемента векторов УЭ узлов и наборов листьев или номера корректируемого поля МВ. Элемент И 11 закрывает прохождение синхросигнала C_2 на вход элементов В.УЭ(нл) и В.МВ при обработке узлов. Аналогично, элемент запрета 12 закрывает прохождение синхросигнала C_2 на вход элемента В.УЭ(у) при обработке наборов листьев. Элемент запрета 13 обеспечивает прохождение синхросигнала C_2 на вход элемента В.МВ только в случае наличия частичного соответствия. Блок элементов запрета 14 предназначен для модификации поля МВ набора листьев с частичным соответствием. Элемент ИЛИ 15 служит для инициализации значения регистра 5 по синхросигналам C_1 и C_3. Элементы 16–18 предназначены для инициализации значений поля УЭ узлов дерева B^R.

Представленная схема работает следующим образом. На первом этапе ее работы осуществляется обработка наборов листьев, для этого производится установка сигнала $\rho = 1$. Единичное значение сигнала ρ обеспечивает прохождение значения $N_L(A)$ с выхода регистра РгТКЛА 1 через коммутатор 3 на вход дешифратора 4 и формирует на входе сдвигового регистра 5 битовый вектор из нулей и единицы в $N_L(A)$-ой позиции. Синхросигнал C_1, проходящий через элемент ИЛИ 15, обеспечивает запись битового

вектора в сдвиговый регистр 5, а также, благодаря элементам запрета 6, блоку элементов ИЛИ 7 и элементу ИЛИ 8, обеспечивает инициализацию полей УЭ дерева B^R нулевыми значениями.

С выхода сдвигового регистра 5 значение *i* номера текущего рассматриваемого набора листьев поступает на входы *ra* элементов A.ТС(нл), A.НС(нл) и A.МВ, на выходах *rd* которых формируются соответствующие значения полей *i*-го набора листьев. Значения признаков δ^+ и δ^- с выхода элемента A.ТС(нл) формируют на выходе элемента запрета 6 единичное значение только в случае наличия полного соответствия $\left(\delta^+ = \delta^- = 1,\right.$ синхросигнал C_1 при этом входе имеет в данный момент нулевое значение), которое поступает на вход *wd* элемента В.УЭ(нл): в этом случае *r*-изоморфный эквивалент рассматриваемого *i*-го набора листьев дерева A^R помечается как удаляемый. Значение номера *r*-изоморфного эквивалента *i*-го набора листьев дерева A^R с выхода *rd* элемента A.НС(нл) проходит через коммутатор 9 благодаря единичному значению сигнала ρ на вход дешифратора 10. Значение номера набора листьев в дереве B^R, *r*-изоморфного *i*-му набору листьев дерева A^R, в унитарном коде с выхода дешифратора 10 проходит через блок элементов ИЛИ 7 и поступает на вход *wa* элемента В.УЭ(нл). Синхроимпульс C_2 проходит через элемент И 11 благодаря единичному значению сигнала ρ и через элемент ИЛИ 8 поступает на вход C_w элемента В.ЭУ: происходит запись значения поля УЭ рассматриваемого набора листьев. При этом не происходит модификации значений элемента В.УЭ(у), т.к. синхросигнал C_2 не проходит через элемент запрета 12. Спустя интервал времени, формируемый элементом задержки, синхроимпульс C_2 поступает на синхровход сдвигового регистра 5, что обеспечивает сдвиг его содержимого на одну позицию в сторону младших разрядов, т.е. переход к новой итерации работы алгоритма. По пришествии $N_L(A)$ синхроимпульсов C_2 обработка наборов листьев завершается.

В случае, если между парой наборов листьев в составе обрабатываемых деревьев присутствует отношение неполного соответствия, необходимо произвести модификацию поля МВ набора листьев в дереве B^R (удалить из него вершины, входящие в набор листьев в дереве A^R). Указанное действие осуществляется следующим образом. В случае неполного соответствия $\delta^- = 1$, $\delta^+ = 0$: элемент запрета 13 открывается для прохождения синхросигнала C_2 на вход C_w элемента В.МВ. При этом значение поля МВ рассматриваемого набора листьев дерева A^R с выхода *rd* элемента A.МВ поступает на инверсные входы блока элементов запрета 14. Значение с выхода дешифратора 10, поступающее на входы *ra* и *wa* элемента В.МВ, обеспечивает на его выходе *rd* значение поля МВ набора листьев в дереве B^R, *r*-изоморфного рассматриваемому. Битовый вектор с выхода *rd* элемента В.МВ поступает на прямые входы блока элементов запрета 14, на выходах которого формируется обновленное значение поля МВ набора листьев дерева B^R, которое поступает на вход *wd* элемента В.МВ. Таким образом, по пришествии синхросигнала C_2 происходит запись обновленного значения поля МВ набора листьев дерева B^R.

По завершении обработки наборов листьев сигнал ρ устанавливается в нулевое значение. При этом значение $N_T(A)$ поступает через коммутатор 3 и дешифратор 4 на вход сдвигового регистра 5. По пришествии синхросигнала C_3, проходящего через элемент ИЛИ 15 на вход сдвигового регистра 5, в него производится запись значения $N_T(A)$ в унитарной форме. Также синхросигнал C_3, поступающий на входы элемента запрета 16, блока

элементов ИЛИ 17 и элемента ИЛИ 18 обеспечивает инициализацию полей УЭ узлов дерева B^R нулевыми значениями.

С выхода сдвигового регистра значение номера *i*-го рассматриваемого узла дерева *A* поступает на входы *ra* элементов A.ТС(у) и A.НС(у), на выходах *rd* которых формируются значения соответствующих полей рассматриваемого узла. Значения с выхода *rd* элемента A.ТС(у) формируют на выходе элемента запрета 16 единичное значение только в случае наличия полного соответствия $\left(\delta^+ = \delta^- = 1, C_3 = 0\right)$, сформированное значение признака УЭ узла дерева поступает на вход *wd* элемента В.УЭ(у). Значение номера узла в дереве B^R, *r*-изоморфного рассматриваемому *i*-му узлу, проходит через коммутатор 9, дешифратор 10 и блок элементов ИЛИ 17 на вход *wa* элемента В.УЭ, обеспечивая формирование номера узла в унитарном коде. Синхросигнал C_2, проходящий через элемент запрета 12 благодаря нулевому значению сигнала ρ и через элемент ИЛИ 18 на вход C_w элемента В.УЭ(у), обеспечивает запись признака УЭ узла дерева B^R, *r*-изоморфного рассматриваемому. При этом не происходит модификации значений элементов В.УЭ(нл) и В.МВ, т.к. прохождение синхросигнала C_2 на их входы C_w блокируется элементом И 11. По истечении интервала времени, формируемого элементом задержки, синхросигнал C_2 поступает на синхровход сдвигового регистра 5, что обеспечивает сдвиг его двоичного содержимого вправо на 1 разряд и переход к следующей итерации алгоритма. По пришествии $N_T(A)$ синхроимпульсов C_2 обработка наборов листьев завершается.

Таким образом, в результате описанных выше действий производится пометка элементов дерева B^R, готовых к удалению, с использованием полей УЭ, а также модификация значений полей МВ наборов листьев дерева B^R, находящихся в неполном соответствии с наборами листьев дерева A^R.

Временные диаграммы работы схемы в целом аналогичны изображенным на рис. 30 с тем отличием, что по окончании обработки выдается единичное значение сигнала Θ_d, а не Θ_r.

Программная реализация рассмотренного выше алгоритма удаления поддерева требует порядка $O\left(N_L(A)\right)$ шагов на просмотр наборов листьев, порядка $O\left(N_T(A)\right)$ шагов на просмотр узлов дерева и, в худшем случае (имеется неполное соответствие между наборами листьев), порядка $O\left(L_{\max}\right)$ шагов на модификацию поля МВ набора листьев, находящегося в отношении неполной эквивалентности (из свойств *R*-выражений [10] следует, что таких наборов не может быть более одного). Таким образом, суммарное время обработки составляет величину порядка $O\left(N_L(A)+N_T(A)+L_{\max}\right)=O\left(N_R(A)+L_{\max}\right)\simeq O\left(L_{\max}\right)$, т.к. $N_R(A)\ll L_{\max}$. Рассмотренное выше аппаратное решение также производит последовательный просмотр элементов дерева, однако модификация поля МВ при этом выполняется параллельно во времени для всех бит благодаря элементу запрета 14, т.е. временная сложность аппаратно-ориентированной обработки составляет величину порядка $O\left(N_R(A)\right)$. Таким образом, имеет место выигрыш во времени обработки приблизительно в $\frac{N_R(A)+L_{\max}}{N_R(A)}\approx\frac{L_{\max}}{N_R(A)}$ раз. При этом чтение и запись значений полей элементов обрабатываемых деревьев производится параллельно во времени, что также обеспечивает некоторое ускорение обработки.

Модификацию значений полей УЭ элементов дерева B^R теоретически возможно производить параллельно во времени для каждого элемента дерева A^R, т.к. они не связаны

между собой зависимостями по данным, однако аппаратная сложность такого решения будет приблизительно в $N_R(A)$ раз больше, что на современном уровне развития микроэлектроники неоправданно. При этом также потребуется модификация электронной модели дерева. Кроме того, теоретически возможно совмещение во времени обработки узлов и наборов листьев дерева A^R, однако это потребует дублирования сдвиговых регистров и дешифраторов, что также приведет к росту аппаратной сложности схемы.

11. Схемотехническая реализация операции вставки поддерева

Вторым этапом операции подстановки является вставка дерева C^R в дерево B^R на место удаленного поддерева, *r*-изоморфного дереву A^R. При этом операция распадается на две стадии. Первая стадия заключается в копировании элементов табличного представления дерева C^R в дерево B^R с соблюдением требований № 2 и 4 корректности дерева (требования № 3 и 6 корректности дерева удовлетворяются после выполнения операций раскрытия скобок и удаления пустых позиций, рассмотренных ниже). На второй стадии производится настройка связей между скопированными элементами и связи скопированного дерева с уже присутствовавшими до этого элементами дерева B^R (корректировка значений полей СП).

Схема, реализующая предложенный в [10] алгоритм, приведена на рис. 32.

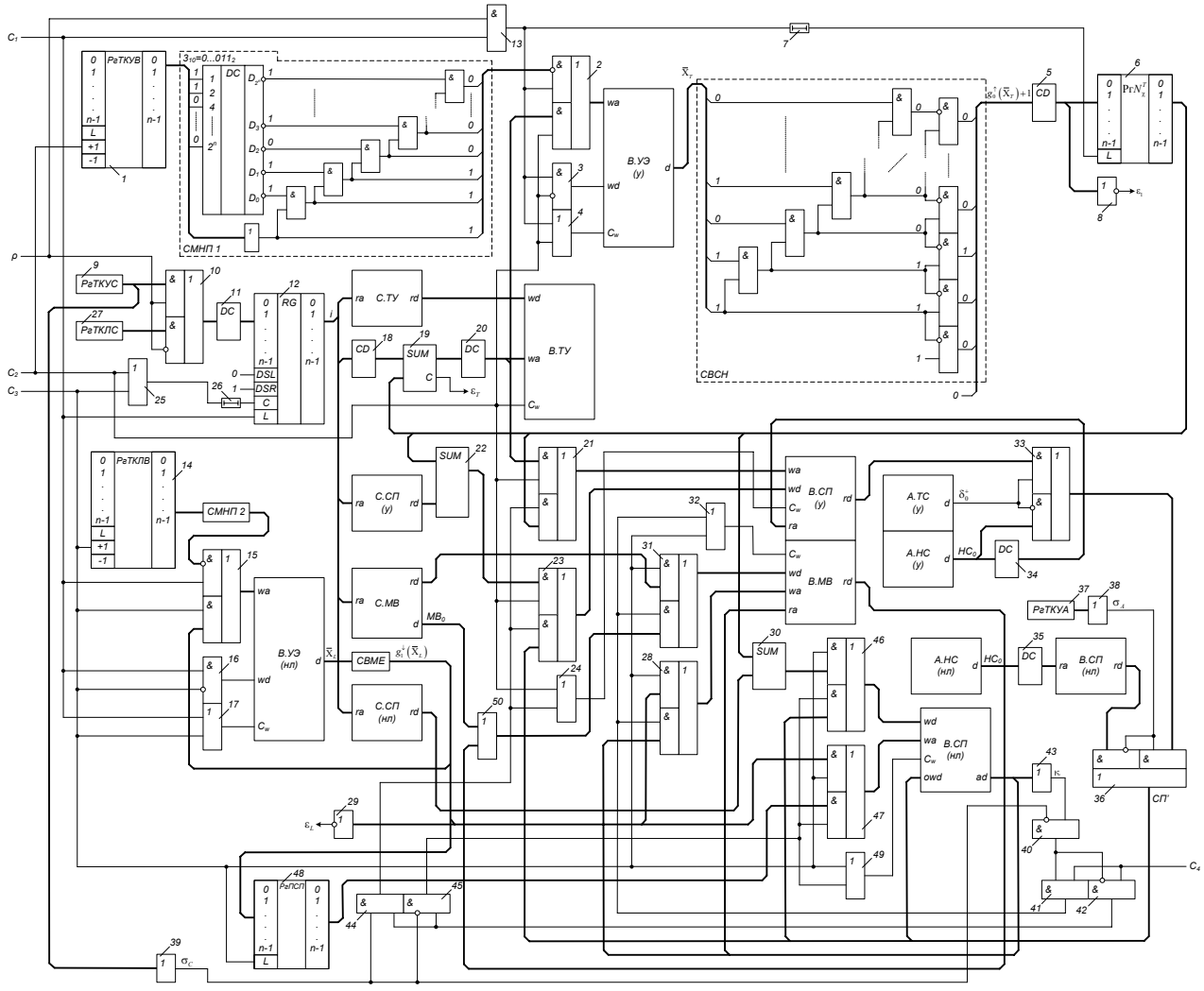


Рис. 32. Схема вставки поддерева (СВП)

Регистр 1 хранит текущее количество узлов дерева B^R и инкрементируется по мере добавления в дерево B^R новых узлов. Коммутатор 2 используется для выбора значения вектора, подаваемого на вход элемента В.УЭ(у) в качестве адреса записи. Элементы 3 и 4 используются при инициализации значений вектора В.УЭ(у). Шифратор 5 используется для преобразования значения номера свободной позиции с выхода схемы СВСН из унитарного в двоичное представление, сохраняющегося в регистре 6. Элемент задержки 7 обеспечивает запись значения в регистр 6 только по окончании его формирования на выходе схемы СВСН. Элемент ИЛИ-НЕ 8 используется для формирования сигнала ошибки ε_1. Регистры 9 и 27 используются для хранения числа узлов и наборов листьев дерева C^R соответственно. Коммутатор 10 в совокупности с дешифратором 11 обеспечивают запись в сдвиговый регистр 12 значения текущего числа элементов (наборов листьев или узлов) в унитарном коде в зависимости от этапа обработки. Элемент И 13 запрещает прохождение синхросигнала C_1 при инициализации значения В.УЭ(нл) на входы элемента В.УЭ(у) и регистра 6. Регистр 14 хранит текущее количество наборов листьев дерева B^R, инкрементируемое по мере добавления новых наборов листьев. Коммутатор 15 в совокупности с элементами 16 и 17 обеспечивает инициализацию значения вектора УЭ наборов листьев дерева B^R (в ходе дальнейшей обработке значения вектора УЭ изменяются согласно алгоритму вставки поддерева [10]). Шифратор 18 используется для преобразования значения текущего рассматриваемого элемента из унитарного кода в двоичный. Сумматор 19 обеспечивает формирование на его выходе значения $N_T^{\chi} + i$ очередной свободной позиции и признака ошибки ε_T. Дешифратор 20 преобразует значение $N_T^{\chi} + i$ в унитарный код для его дальнейшего использования в качестве адреса записи. Коммутаторы 21, 23, 28, 31, 46 и 47 используется для выбора значения элемента дерева B^R модифицируемого элемента в зависимости от этапа алгоритма и особенностей обработки, рассмотренных ниже. Сумматор 22 обеспечивает формирование обновленного значения поля СП копируемого элемента. Элемент ИЛИ 24 служит для прохождения синхросигнала записи на вход элемента В.СП(у). Элементы 25 и 26 обеспечивают прохождения синхросигналов C_2 и C_3 на вход регистра 12. Элемент ИЛИ-НЕ 29 служит для формирования признака ошибки ε_L. Сумматор 30 используется для корректировки значения поля СП копируемого набора листьев. Элемент ИЛИ 32 обеспечивает прохождение синхросигнала на вход элемента В.МВ, активирующего модификацию поля МВ в зависимости от ситуации. Коммутатор 33 обеспечивает формирование на своем выходе значения предка корня подставляемого дерева в зависимости от типа соответствия. Дешифратор 34 обеспечивает преобразование номера узла, *r*-изоморфного корневому узлу дерева A^R, из двоичного кода в унитарный. Дешифратор 35 обеспечивает преобразование номера набора листьев, *r*-изоморфного нулевому набору листьев дерева A^R, из двоичного кода в унитарный. Коммутатор 36 обеспечивает формирование на своем выходе значения номера узла-предка для корня подставляемого дерева. Регистр 37 хранит значение текущего числа узлов дерева A^R, используемое для формирования значения признака σ_A на выходе элемента ИЛИ 38, подаваемого на вход коммутатора 36. Элемент ИЛИ 39 используется для формирования признака σ_C. Элемент ИЛИ 43 обеспечивает формирование признака κ, используемого в совокупности с элементами 40–42, 44 и 45 для управления прохождением синхросигнала C_4. Регистр-защелка 48 используется для хранения значения первой свободной позиции, формируемого на выходе схемы СВМЕ. Элемент ИЛИ 49 обеспечивает прохождение синхросигнала записи к элементу В.СП(нл) в зависимости от обрабатываемой ситуации. Блок элементов ИЛИ 50 используется при объединении уже имеющегося набора листьев с единственным набором листьев, входящим в состав подставляемого дерева C^R.

Схема работает следующим образом. В начале обработки сигнал ρ устанавливается в единичное значение, что подготавливает схему к обработке узлов. Значение $N_T(B)$, хранящееся в регистре РгТКУВ 1, поступает на вход схемы маскировки неиспользуемых позиций СМНП 1, на выходе которой формируется вектор вида $\underbrace{11...1}_{N_T(B)}00...0$. Единичные биты вектора соответствуют занятым позициям, нулевые – свободным (элементы, помеченные на предыдущем шаге как удаленные, пока считаются занятыми). Сформированный вектор поступает на вход коммутатора 2. По пришествии синхросигнала C_1, проходящего через элемент И 13, открытый единичным значением сигнала ρ, и открывающего коммутатор 2, на входе *wa* элемента В.УЭ(у) оказывается сформирован вектор вида $\underbrace{00...0}_{N_T(B)}11...1$. На входе *wd* элемента В.УЭ(у) при этом благодаря элементу запрета 3 оказывается единичное значение (синхросигнал C_2 в этот момент времени имеет нулевое значение). Также синхросигнал C_1 проходит через элемент ИЛИ 4 на вход C_w элемента В.УЭ(у), что обеспечивает запись единичных значений в элементы, являющиеся свободными. Таким образом, единичное значение в *i*-м элементе В.УЭ(у) свидетельствует о том, что элемент либо не был использован до этого, либо был помечен как удаленный, т.е., другими словами, может быть использован для приема и хранения узлов подставляемого дерева C^R (далее будем именовать такие позиции свободными). Логика пометки свободных позиций поясняется на рис. 33.

Элемент помечен как удаленный	*Элемент занят/свободен*	*Результат*
да	*занят*	*1*
да	*свободен*	*
нет	*занят*	*0*
нет	*свободен*	*1*

УЭ$_i$=x	*СМНП$_i$=y*	*f*
1	*1*	*1*
1	*0*	*
0	*1*	*0*
0	*0*	*1*

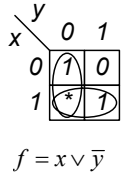


Рис. 33. Синтез логической функции для определения позиций, пригодных к приему элементов дерева C^R (звездочкой помечена ситуация, которая не может встретиться на практике, т.к. свободный элемент за пределами младших $N_T(B)$ позиций не может быть помечен как удаленный)

Вектор свободных элементов $\overline{\mathrm{X}}_T$ с выхода *d* элемента В.УЭ(у) поступает на вход схемы выделения старшего нуля СВСН, на выходе которой формируется вектор из нулей и единицы в позиции старшего нуля в векторе $\overline{\mathrm{X}}_T$. Старший бит сформированного вектора отбрасывается, а в качестве младшего бита добавляется ноль, чем фактически обеспечивается инкремент значения $g_0^{\uparrow}\left(\overline{\mathrm{X}}_T\right)$ в унитарном коде, после чего вектор поступает на вход шифратора 5. Сформированное на выходе шифратора 5 значение $N_{\chi}^T = g_0^{\uparrow}\left(\overline{\mathrm{X}}_T\right)+1$ в двоичном коде поступает на вход регистра РгN_{χ}^T 6. Элемент задержки 7 обеспечивает появление синхросигнала C_1 на синхровходе регистра 6 по окончании формирования значения N_{χ}^T на выходе шифратора 5 (элемент И 13 открыт для прохождения синхросигнала единичным значением сигнала ρ): сформированное значение записывается в регистр. Возможна ситуация, когда вектор на выходе схемы СВСН состоит из одних нулей (все позиции в табличном представлении дерева заняты). В таком случае на выходе элемента ИЛИ-НЕ 8 будет сформировано единичное значение признака ε_1, свидетельствующее о невозможности выполнения операции: работа схемы на этом прекращается.

Значение $N_T(C)$ с выхода регистра РгТКУС 9 проходит через коммутатор 10, открытый единичным значением сигнала ρ, дешифратор 11 и поступает на вход сдвигового регистра 12 в унитарном коде, где фиксируется по пришествии синхроимпульса C_1 на вход регистра 12.

Значение $N_L(B)$ с выхода регистра РгТКЛВ 14 поступает на вход схемы маскировки неиспользуемых позиций СМНП 2, на выходе которой формируется вектор $\underbrace{11...1}_{N_L(B)}00...0$. Синхросигнал C_1, аналогично рассмотренной выше последовательности действий по формированию вектора $\overline{X}_T$, в совокупности с коммутатором 15, элементом запрета 16 и элементом ИЛИ 17 обеспечивает формирование и сохранение в элементе В.УЭ(нл) вектора $\overline{X}_L$.

С выхода сдвигового регистра 12 значение i номера рассматриваемого узла T_i^C поступает на вход *ra* элемента С.ТУ, на выходе *rd* которого формируется значение поля ТУ узла T_i^C, поступающее на вход *wd* элемента В.ТУ. Также значение i с выхода сдвигового регистра 12 в унитарном коде поступает на вход шифратора 18, преобразующего его в двоичный код. С выхода шифратора 18 значение i поступает на первый вход сумматора 19, при этом на второй вход сумматора 19 с выхода регистра 6 подается значение N_χ^T. В результате сложения на первом выходе сумматора 19 формируется значение $i+N_\chi^T$ номера текущей свободной позиции в дереве B^R в двоичном коде, поступающее на вход дешифратора 20. Если очередная свободная позиция находится за пределами $N_{T_{\max}}$ (другими словами, свободных позиций нет), возникает переполнение сумматора 19 и на его втором выходе появляется единичное значение переноса ε_T, работа устройства на этом прекращается. С выхода дешифратора 20 значение текущей свободной позиции в дереве B^R поступает на вход *wa* элемента В.ТУ, на вход коммутатора 2 и на вход коммутатора 21. Также значение i с выхода сдвигового регистра 12 поступает на вход *ra* элемента С.СП(у), на выходе *rd* которого формируется значение поля СП узла T_i^C, поступающее на первый вход сумматора 22. На второй вход сумматора 22 с выхода регистра 6 при этом подается значение N_χ^T. В результате суммирования на выходе сумматора 22 формируется обновленное значение поля СП′ $\left(N_\chi^T + СП_i\right)$ узла, вставляемого в дерево B^R, которое подается на первый вход коммутатора 23. При появлении синхросигнала C_2 он проходит на вход C_w элемента В.ТУ, что инициирует запись (копирование) значения поля ТУ из дерева C^R в текущую свободную позицию дерева B^R. Также синхросигнал C_2 открывает коммутатор 21 для прохождения значения $i+N_\chi^T$ с выхода дешифратора 20 на вход *wa* элемента В.СП(у) и открывает коммутатор 23 для прохождения значения $N_\chi^T + СП_i$ с выхода сумматора 22 на вход *wd* элемента В.СП(у). Проходя через элемент ИЛИ 24, синхросигнал C_2 поступает на вход C_w элемента В.СП(у), что инициирует запись обновленного значения поля СП копируемого в дерево B^R узла. Также синхросигнал C_2 открывает коммутатор 2 для прохождения значения $i+N_\chi^T$ с выхода дешифратора 20 на вход *wa* элемента В.УЭ(у), формирует на выходе элемента запрета 3 нулевое значение, поступающее на вход *wd* элемента В.УЭ(у), и, проходя через элемент ИЛИ 4 на вход C_w элемента В.УЭ(у), обеспечивает сброс $\left(i+N_\chi^T\right)$-го бита в векторе $\overline{X}_T$: $\left(i+N_\chi^T\right)$-ая позиция в табличном

представлении узлов дерева B^R теперь считается занятой. Также синхросигнал C_2, поступающий на вход регистра РгТКУВ 1, обеспечивает инкремент значения $N_T(B)$. Далее, синхросигнал C_2, проходящий через элемент ИЛИ 25, через промежуток времени, определяемый элементом задержки 26, появляется на синхровходе сдвигового регистра 12, что обеспечивает сдвиг его содержимого в сторону младших разрядов, т.е. переход в новой итерации работы алгоритма. После появления $N_T(C)$ синхроимпульсов C_2 (в случае отсутствия ошибок) осуществляется копирование всех узлов дерева C^R в дерево B^R и корректировка значений их полей СП.

После этого производится переключение сигнала ρ в нулевое значение (осуществляется переход к обработке наборов листьев). Нулевое значение сигнала ρ открывает коммутатор 10 для прохождения значения $N_L(C)$ с выхода регистра РгТКЛС 27 через дешифратор 11 на вход сдвигового регистра 12. Появление синхросигнала C_1, поступающего на вход сдвигового регистра 12, обеспечивает запись значения $N_L(C)$ в сдвиговый регистр 12. При этом элемент И 13, закрытый нулевым значением сигнала ρ, закрыт для прохождения синхросигнала C_1 на входы коммутатора 2, элемента запрета 3, элемента ИЛИ 4 и регистра РгN_χ^T 6 (значения, хранимые элементом В.УЭ(у) и регистром РгN_χ^T 6 остаются прежними). Синхросигнал C_1 также проходит на входы коммутатора 15, элемента запрета 16 и элемента ИЛИ 17, чем обеспечивает повторную инициализацию элемента В.УЭ(нл) значениями вектора $\overline{\mathrm{X}}_L$ (аналогично рассмотренному выше). Значение вектора $\overline{\mathrm{X}}_L$ с выхода *d* элемента В.УЭ(нл) поступает на вход схемы выделения младшей единицы СВМЕ, на выходе которой формируется значение младшей свободной позиции $g_1^{\downarrow}(\overline{\mathrm{X}}_L)$ в унитарном коде. Сформированное значение поступает на вход коммутатора 15, на вход коммутатора 28, на вход коммутатора 47, на вход регистра РгПСП 48 и на вход элемента ИЛИ-НЕ 29. Если все биты вектора $\overline{\mathrm{X}}_L$ имеют нулевые значения (все позиции наборов листьев дерева B^R заняты), на выходе схемы СВМЕ также будет сформирован вектор вида 00…0, который обеспечит на выходе элемента ИЛИ-НЕ 29 единичное значение признака ошибки ε_L, работа устройства на этом прекращается. Значение *i* номера текущего рассматриваемого набора листьев L_i^C с выхода сдвигового регистра 12 поступает на входы *ra* элементов С.МВ и С.СП(нл), на выходах *rd* которых формируются значения полей МВ и СП текущего набора листьев соответственно. Значение с выхода *rd* элемента С.СП(нл) поступает на первый вход сумматора 30, на втором входе при этом присутствует значение N_χ^T, поступающее на него с выхода регистра РгN_χ^T 6. На выходе сумматора 30 формируется обновленное значения поля СП текущего набора листьев (с учетом того, что номер узла-предка, на который ссылается рассматриваемый набор листьев, после копирования в дерево B^R был увеличен на величину N_χ^T), которое подается на вход коммутатора 46. Значение с выхода *rd* элемента С.МВ подается на вход коммутатора 31. Появление синхросигнала C_3, поступающего на вход регистра РгТКЛВ 14, обеспечивает инкремент значения $N_L(B)$. Также синхросигнал C_3 открывает коммутатор 15 для прохождения значения $g_1^{\downarrow}(\overline{\mathrm{X}}_L)$ с выхода схемы СВМЕ на вход *wa* элемента В.УЭ(нл), формирует на выходе элемента запрета 16 нулевое значение, поступающее на вход *wd* элемента В.УЭ(нл), и, проходя через элемент ИЛИ 17 на вход C_w элемента В.УЭ(нл), обеспечивает запись обновленных данных в элемент В.УЭ(нл): элемент

$\chi_k^L,\ k = g_1^{\downarrow}\left(\overline{\mathrm{X}}_L\right)$ вектора $\overline{\mathrm{X}}_L$ помечается как занятый. Также синхросигнал C_3 обеспечивает запись текущего значения $g_1^{\downarrow}\left(\overline{\mathrm{X}}_L\right)$ с выхода схемы СВМЕ в регистр РгПСП 48. Также синхросигнал C_3 открывает коммутаторы 46 и 47 для прохождения значений с выхода сумматора 30 и выхода схемы СВМЕ на входы *wd* и *wa* элемента В.СП(нл) соответственно и, проходя через элемент ИЛИ 49 на вход C_w элемента В.СП(нл), обеспечивает запись обновленного значения поля СП рассматриваемого набора листьев. Также синхросигнал C_3 открывает коммутатор 31 для прохождения значения с выхода *rd* элемента С.МВ на вход *wd* элемента В.МВ, коммутатор 28 для прохождения значения с выхода схемы СВМЕ на вход *wa* элемента В.МВ и, проходя через элемент ИЛИ 32 на вход C_w элемента В.МВ, обеспечивает запись (копирование) поля МВ текущего набора листьев. Далее, спустя интервал времени, формируемый элементом задержки 26, синхросигнал C_3, прошедший через элемент ИЛИ 25, попадает на синхровход сдвигового регистра 12, что обеспечивает сдвиг его двоичного содержимого в сторону младших разрядов, т.е. переход к новой итерации работы алгоритма. После появления $N_L\left(C\right)$ синхроимпульсов C_3 (при условии отсутствия ошибок) все наборы листьев дерева C^R оказываются скопированными в дерево B^R.

Заключительным этапом обработки является настройка взаимосвязи корня дерева C^R (его значения поля СП) с родительским узлом в дереве B^R. Для этого производится определение значения СП'. Значение поля НС корневого узла дерева A^R (в случае его наличия) с выхода *d* элемента А.НС(у) поступает на первый вход коммутатора 33 и на вход дешифратора 34. С выхода дешифратора 34 значение номера узла, *r*-изоморфного корню дерева A^R, в унитарном коде поступает на вход *ra* элемента В.СП(у), на выходе *rd* которого формируется значение номера его предка, поступающее на второй вход коммутатора 33. Значение на выходе коммутатора 33 определяется типом соответствия корневого элемента дерева A^R $\left(\delta_0^+\right)$, поступающим с выхода *d* элемента А.ТС(у) на входы коммутатора 33. Значение поля НС единственного набора листьев дерева A^R (в случае отсутствия узлов) с выхода *d* элемента А.НС(нл) проходит через дешифратор 35 на вход *ra* элемента В.СП(нл), на выходе *rd* которого формируется значение. Значения с выхода коммутатора 33 и с выхода *rd* элемента В.СП(нл) поступают на первый и второй входы коммутатора 36 соответственно. Значение $N_T\left(A\right)$ с выхода регистра РгТКУА 37 поступает на вход элемента ИЛИ 38, на выходе которого формируется значение признака σ_A отсутствия узлов в составе дерева A^R, поступающее на входы коммутатора 36. Искомое значение СП', формируемое на выходе коммутатора 36, определяется значением признака σ_A. Значение с выходе коммутатора 36 поступает на вход *owd* элемента В.СП(нл) и на второй вход коммутатора 23.

В общем случае дерево C^R представлено как узлами, так и наборами листьев $\left(N_T\left(C\right)>0\right)$. При этом значение с выхода регистра РгТКУС 9, поступающее на вход элемента ИЛИ 39, формирует на его выходе единичное значение признака σ_C, поступающее на вход элемента запрета 40. На выходе элемента запрета 40 при этом формируется нулевое значение, закрывающее элемент И 41 и открывающее элемент запрета 42 для прохождения синхросигнала C_4. Единичное значение синхросигнала C_4, проходящего через элемент запрета 42 и затем через элемент И 44, открытый единичным сигналом σ_C с выхода элемента ИЛИ 39, обеспечивает формирование на входе *wa* элемента В.СП(у) значения N_χ^T, проходящего с выхода регистра РгN_χ^T 6 через коммутатор 21, значения СП' на входе *wd* элемента В.СП(у), проходящего с выхода коммутатора 36 через коммутатор 23, и

единичного значения на выходе элемента ИЛИ 24, поступающего на вход C_w элемента В.СП(у): производится запись обновленного значения СП' узла, бывшего корнем дерева C^R до проведения операции вставки поддерева. При этом синхросигнал C_4 не проходит через элемент запрета 45, закрытый единичным значением признака σ_C с выхода элемента ИЛИ 39.

Особым случаем, влияющим на логику работу схемы, является ситуация, когда дерево C представлено единственным набором листьев. При этом на выходе элемента ИЛИ 39 присутствует нулевое значение признака σ_C, а значение на выходе элемента запрета 40 определяется значением признака κ присутствия набора листьев у узла с номером СП'. Определение значения признака κ производится следующим образом. Значение на входе *owd* элемента В.СП(нл) инициирует ассоциативный поиск набора листьев, значение поля СП которого совпадает с СП'. В результате выполнения проверки возможны две различные ситуации, влияющие на дальнейшую обработку.

В случае отсутствия такого набора на выходе *ad* элемента В.СП(нл) формируется вектор 00…0, который поступает на вход элемента ИЛИ 43 и формирует на его выходе нулевое значение признака κ. Значение признака $\kappa = 0$ поступает на вход элемента запрета 40 и формирует на его выходе нулевое значение, закрывающее элемент И 41 и открывающее элемент запрета 42 для прохождения синхросигнала C_4 . При этом синхросигнал C_4 проходит через элемент запрета 42 и элемент запрета 45, открытый нулевым значением признака σ_C с выхода элемента ИЛИ 39, на вход коммутатора 46, что открывает его для прохождения значения СП' с выхода коммутатора 36 на вход *wd* элемента В.СП(нл). Аналогично, синхросигнал C_4 , прошедший через элементы запрета 42 и 45, открывает коммутатор 47 для прохождения значения с выхода регистра РгПСП 48 на вход *wa* элемента В.СП(нл) и, проходя через элемент ИЛИ 49 и попадая на вход C_w элемента В.СП(нл), инициирует запись обновленного значения поля СП единственного набора листьев, скопированного из дерева C^R в дерево B^R. При этом синхросигнал C_4 не проходит через элемент И 44, закрытый нулевым значением признака σ_C с выхода элемента ИЛИ 39.

Если же на выходе *ad* элемента В.СП(нл) присутствует ненулевое значение (у узла с номером СП' уже есть дочерний набор листьев), на выходе элемента ИЛИ 43 будет сформировано единичное значение признака κ. Благодаря тому, что $\sigma_C = 0$ и $\kappa = 1$, на выходе элемента запрета 40 будет сформировано единичное значение, открывающее элемент И 41 и закрывающее элемент запрета 42 для прохождения синхросигнала C_4. При этом синхросигнал C_4 , проходящий через элемент И 41 на входы коммутаторов 28 и 31, открывает их. На вход *wa* элемента В.МВ при этом через открытый коммутатор 28 проходит значение с выхода *ad* элемента В.СП(нл) (фактически, номер найденного дочернего набора листьев). Значение поля МВ единственного набора листьев дерева C поступает на первые входы блока элементов ИЛИ 50 с выхода *d* элемента С.МВ, при этом на вторые входы блока элементов ИЛИ 50 подается значение поля МВ найденного дочернего набора листьев с выхода *rd* элемента В.МВ, в результате чего на выходе блока элементов ИЛИ 50 формирует обновленное значение поля МВ (единственный набор листьев дерева C^R добавлен в найденный дочерний набор листьев). С выхода блока элементов ИЛИ 50 обновленное значение поля МВ проходит через открытый коммутатор 31 на вход *wd* элемента В.МВ. Синхросигнал C_4 , проходящий через элементы И 41 и ИЛИ 32, попадает на вход C_w элемента В.МВ, что обеспечивает запись обновленного значения поля МВ.

В результате описанных выше действий осуществляется вставка элементов дерева C^R в дерево B^R и настройка взаимосвязей между ними.

Временные диаграммы, поясняющие принцип работы схемы в общем случае (обрабатываемые деревья представлены узлами и наборами листьев), приведены на рис. 34.

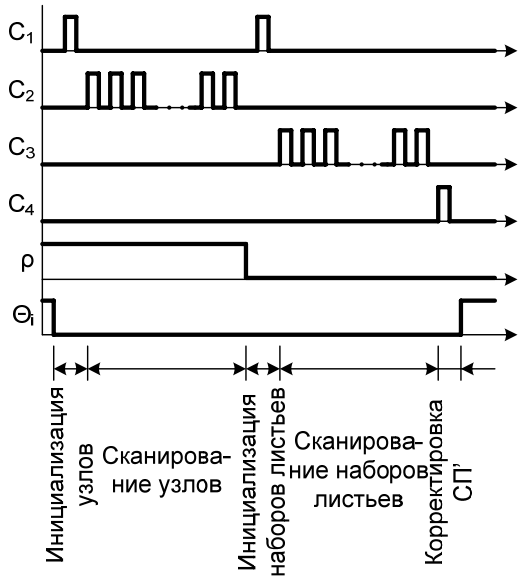


Рис. 34. Временные диаграммы работы схемы вставки поддерева

Затраты на выполнение отдельных элементарных операций рассмотренного выше алгоритма приведены в таблице 2. (Следует отметить, что операция определения младшей свободной позиции выполняется перед копированием каждого набора листьев.)

Таблица 2. Затраты на выполнение элементарных операций алгоритма вставки поддерева

№	Операция	Программная реализация, действий (не более)	Аппаратная реализация, действий (не более)
1	Определение значений вектора $\overline{X}_T$	N_T	1
2	Определения значения N_χ^T	N_T	1
3	Копирование узлов	N_T	N_T
4	Определение значений вектора $\overline{X}_L$	N_L	1
5	Определение младшей свободной позиции в $\overline{X}_L$	N_L	1
6	Копирование наборов листьев	N_L	N_L
7	Корректировка СП'	N_L	1
Итого:		$3N_T + 2N_L + N_L^2$	$N_T + N_L + 4$

Анализ таблицы 2 показывает, что программная реализация предложенного выше алгоритма характеризуется асимптотикой $O\left(N_L^2\right)$, в то время как реализация алгоритма с использованием схемы (рис. 33) характеризуется асимптотикой $O\left(N_R\right)$, т.е. имеет место выигрыш во времени обработки в $\dfrac{3N_T + 2N_L + N_L^2}{N_T + N_L + 4} \approx N_R$ раз. Указанный выигрыш достигается благодаря использованию схем СМНП 1 и 2, СВСН, а также возможности ассоциативного поиска информации в ОСЭМД. Кроме того, приведенные выше цифры не

отражают того, что обработка отдельных полей элементов дерева с использованием схемы (рис. 33) производится параллельно, что еще несколько повышает получаемый выигрыш во времени.

Теоретически возможно построение схемы, в которой копирование узлов будет осуществляться с использованием комбинационной схемы вращения за 1 такт, а наборов листьев – с использованием специализированного коммутатора с N_L входами и N_L выходами. Однако, учитывая достаточно большую аппаратную сложность подобных схем (квадратично зависящую от числа их входов), на данном этапе развития микроэлектроники данное предложение следует считать неоправданным. Кроме того, теоретически возможно совмещение во времени обработки узлов и наборов листьев дерева C^R, однако это потребует дублирования сдвиговых регистров и введения дешифраторов, что также приведет к росту аппаратной сложности схемы.

12. Схемотехническая реализация операции раскрытия скобок

Реализация операции подстановки, подробно рассмотренная в предыдущем разделе, в некоторых случаях может приводить к нарушению требования корректности дерева № 6, что может повлечь за собой невозможность редукции системы *R*-выражений некоторых алгоритмов управления или ошибочное функционирование некоторых блоков разрабатываемого акселератора (например, схемы определения отношения нестрогого включения). Для приведения полученного в ходе операции подстановки дерева в соответствие с требованием корректности № 6 необходимо выполнение операции раскрытия скобок.

Исходя из сформулированной в [10] особенности операции раскрытия скобок можно заметить, что для приведения полученного на предыдущем шаге *R*-выражения в соответствие с требованием корректности № 6, необходима проверка лишь одной пары узлов (а не всех возможных пар узлов предок-потомок, как это было предложено в [3]), что приблизительно в $N_T(B)$ раз сокращает время выполнения операции и упрощает структуру схемы для ее выполнения.

Схема, реализующая действия описанного выше алгоритма, приведена на рис. 35.

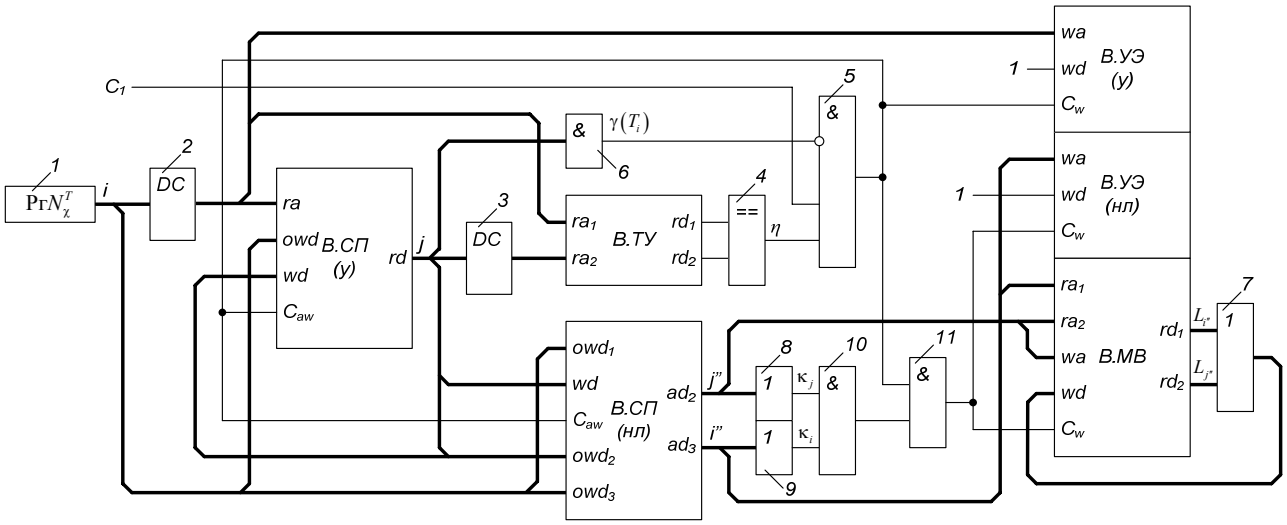


Рис. 35. Схема операции раскрытия скобок (СРС)

Регистр 1 используется для хранения значения N_χ^T, дешифратор 2 преобразует значение N_χ^T из двоичного кода в унитарный, дешифратор 3 преобразует значение номера узла-предка

T_j^B из двоичного кода в унитарный, элемент сравнения 4 используется для формирования признака η совпадения типа рассматриваемой пары узлов, элемент запрета 5 управляет прохождением синхросигнала C_1 на входы элементов В.УЭ(у), В.УЭ(нл) и В.МВ, элемент И 6 формирует значение признака $\gamma(T_i)$ наличия предка у рассматриваемого узла, блок элементов ИЛИ 7 реализует операцию объединения наборов листьев $L_{i''}$ и $L_{j''}$, элементы ИЛИ 8 и 9 формируют значения признаков κ_i и κ_j наличия наборов листьев у узлов T_i и T_j, элементы И 10 и 11 управляют прохождением синхросигнала C_1 на вход элемента В.МВ.

Схема работает следующим образом. Значение номера узла $i = N_\chi^T$ с выхода регистра $\text{Рг}N_\chi^T$ 1 в двоичном коде поступает на вход дешифратора 2. Значение с выхода дешифратора 2 в унитарном коде поступает на вход *ra* элемента В.СП(у), на выходе *rd* которого формируется значение *j* в двоичном коде. Сформированное значение поступает на вход дешифратора 3, на выходе которого формируется значение *j* в унитарном коде. Значения номеров рассматриваемой пары узлов *i* и *j* с выходов дешифраторов 2 и 3 подаются на входы ra_1 и ra_2 элемента В.ТУ, на выходах rd_1 и rd_2 которого формируются значения полей ТС рассматриваемых узлов. С выходов элемента В.ТУ сформированные значения поступают на входы элемента сравнения 4, на выходе которого формируется двоичное значение признака η совпадения типа рассматриваемых узлов, поступающее на вход элемента запрета 5. Значение с выхода *rd* элемента В.СП(у) также поступает на вход элемента И 6, на выходе которого формируется двоичное значение признака γ отсутствия предка у *i*-го узла. Сформированное значение признака γ поступает на вход элемента запрета 5. Значения *i* и *j* с выхода регистра $\text{Рг}N_\chi^T$ 1 и с выхода *rd* элемента В.СП(у) поступают соответственно на входы owd_3 и owd_2 элемента В.СП(нл), на выходах ad_3 и ad_2 которого формируются номера дочерних наборов листьев i'' и j'' соответственно. Значение с выхода ad_2 элемента В.СП(нл) поступает на выходы ra_2 и *wa* элемента В.МВ, а значение с выхода ad_3 элемента В.СП(нл) – на входы *wa* элемента В.УЭ(нл) и ra_1 элемента В.МВ. На вход *wd* элемента В.УЭ(нл) поступает сигнал логической единицы: элемент подготовлен к пометке i''-го набора листьев как удаленного. На выходах rd_1 и rd_2 элемента В.МВ формируются значения полей МВ i''-го и j''-го наборов листьев, поступающие на входы блока элементов ИЛИ 7. На выходе блока элементов ИЛИ 7 формируется значение объединенного набора листьев, поступающее на вход *wd* элемента В.МВ. Значения с выходов ad_2 и ad_3 элемента В.СП(нл) поступают на входы элементов ИЛИ 8 и 9 соответственно и формируют на их выходах значения двоичных признаков κ_i и κ_j присутствия наборов листьев у узлов с номерами *i* и *j*. Сформированные значения поступают на входы элемента И 10 и образуют единичное значение, открывающее элемент И 11 для прохождения синхросигнала C_1, на его выходе только в случае наличия наборов листьев у обоих узлов. На вход *wa* элемента В.УЭ(у) с выхода дешифратора 2 поступает значение *i* в унитарном коде, при этом на входе *wd* присутствует сигнал логической единицы: элемент В.УЭ(у) подготовлен к пометке *i*-го узла дерева как удаленного. На входы *owd* элемента В.СП(у) и owd_1 элемента В.СП(нл) с выхода регистра $\text{Рг}N_\chi^T$ подается значение N_χ^T. При этом на входах *wd* элементов В.СП(у) и В.СП(нл) присутствует значение *j* с выхода *rd* элемента В.СП(у).

С появлением единичного значения синхросигнала C_1 начинается модификация полей элементов дерева, участвующих в обработке. Если имеет место совпадение типов узлов T_i и T_j, на выходе элемента сравнения 4 сформировано единичное значение признака η, при

этом факт присутствия узла T_j в дереве определяется нулевым значением признака γ на выходе элемента И 6. В случае соблюдения условий преобразования $\left((\eta=1)\wedge(\gamma=0)\right)$ синхросигнал C_1 проходит через элемент запрета 5 и появляется на его выходе. С выхода элемента запрета 5 синхроимпульс C_1 попадает на вход C_{aw} элемента В.СП(у), обеспечивая настройку полей СП потомков узла T_i на узел T_j , на вход C_{aw} элемента В.СП(нл), обеспечивая настройку поля СП дочернего набора листьев узла T_i на узел T_j, и на вход C_w элемента В.УЭ(у), обеспечивая пометку узла T_i как удаленного. Если у обоих узлов T_i и T_j имеются дочерние наборы листьев, на выходе элемента И 10 сформировано единичное значение, открывающее элемент И 11 для прохождения синхросигнала C_1 на входы C_w элементов В.МВ и В.УЭ(нл), что обеспечивает пометку i''-го набора листьев как удаленного и добавление содержимого его поля МВ к содержимому поля МВ j''-го набора листьев.

Таким образом, в результате описанных выше действий осуществляется раскрытие скобок в полученном после подстановки дереве B^R , после чего оно удовлетворяет требованию корректности № 6.

Временная диаграмма работа устройства ввиду ее простоты не приводится: модификация полей элементов дерева B^R осуществляется в результате появления единичного синхроимпульса C_1.

Затраты времени на выполнение элементарных операций приведенного выше алгоритма при его программной и аппаратной реализациях приведены в таблице 3.

Таблица 3. Затраты на выполнение элементарных операций алгоритма раскрытия скобок

№	Операция	Программная реализация, действий (не более)	Аппаратная реализация, действий (не более)
1	Нахождение значений полей узлов T_i и T_j	1	1
2	Корректировка полей СП потомков узла T_i	$N_L + N_T = N_R$	1
3	Объединение дочерних наборов листьев $L_{i''}$ и $L_{j''}$	$L_{\max}$	1
Итого:		$1+N_R+L_{\max}$	1

Приведенные в таблице данные показывают, что основной выигрыш в скорости обработки при использовании разработанного аппаратного решения по сравнению с программной реализацией достигается за счет возможности ассоциативного поиска информации о потомках указанного узла и за счет параллельной обработки элементов векторов МВ. Средний выигрыш во времени обработки составляет величину порядка $O\left(N_R+L_{\max}\right)$. Также необходимо отметить, что параллельная работа элементов, хранящих различные поля дерева, и использование нескольких портов для чтения и ассоциативного поиска данных также несколько повышают общий выигрыш в скорости обработки.

13. Схемотехническая реализация операции удаления пустых позиций

В результате выполнения описанных выше действий реализуется операция подстановки, в ходе которой из деревьев A^R, B^R и C^R формируется обновленное дерево B^R путем удаления из него поддерева, *r*-изоморфного дереву A^R, с последующей вставкой дерева C^R в качестве поддерева на место удаленного. При этом полученное в результате ряда рассмотренных выше операций дерево B^R отвечает всем требованиям корректности, за исключением требования № 3 (в некоторых случаях). Несоответствие указанному требованию заключается в наличии «пропусков», образованных в результате удаления части элементов дерева, и в некорректных значениях полей ТКУ и ТКЛ (поля ТКУ и ТКЛ дерева B^R, изначально корректно инициализированные, были инкрементированы в ходе операции вставки поддерева на число добавленных узлов и наборов листьев соответственно, однако не были декрементированы на число удаленных элементов).

В ходе операции удаления пустых позиций осуществляется приведение полученного дерева в соответствие с требованием корректности № 3 путем ликвидации «пропусков» и последующей корректировки значений полей ТКУ и ТКЛ. Описание алгоритма преобразования приведено в [10]. В результате выполнения действий описанного выше алгоритма полученное в результате преобразований дерево B^R удовлетворяет всем требованиям корректности и может быть корректно обработано в ходе дальнейших действий.

Схемотехническая реализация операции удаления пустых позиций приведена на рис. 36.

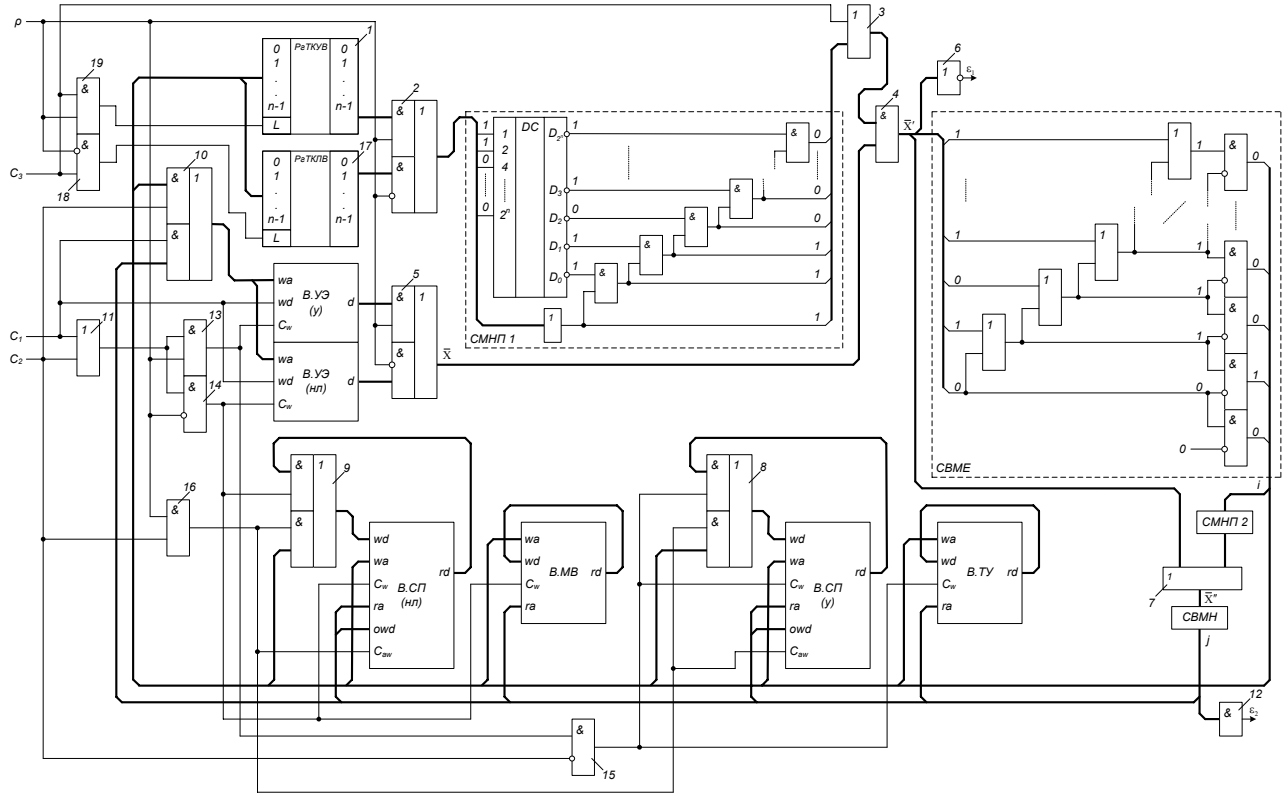


Рис. 36. Схема операции удаления пустых позиций (СУПП)

Регистры 1 и 17 хранят значения $N_T(B)$ и $N_L(B)$ соответственно, коммутаторы 2 и 5 используются для коммутации входных значений в зависимости от этапа алгоритма (обработка узлов или наборов листьев), блок элементов ИЛИ 3 используется на

завершающем этапе работы алгоритма при корректировке значений $N_T(B)$ и $N_L(B)$, блок элементов И 4 используется для формирования вектора $\overline{\mathrm{X}}'$, элемент ИЛИ-НЕ 6 формирует на выходе значение признака ε_1 завершения обработки, блок элементов ИЛИ 7 используется для формирования значений вектора $\overline{\mathrm{X}}''$, коммутаторы 8–10 используются для коммутации входных значений при копировании значений полей по синхросигналу C_1 и последующей ассоциативной замене значений полей потомков по синхросигналу C_2, элементы ИЛИ 11, И 13, запрета 14 и 15, И 16 используются для управления прохождением синхросигналов на входы коммутаторов 8–10 и записью обновленных значений, элемент И 12 используется для формирования признака ε_2 завершения обработки, элементы запрета 18 и И 19 управляют прохождением синхросигнала C_3 при корректировке значений $N_T(B)$ и $N_L(B)$.

Схема работает следующим образом. Перед началом обработки синхросигнал ρ устанавливается в единичное значение: производится подготовка к обработке узлов дерева. Значение $N_T''(B)$ с выхода регистра РгТКУВ 1 проходит через коммутатор 2, открытый единичным значением сигнала ρ, на вход схемы маскировки неиспользуемых позиций СМНП 1. На выходе схемы СМНП 1 формируется двоичный вектор вида $\underbrace{11...1}_{N_T''(B)}00...0$, который проходит через блок элементов ИЛИ 3 (синхросигнал $C_3 = 0$ в данный момент времени) на первый вход блока элементов И 4. На второй вход блока элементов И 4 с выхода *d* элемента В.УЭ(у) поступает значение $\overline{\mathrm{X}}_T$, проходящее через коммутатор, открытый единичным значением сигнала ρ. На выходе блока элементов И 4 формируется значение вектора $\overline{\mathrm{X}}_T'$, поступающее на вход элемента ИЛИ-НЕ 6, на вход схемы выделения младшей единицы СВМН и на первый вход блока элементов ИЛИ 7. На выходе элемента ИЛИ-НЕ 6 формируется значение признака ε_1. Если $\varepsilon_1 = 1$, обработка узлов дерева прекращается и производится переход к обработке наборов листьев. На выходе схемы СВМЕ формируется значение *i* номера младшей свободной позиции, поступающее на вход схемы СМНП 2, а также на входы *wa* элементов В.ТУ, В.СП(у), В.МВ и В.СП(нл), на первые входы коммутаторов 8, 9 и 10. На выходе схемы СМНП 2 формируется двоичный вектор вида $\underbrace{11...1}_{i}00...0$, поступающий на второй вход блока элементов ИЛИ 7. На выходе блока элементов ИЛИ 7 формируется вектор $\overline{\mathrm{X}}_T''$, поступающий на вход схемы выделения младшего нуля СВМН. На выходе схемы СВМН формируется значение *j* младшей занятой позиции правее свободной, поступающее на вход элемента И 12, а также на входы *ra* элементов В.ТУ и В.МВ, на входы *ra* и *owd* элементов В.СП(у) и В.СП(нл) и на второй вход коммутатора 10. На выходе элемента И 12 формируется значение признака ε_2. Если $\varepsilon_2 = 1$, обработка узлов дерева прекращается и производится переход к обработке наборов листьев.

Значения, поданные на входы *ra* элементов В.ТУ, В.СП(у), В.МВ и В.СП(нл) формируют на их выходах *rd* значения соответствующих полей элементов дерева. Значения с выходов *rd* элементов В.ТУ и В.МВ поступают на входы *wd* элементов В.ТУ и В.МВ соответственно. Значение с выходов *rd* элементов В.СП(у) и В.СП(нл) поступают на вторые входы коммутаторов 8 и 9 соответственно.

Далее синхросигнал C_1 устанавливается в единичное значение. Единичное значение синхросигнала C_1 поступает на входы *wd* элементов В.УЭ(у) и В.УЭ(нл). Также синхросигнал C_1 открывает коммутатор 10 для прохождения значения *j* с выхода схемы СВМН на входы *wa* элементов В.УЭ(у) и В.УЭ(нл). Также синхросигнал проходит через элемент ИЛИ 11 на входы элемента И 13 и элемента запрета 14. Элемент запрета 14 закрыт

для прохождения синхросигнала C_1 единичным значением сигнала ρ. Синхросигнал C_1 проходит через элемент И 13, открытый единичным значением сигнала ρ, и попадает на вход C_w элемента В.УЭ(у), а также на входы C_w элементов В.ТУ и В.СП(у) через элемент запрета 15, открытый нулевым значением сигнала C_2. Кроме того, синхросигнал C_1 также открывает коммутатор 8 для прохождения значения с выхода *rd* элемента В.СП(у) на его вход *wd*. Таким образом, производится пометка *j*-го узла как удаленного и копирование его полей ТУ и СП из *j*-ой позиции табличного представления в *i*-ю.

Далее синхросигнал C_1 принимает нулевое значение, а синхросигнал C_2 устанавливается в единичное значение. Появление синхросигнала C_2 открывает коммутатор 10 для прохождения значения *i* с выхода схемы СВМЕ на входы *wa* элементов В.УЭ(у) и В.УЭ(нл). Также синхросигнал C_2 проходит через элемент ИЛИ 11 и попадает на входы элемента И 13 и элемента запрета 14. Аналогично рассмотренному выше синхросигнал не проходит через элемент запрета 14 и проходит через элемент И 13 на вход C_w элемента В.УЭ(у): благодаря нулевому значению сигнала C_1, присутствующему на входах *wd* элементов В.УЭ(у) и В.УЭ(нл), обеспечивается пометка *i*-го узла как занятого. При этом синхросигнал C_2 не появляется на выходе элемента запрета 15: поля СП и ТУ узлов дерева остаются без изменений. Также синхросигнал C_2 проходит через элемент И 16 и открывает коммутаторы 8 и 9 для поступления значения *i* с выхода схемы СВМЕ на входы *wd* элементов В.СП(у) и В.СП(нл) соответственно. Далее синхросигнал C_2, попадающий на входы C_{aw} элементов В.СП(нл) и В.СП(у), обеспечивает замену значения *j* на *i* для полей СП всех элементов дерева.

Далее на выходе элемента В.УЭ(у) появляется измененный вектор $\overline{\mathrm{X}}_T$, что приводит к формированию новых значений векторов $\overline{\mathrm{X}}'_T$ и $\overline{\mathrm{X}}''_T$, получению новой пары значений *i* и *j* и новой итерации работы схемы аналогично рассмотренному выше. Завершение обработки узлов происходит при появлении единичного значения признака ε_1 или ε_2.

При переходе к обработке наборов листьев сигнал ρ устанавливается в нулевое значение. Это приводит к тому, что на вход схемы СМНП 1 с выхода коммутатора 2 поступает значение $N''_L(B)$ с выхода регистра РгТКЛВ 17, а на второй вход блока элементов И 4 с выхода коммутатора 5 – вектор $\overline{\mathrm{X}}_L$. Аналогично рассмотренному выше осуществляется формирование векторов $\overline{\mathrm{X}}'_L$ (на выходе блока элементов ИЛИ 4) и $\overline{\mathrm{X}}''_L$ (на выходе блока элементов ИЛИ 7), признаков ε_1 и ε_2 и значений *i* и *j*, поступающих на входы элементов В.СП(нл), В.МВ, В.СП(у), В.ТУ и коммутаторов 8, 9 и 10.

Далее синхросигнал C_1 устанавливается в единичное значение. Единичное значение синхросигнала C_1 поступает на входы *wd* элементов В.УЭ(у) и В.УЭ(нл). Также синхросигнал C_1 открывает коммутатор 10 для прохождения значения *j* с выхода схемы СВМН на входы *wa* элементов В.УЭ(у) и В.УЭ(нл). Также синхросигнал проходит через элемент ИЛИ 11 и поступает на входы элемента И 13 и элемента запрета 14. Элемент И 13 закрыт для прохождения синхросигнала C_1 нулевым значением сигнала ρ, элемент запрета 14 – открыт. С выхода элемента запрета 14 синхросигнал C_1 поступает на вход коммутатора 9 и открывает его для прохождения значения с выхода *rd* элемента В.СП(нл) на его вход *wd*. Появление синхросигнала C_1 на входах C_w элементов В.УЭ(нл), В.СП(нл) и В.МВ обеспечивает пометку *j*-го набора листьев как удаленного и копирование значений полей СП и МВ наборов листьев из *j*-ой позиции табличного представления в *i*-ю.

Далее синхросигнал C_1 принимает нулевое значение, а синхросигнал C_2 устанавливается в единицу. Единичное значение синхросигнала C_2 открывает коммутатор 10 для прохождения значения i с выхода схемы СВМЕ на входы wa элементов В.УЭ(у) и В.УЭ(нл), при этом на их входах wd присутствует нулевое значение синхросигнала C_1. Также синхросигнал C_2 проходит через элемент ИЛИ 11 и поступает на входы элемента И 13 и элемента запрета 14. Аналогично рассмотренному выше, синхросигнал C_2 не проходит на выход элемента И 13 и проходит на выход элемента запрета 14 (элемент И 16 также закрыт для прохождения синхросигнала C_2 нулевым значением сигнала ρ). С выхода элемента запрета синхросигнал поступает на вход C_w элемента В.УЭ(нл): i-я позиция в табличном представлении наборов листьев помечается как занятая. Также синхросигнал с выхода элемента запрета 14 попадает на входы C_w элементов В.СП(нл) и В.МВ, не изменяя значения полей СП и МВ наборов листьев (фактически копирование значений полей производится повторно).

Далее на выходе элемента В.УЭ(нл) появляется измененный вектор $\overline{\mathrm{X}}_L$, что приводит к формированию новых значений векторов $\overline{\mathrm{X}}'_L$ и $\overline{\mathrm{X}}''_L$, получению новой пары значений i и j и новой итерации работы схемы аналогично рассмотренному выше. Завершение обработки наборов листьев происходит при появлении единичного значения признака ε_1 или ε_2.

В результате описанных выше действий осуществляется ликвидация всех пропусков в составе табличного представления узлов и наборов листьев, а на выходах элементов В.УЭ(у) и В.УЭ(нл) присутствуют вектора $\overline{\mathrm{X}}_T = \underbrace{00...0}_{N_T(B)}11...1$ и $\overline{\mathrm{X}}_L = \underbrace{00...0}_{N_L(B)}11...1$ соответственно. Далее сигнал ρ устанавливается равным единице, что приводит к появлению на выходе коммутатора 5 значения $\overline{\mathrm{X}}_T$, поступающего на второй вход блока элементов И 4. Появление синхросигнала C_3 формирует на выходе блока элементов ИЛИ 3 вектор из единичных значений, поступающей на первый вход блока элементов И 4. Благодаря этому на выходе блока элементов И 4 оказывается сформирован вектор $\overline{\mathrm{X}}_T$, поступающий на вход схемы СВМЕ. На выходе схемы СВМЕ формируется значение номера младшей свободной позиции, поступающее на входы регистров РгТКУВ 1 и РгТКЛВ 17. Также синхросигнал C_3, не проходя через элемент запрета 18 и проходя через элемент И 19 благодаря единичному значению сигнала ρ, поступает на синхровход регистра РгТКУВ 1, что обеспечивает запись значения обновленного значения $N_T(B)$. Аналогично осуществляется корректировка значения $N_L(B)$: сигнал ρ переключается в ноль, на выходе коммутатора 5 и, следовательно, блока элементов И 4 формируется значение вектора $\overline{\mathrm{X}}_L$, а на выходе схемы СВМЕ – $N_L(B)$. Синхроимпульс C_3 не проходит через элемент И 19 и проходит через элемент запрета 18 на синхровход регистра РгТКЛВ 17, обеспечивая запись значения $N_L(B)$ с выхода схемы СВМЕ.

Таким образом, в результате описанных выше действий элементы В.СП(нл), В.МВ, В.СП(у), В.ТУ и регистры РгТКУВ 1 и РгТКЛВ 17 содержат табличное представление дерева B^R, модифицированного в ходе операции подстановки, отвечающего всем требованиям корректности и готового к дальнейшим преобразованиям.

Временные диаграммы, поясняющие принцип работы схемы, приведены на рис. 37.

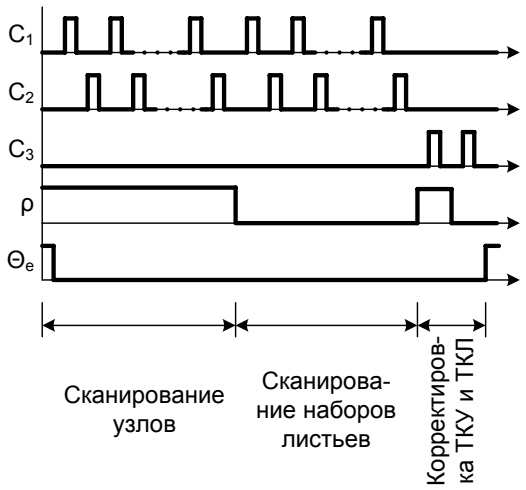


Рис. 37. Временная диаграмма работы схемы удаления пустых позиций

Временные затраты на реализацию элементарных операций приведены в таблице 4.

Таблица 4. Затраты на выполнение элементарных действий операции удаления пустых позиций

№	Операция	Программная реализация, действий (не более)	Аппаратная реализация, действий (не более)
1	Поиск младшей свободной позиции (узлы)	N_T	1
2	Поиск занятой позиции правее свободной (узлы)	N_T	1
3	Копирование узла из занятой позиции в свободную	≈ 1	1
4	Корректировка значений полей СП	$N_T + N_L = N_R$	1
5	Поиск младшей свободной позиции (наборы листьев)	N_L	1
6	Поиск занятой позиции правее свободной (наборы листьев)	N_L	1
7	Копирование набора листьев из занятой позиции в свободную	$L_{\max}$	1
Итого:		$\left(N_T + N_T + 1 + N_R\right) N_T + \\ + \left(N_L + N_L + L_{\max}\right) N_L = \\ = L_{\max} N_L + 3N_T^2 + 2N_L^2 + N_L N_T + N_T$	$\left(1+1+1+1\right) N_T + \\ + \left(1+1+1\right) N_L = \\ = 4N_T + 3N_L$

Предложенное схемотехническое решение поставленной задачи имеет выигрыш в скорости обработки по сравнению с программной реализацией за счет быстрого выделения номеров *i* и *j* позиций (теоретически не зависящего от числа позиций в обрабатываемых векторах), с также за счет ассоциативной корректировки значений полей СП и быстрого параллельного копирования всех бит полей элементов дерева. С учетом того, что $N_T \approx N_L$, выигрыш в скорости обработки от использования схемотехнического решения по сравнению с программной реализацией составляет

$$\frac{L_{\max}N_L + 3N_T^2 + 2N_L^2 + N_L N_T + N_T}{4N_T + 3N_L} \approx \frac{L_{\max}N_R + 6N_R^2 + N_R}{7N_R} =$$

$$= \frac{1}{7}L_{\max} + \frac{6}{7}N_R + \frac{1}{7} \simeq O\left(L_{\max} + N_R\right).$$

В полученной формуле не учитывается выигрыш, получаемый за счет параллельной работы элементов ОСЭМД, хранящих различные поля дерева, поэтому реальный выигрыш в скорости обработки должен быть несколько выше.

Теоретически возможна параллельная обработка узлов и наборов листьев, однако это потребует дублирования схем выделения граничных значений и введения дополнительных элементов для синхронизации процедур обработки элементов дерева при выполнении операции ассоциативной корректировки значений полей СП элементов дерева, что на данном этапе развития микроэлектроники считается неоправданным. Также теоретически возможно параллельное копирование занятых позиций в свободные с использованием сложного коммутатора с достаточно большой аппаратной сложностью, синтез которого является нетривиальной задачей.

14. Оценка аппаратной сложности акселератора

С целью абстрагироваться от деталей схемотехнической реализации ПЛИС различных семейств или же тонкостей заказного исполнения СБИС акселератора в соответствии с каким-либо техпроцессом оценку аппаратной сложности разработанного акселератора выполним в эквивалентных вентилях (ЭВ), при этом под ЭВ будем подразумевать одно- или двухвходовой логический элемент, реализующий какую-либо булеву функцию (НЕ, И, ИЛИ, исключающее ИЛИ, эквивалентность, импликацию и т.д.). Сперва выполним оценку аппаратной сложности схем, входящих в состав акселератора, а затем – сложности акселератора в целом. (Оценку аппаратной сложности многопортовой ассоциативной памяти акселератора проведем отдельно, не включая ее в состав аппаратных сложностей схем отдельных преобразований.)

Схема операции определения принадлежности вершины дереву по сути представляет собой мультиплексор $L_{\max} \to 1$, поэтому ее сложность определяется сложностью мультиплексора:

$$R_1 = 13\left(L_{\max} - 1\right) = 13L_{\max} - 13 \;\; ЭВ.$$

Аппаратная сложность схемы операции определения ω-мощности складывается из аппаратной сложности входящих в ее состав схем. С учетом, что сложность дешифратора с N выходами составляет $3\left(N-1\right)ЭВ$, а сдвигового регистра на N разрядов – $12N\,ЭВ$, сложность схемы определяется как

$$R_2 = 3\left\lceil \log_2 N_{T_{\max}} \right\rceil + 3\left(N_{T_{\max}} - 1\right) + 12N_{T_{\max}} + 3\left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+ 3\left(N_{T_{\max}} - 1\right) + \left\lceil \log_2 N_{T_{\max}} \right\rceil + 1 + 1 + 3\left\lceil \log_2 N_{T_{\max}} \right\rceil + 1 + N_{T_{\max}} +$$
$$+ \left\lceil \log_2 N_{T_{\max}} \right\rceil + 1 + \left\lceil \log_2 N_{T_{\max}} \right\rceil - 1 + \left\lceil \log_2 N_{T_{\max}} \right\rceil + 1 + 1 =$$
$$= 13\left\lceil \log_2 N_{T_{\max}} \right\rceil + 19N_{T_{\max}} - 1 \;\; ЭВ.$$

Схема операции проверки *r*-изоморфизма деревьев включает в своем составе шифратор, сложность которого составляет $3\left(N-1\right) - \left\lceil \log_2 N \right\rceil + 1 \;\; ЭВ$ для шифратора с N входами. Ее аппаратная сложность составляет

$$R_3 = 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left(N_{T_{\max}} - 1\right) + 12N_{T_{\max}} + 1 + N_{T_{\max}} + N_{T_{\max}} + N_{T_{\max}} +$$
$$+ N_{T_{\max}} + N_{T_{\max}} + N_{T_{\max}} + 1 + L_{\max} +$$
$$+ \left(L_{\max} + L_{\max} + L_{\max} + L_{\max} - 1 + L_{\max} + L_{\max} - 1\right) N_{L_{\max}} +$$
$$+ 3\left(N_{T_{\max}} - 1\right) - \left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 1 + 3\left(N_{T_{\max}} - 1\right) + 3N_{T_{\max}} + 3N_{T_{\max}} +$$
$$+ 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + N_{T_{\max}} + 3\left(N_{T_{\max}} - 1\right) + N_{T_{\max}} - 1 + 1 + N_{T_{\max}} + 1 + 1 +$$
$$+ 3\left(N_{T_{\max}} - 1\right) + 3\left(N_{T_{\max}} - 1\right) + \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + 1 + N_{T_{\max}} + 3\left(N_{T_{\max}} - 1\right) +$$
$$+ \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + 2N_{T_{\max}} + 3N_{T_{\max}} =$$
$$= 7\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 53N_{T_{\max}} - 17 + L_{\max} + 6L_{\max} N_{L_{\max}} \ \text{ЭВ}.$$

Схема удаления поддерева характеризуется сложностью

$$R_4 = 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left(N_{T_{\max}} - 1\right) + 12N_{T_{\max}} + 2 + N_{T_{\max}} + 1 +$$
$$+ 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left(N_{T_{\max}} - 1\right) + 1 + 1 + 2 + L_{\max} + 1 + 2 + N_{T_{\max}} + 1 =$$
$$= 6\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 20N_{T_{\max}} + 5 + L_{\max} \ \text{ЭВ}.$$

В состав схемы вставки входят регистры, сложность которых характеризуется величиной $4N$ (при условии хранения N бит). Ее аппаратная сложность составляет

$$R_5 = 3N_{T_{\max}} + 1 + 1 + 3\left(N_{T_{\max}} - 1\right) - \left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 1 + 4\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 2 +$$
$$+ \left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 2\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left(N_{T_{\max}} - 1\right) + 12N_{T_{\max}} + 1 + 3N_{T_{\max}} + 1 +$$
$$+ 1 + 3\left(N_{T_{\max}} - 1\right) - \left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 1 + 3\left(N_{T_{\max}} - 1\right) + 3N_{T_{\max}} +$$
$$+ 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 1 + 1 + 2 + 3\left(N_{T_{\max}} - 1\right) + N_{T_{\max}} +$$
$$+ 3L_{\max} + 1 + 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left(N_{T_{\max}} - 1\right) + 3\left(N_{T_{\max}} - 1\right) + 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor +$$
$$+ \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + 1 + 1 + 1 + N_{T_{\max}} - 1 + 1 +$$
$$+ 1 + 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3N_{T_{\max}} + 4N_{T_{\max}} + 1 + L_{\max} =$$
$$= 51N_{T_{\max}} - 5 + 20\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 4L_{\max} \ \text{ЭВ}.$$

Схема раскрытия скобок характеризуется аппаратной сложностью

$$R_6 = 3\left(N_{T_{\max}} - 1\right) + 3\left(N_{T_{\max}} - 1\right) + 1 + 2 + \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 +$$
$$+ L_{\max} + \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + \left\lfloor \log_2 N_{T_{\max}} \right\rfloor - 1 + 1 + 1 =$$
$$= 6N_{T_{\max}} - 4 + 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + L_{\max} \ \text{ЭВ}.$$

Схема удаления пропусков характеризуется аппаратной сложностью

$$R_7 = 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + N_{T_{\max}} + N_{T_{\max}} + 3N_{T_{\max}} + N_{T_{\max}} + N_{T_{\max}} +$$
$$+ 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 3N_{T_{\max}} + 1 + N_{T_{\max}} - 1 + 1 +$$
$$+ 1 + 1 + 1 + 1 + 1 =$$
$$= 9\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + 11N_{T_{\max}} + 6 \ \text{ЭВ}.$$

Зная аппаратную сложность схем операций преобразования R-выражений, оценим сложность комбинационной части акселератора:

$$R_{комб} = 4R_1 + 4R_2 + 2R_3 + 2\cdot 2\left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+4\cdot 3\left(N_{T_{\max}} + 2N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil + L_{\max} N_{T_{\max}} + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) +$$
$$+R_4 + R_5 + R_6 + R_7 =$$
$$= 8\left\lceil \log_2 L_{\max} \right\rceil + 24N_{T_{\max}} L_{\max} + 60L_{\max} - 88 + 132\left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+282N_{T_{\max}} + 24N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil \ ЭВ.$$

(Полученная формула не включает сложность устройства управления и контроллера шины PCI Express, т.к. в зависимости от реализации они могут меняться в достаточно широких пределах.)

Аппаратная сложность памяти акселератора главным образом определяется аппаратными затратами на реализацию однородной среды электронной модели дерева. Учитывая, что для различных полей состав необходимых операций различен (например, возможность ассоциативного поиска используется только для полей СП), сложность ячейки среды общего вида составляет

$$R_{яч} = 4 + 2K_1 + 4K_2 + 3\alpha,$$

где $K_1 \geq 1$ – число портов чтения, $K_2 \geq 0$ – число портов для ассоциативного поиска, $\alpha \in \{0,1\}$ – признак необходимости ассоциативной записи. В случае каждого из полей она индивидуальна, что в конечном счете позволяет снизить аппаратную сложность памяти в целом.

Однородная среда, образованная матрицей из $N \times M$ ячеек, характеризуется аппаратной сложностью

$$R_{ср} = R_{яч} \cdot N \cdot M + N,$$

причем слагаемое N отвечает за блок инверторов на входе *owd*.

Формулы для расчета аппаратной сложности ячеек и элементов ОСЭМД, используемых для хранения различных полей, приведены в таблице 5.

Таблица 5. Аппаратная сложность и функциональные характеристики ячеек среды электронной модели дерева

Поле	**Размер, бит**	**Число портов чтения**	**Число портов ассоциативного поиска**	**Возможность ассоциативной записи**	**Сложность ячейки, ЭВ**	**Сложность среды, ЭВ**
ТУ	1	2	–	–	8	$8N_{T_{\max}} + 1$
СП(у)	$\log_2 N_{T_{\max}}$	1	1	+	13	$13N_{T_{\max}} \log_2 N_{T_{\max}} + \log_2 N_{T_{\max}}$
МУ	$\log_2 L_{\max}$	2	–	–	8	$8N_{T_{\max}} \log_2 L_{\max} + \log_2 L_{\max}$
ТС(у)	2	2	–	–	8	$16N_{T_{\max}} + 2$

НС(у)	$\log_2 N_{T_{\max}}$	1	–	–	6	$6N_{T_{\max}} \log_2 N_{T_{\max}} + +\log_2 N_{T_{\max}}$
УЭ(у)	1	1	–	–	6	$6N_{T_{\max}} + 1$
МВ	$L_{\max}$	2	–	–	8	$8N_{T_{\max}} L_{\max} + L_{\max}$
СП(нл)	$\log_2 N_{T_{\max}}$	1	3	+	21	$21N_{T_{\max}} \log_2 N_{T_{\max}} + +\log_2 N_{T_{\max}}$
ТС(нл)	2	1	–	–	6	$12N_{T_{\max}} + 2$
НС(нл)	$\log_2 N_{T_{\max}}$	1	–	–	6	$6N_{T_{\max}} \log_2 N_{T_{\max}} + +\log_2 N_{T_{\max}}$
УЭ(нл)	1	1	–	–	6	$6N_{T_{\max}} + 1$
ТКУ	$\log_2 N_{T_{\max}}$	–	–	–	–	$4\log_2 N_{T_{\max}}$
ТКЛ	$\log_2 N_{T_{\max}}$	–	–	–	–	$4\log_2 N_{T_{\max}}$

Таким образом, аппаратная сложность ОСЭМД, требуемой для хранения одного *R*-выражения, составляет величину

$$R_R = 48N_{T_{\max}} + 7 + 54N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil + +5\left\lceil \log_2 N_{T_{\max}} \right\rceil + 8N_{T_{\max}} L_{\max} + L_{\max} \;\; ЭВ.$$

Во время обработки пары выражений системы Ξ акселератор оперирует четырьмя *R*-выражениями, что с учетом сложности регистров РгННВ и РгНКВ, определяет аппаратную сложность памяти акселератора:

$$R_{пам} = 4R_R + 2 \cdot 4\left\lceil \log_2 L_{\max} \right\rceil =$$
$$= 4\begin{pmatrix} 48N_{T_{\max}} + 7 + 54N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil + \\ +5\left\lceil \log_2 N_{T_{\max}} \right\rceil + 8N_{T_{\max}} L_{\max} + L_{\max} \end{pmatrix} + 8\left\lceil \log_2 L_{\max} \right\rceil =$$
$$= 192N_{T_{\max}} + 28 + 216N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil + 20\left\lceil \log_2 N_{T_{\max}} \right\rceil + 32N_{T_{\max}} L_{\max} +$$
$$+4L_{\max} + 8\left\lceil \log_2 N_{T_{\max}} \right\rceil \;\; ЭВ.$$

Зная сложность комбинационной части акселератора и его памяти, можно оценить его интегральную аппаратную сложность:

$$R = R_{комб} + R_{пам} =$$
$$= 8\left\lceil \log_2 L_{\max} \right\rceil + 24N_{T_{\max}} L_{\max} + 60L_{\max} - 88 + 132\left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+282N_{T_{\max}} + 24N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+192N_{T_{\max}} + 28 + 216N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil + 20\left\lceil \log_2 N_{T_{\max}} \right\rceil + 32N_{T_{\max}} L_{\max} +$$
$$+4L_{\max} + 8\left\lceil \log_2 N_{T_{\max}} \right\rceil =$$
$$= 16\left\lceil \log_2 L_{\max} \right\rceil + 56N_{T_{\max}} L_{\max} + 64L_{\max} - 60 + 152\left\lceil \log_2 N_{T_{\max}} \right\rceil +$$
$$+474N_{T_{\max}} + 250N_{T_{\max}} \left\lceil \log_2 N_{T_{\max}} \right\rceil \;\; ЭВ.$$

С использованием полученной формулы можно получить оценки аппаратной сложности ускорителя для случаев различного числа вершин в составе граф-схем обрабатываемых алгоритмов управления (таблица 6).

Таблица 6. Зависимость аппаратной сложности акселератора от числа вершин обрабатываемых алгоритмов управления

Число вершин в алгоритме управления $\left(L_{\max}\right)$	**Максимальное число узлов и наборов листьев в *R*-выражениях** $\left(N_{L_{\max}} = N_{T_{\max}}\right)$	**Аппаратная сложность акселератора, ЭВ**
100	10	80 928
1 000	32	2 034 328
10 000	100	60 816 488

Таким образом, аппаратная сложность акселератора при обработке алгоритмов управления, содержащих 10 000 вершин, составляет величину порядка 60 млн. ЭВ, что позволяет его ПЛИС реализацию в ближайшей временной перспективе [26], а заказное исполнение возможно с использованием современных техпроцессов уже в настоящее время.

15. Оценка быстродействия акселератора

Анализ выигрыша во времени обработки, получаемого с использованием схем выполнения элементарных операций над *R*-выражениями, показывает, что на всех этапах наблюдается снижение времени обработки как минимум в *N* раз. Теоретически это позволяет уменьшить время, затрачиваемое на построение множества сечений, с 2,8 года [14] (при $N = 10\,000$) до 2,4 часа. Покажем, что реально снижение времени обработки является еще более существенным.

Оценим время обработки пары *R*-выражений как сумму времен переходных процессов в комбинационных цепях акселератора при выполнении всех операций преобразования *R*-выражений. Для этого в каждой из схем необходимо выделение наиболее длинной цепи логических элементов, дающей максимальную задержку.

Время работы схем СППВД определяется как $t_{СППВД} = t_{ОСЭМД1} + t_{MUX} = 1 + \left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{MUX}$, где $t_{ОСЭМД1}$ – задержка формирования сигнала на выходе памяти, а t_{MUX} – задержка формирования сигнала на выходе мультиплексора. С учетом того, что мультиплексор может быть синтезирован различными способами, значение t_{MUX} может изменяться в достаточно широких пределах: $t_{MUX} \in \left[t_{MUX_{\min}}; t_{MUX_{\max}}\right]$. Минимальное значение задержки достигается в случае синтеза мультиплексора непосредственно по ДНФ формулы, задающей принцип его работы. Для мультиплексора $8 \to 1$, как известно, формула имеет следующий вид:

$$F = E\left(d_0\overline{a}_2\overline{a}_1\overline{a}_0 \vee d_1\overline{a}_2\overline{a}_1 a_0 \vee d_2\overline{a}_2 a_1\overline{a}_0 \vee \ldots \vee d_7 a_2 a_1 a_0\right),$$

где E – сигнал разрешения работы; d_i, $i = \overline{0,7}$ – информационные входы; a_i, $i = \overline{0,2}$ – адресные входы. При этом величина $t_{MUX_{\min}} = 4t_0$ ($3t_0$ в случае отсутствия необходимости во входе E). Максимальное значение задержки достигается при использовании пирамидальной схемы увеличения разрядности мультиплексора (из мультиплексоров $n \to 1$ конструируется мультиплексор $n^2 \to 1$): $t_{MUX_{\max}} = 3t_0\left\lceil \log_2 n \right\rceil^{\left(2^m\right)}$, где $\lceil x \rceil^{\left(2^m\right)}$ – операция округления вверх до ближайшей целой степени двойки, $\lceil x \rceil^{\left(2^m\right)} = 2^{\lceil \log_2 \lceil x \rceil \rceil}$ (например, $t_{MUX_{\max}}$ при $n = 10\,000$ составляет $3t_0 \cdot 16 = 48t_0$). Таким образом,

$$t_{СППВД} \in \left[\left(\left\lceil \log_2 N_{T_{\max}} \right\rceil + 4\right)t_0; \left(\left\lceil \log_2 N_{T_{\max}} \right\rceil + 1 + 3\left\lceil \log_2 L_{\max} \right\rceil^{\left(2^m\right)}\right)t_0\right].$$

Время работы схемы СРМД определяется как

$$t_{СРМД} = t_{СРМД_0} + t_{СРМД_1},$$

где $t_{СРМД_0}$ – длительность инициализации (от переключений сигнала ρ до записи значения в регистр 5), $t_{СРМД_1}$ – длительность обработки, состоящая из длительности обработки наборов листьев и длительности обработки узлов. Переключение коммутатора 3, работа дешифратора 4 и запись значения в регистр 5 производятся за время $2 + t_{DC} + 2$, причем $t_{DC} \in \left[2t_0; 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)}\right]$ (аналогично рассмотренному выше, разрядность дешифратора, как и разрядность мультиплексора, допускает наращивание с использованием пирамидальной схемы). Таким образом, $t_{СРМД_0} \in \left[12t_0; 8t_0 + 4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)}\right]$ (инициализация проводится дважды).

При обработке наборов листьев одной из наиболее длинных цепочек логических элементов является цепочка 5–МВ–7–9–11–12–13–15–17–20–МУ: время задержки распространения сигнала по ней составляет

$$t_0 + t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{СПБ} + t_0 + t_0 + 2t_0 + t_{SUM} + t_0 + t_0 + t_0 + t_0 =$$
$$= 9t_0 + t_0 \log_2 N_{T_{\max}} + t_{СПБ} + t_{SUM}.$$

Время работы схемы подсчета бит зависит от деталей ее схемотехнической реализации и определяется величинами $t_{СПБ_{\min}} = 3t_0$ (синтез по ДНФ) и $t_{СПБ_{\max}} = \left(n + 1 + 2\left\lceil \log_2 n \right\rceil\right)t_0$ (синтез с использованием схем упорядочения двоичного вектора, выделения старшей единицы и шифратора). Время работы сумматора также зависит от принципа его построения: $t_{SUM_{\min}} = 3t_0$ (синтез по ДНФ), $t_{SUM_{\max}} = 2\left(n-1\right)t_0$ (сумматор с последовательным переносом). Таким образом, длительность обработки наборов листьев находится в следующих пределах:

$$\left[15t_0\overline{n}_L; \left(4 + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + L_{\max} + 4\left\lceil \log_2 L_{\max} \right\rceil\right)t_0\overline{n}_L\right].$$

При обработке узлов одной из наиболее длинных цепочек логических элементов является цепочка 5–СП(у)–6–8–ТУ–9–11–12–13–15–17–20–МУ: время задержки распространения сигнала по ней составляет

$$t_0 + t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2t_0 + t_{DC} + t_0 + t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_0 + t_0 + 2t_0 +$$
$$+ t_{SUM} + t_0 + t_0 + t_0 + t_0 =$$
$$= 12t_0 + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{DC} + t_{SUM}.$$

Длительность обработки узлов находится в следующих пределах:

$$\left[\left(17 + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil\right)t_0\overline{n}_T; \left(10 + 4\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)}\right)t_0\overline{n}_T\right].$$

Таким образом,

$$t_{СРМД_1} \in \left[\begin{array}{l} 15t_0\overline{n}_L + \left(17t_0 + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil\right)\overline{n}_T; \\ \left(4t_0 + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + L_{\max}t_0 + 4t_0\left\lceil \log_2 L_{\max} \right\rceil\right)\overline{n}_L + \\ + \left(10t_0 + 4t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)}\right)\overline{n}_T \end{array}\right].$$

С учетом того, что $\overline{n}_T \le \overline{n}_L$ и $\overline{n}_R \approx 2\overline{n}_L$, получаем, что

$$t_{СРМД_1} \in \left[\begin{array}{l}\left(16+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R; \\ \left(7+2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}+2\left\lceil \log_2 N_{T_{\max}} \right\rceil+\dfrac{L_{\max}}{2}+2\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R\end{array}\right],$$

$$t_{СРМД} \in \left[\begin{array}{l}\left(16+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R + 12t_0; \\ \left(7+2\left\lceil \log_2 N_{T_{\max}} \right\rceil+\dfrac{L_{\max}}{2}+2\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R + 8t_0 + \left(2\overline{n}_R+4\right) t_0 \left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\end{array}\right].$$

Время работы схемы СПОНВ определяется как

$$t_{СПОНВ} = t_{СПОНВ_0} + t_{СПОНВ_1} + t_{СПОНВ_2},$$

где $t_{СПОНВ_0} = t_{СРМД_0}$ – длительность инициализации, $t_{СПОНВ_1}$ – длительность обработки наборов листьев, $t_{СПОНВ_2}$ – длительность обработки узлов. Одной из наиболее длинных цепочек логических элементов при обработке наборов листьев является цепочка В.МВ–14–19–20–А.СП(нл)–22–23–11–А.ТС(у): время задержки распространения сигнала по ней составляет

$$t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_0 + t_0 + t_0\left\lceil \log_2 L_{\max} \right\rceil + \left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + t_{DC} + 2t_0 + t_0 + t_0 =$$

$$= 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_0\left\lceil \log_2 L_{\max} \right\rceil + 7t_0 + t_{DC}.$$

Длительность обработки наборов листьев находится в следующих пределах:

$$t_{СПОНВ_1} \in \left[\begin{array}{l}\left(2\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 L_{\max} \right\rceil + 9\right) t_0 \overline{n}_L; \\ \left(2\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)} + \left\lceil \log_2 L_{\max} \right\rceil + 7\right) t_0 \overline{n}_L\end{array}\right].$$

Одной из наиболее длинных цепочек логических элементов при обработке узлов является цепочка А.СП(у)–33–А.ТС(у)–СОТСП–24–10–А.ТС(у): время задержки распространения сигнала по ней составляет

$$\left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + t_{DC} + \left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + t_0 + 2t_0 + t_0 + t_0 =$$

$$= 7t_0 + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{DC},$$

а длительность обработки узлов находится в пределах

$$t_{СПОНВ_2} \in \left[\left(9+2\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_T; \left(7+2\left\lceil \log_2 N_{T_{\max}} \right\rceil+2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 \overline{n}_T\right].$$

Таким образом,

$$t_{СПОНВ} \in \left[\begin{array}{l}12t_0 + \left(9+2\left\lceil \log_2 N_{T_{\max}} \right\rceil+\dfrac{1}{2}\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R; \\ \left(8+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 + \\ +\left(7+2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}+2\left\lceil \log_2 N_{T_{\max}} \right\rceil+\dfrac{1}{2}\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R\end{array}\right].$$

Время работы схемы СУП определяется как

$$t_{СУП} = t_{СУП_0} + t_{СУП_1} + t_{СУП_2},$$

где $t_{СУП_0} = t_{СРМД_0}$ – длительность инициализации, $t_{СУП_1}$ – длительность обработки наборов листьев, $t_{СУП_2}$ – длительность обработки узлов. Одной из наиболее длинных цепочек логических элементов при обработке наборов листьев является цепочка А.НС(нл)–9–10–7–В.УЭ(нл): время задержки распространения сигнала по ней составляет

$$t_0\left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) + 2t_0 + t_{DC} + t_0 + t_0 = 5t_0 + t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{DC},$$

а

$$t_{СУП_1} \in \left[\left(7+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_L; \left(5+\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 \overline{n}_L\right].$$

При обработке узлов наиболее длинной цепочкой является цепочка A.НС(у)–9–10–17–В.УЭ(у), при этом можно заметить, что время задержки распространения сигнала по ней совпадает с полученным выше значением $5t_0 + t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_{DC}$. Таким образом,

$$t_{СУП} \in \left[\begin{array}{l} 12t_0 + \left(7+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R; \\ \left(8+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 + \left(5+\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 \overline{n}_R \end{array}\right].$$

Время работы схемы СВП определяется как

$$t_{СВП} = t_{СВП_0} + t_{СВП_1} + t_{СВП_2},$$

где $t_{СВП_0} = t_{СРМД_0}$ – длительность инициализации, $t_{СВП_1}$ – длительность копирования наборов листьев, $t_{СВП_2}$ – длительность копирования узлов. Одной из наиболее длинных цепочек логических элементов при обработке наборов листьев является цепочка 14–СМНП 2–15–В.УЭ(нл)–СВМЕ–48–47–В.СП(нл): время задержки распространения сигнала по ней составляет

$$t_0 + t_{СМНП} + 2t_0 + 4t_0 + t_{СВМЕ} + t_0 + 2t_0 + 3t_0 = 13t_0 + t_{СМНП} + t_{СВМЕ}.$$

Время задержки распространения сигнала схемы маскировки неиспользуемых позиций находится в пределах от $t_{СМНП_{\min}} = 3t_0$ (синтез по ДНФ) до $t_{СМНП_{\max}} = \left(2\left\lceil \log_2 n \right\rceil^{\left(2^m\right)} + n\right) t_0$ (синтез с использованием дешифратора и цепочки элементов И). Время задержки распространения сигнала схемы выделения младшей единицы находится в диапазоне от $t_{СВМЕ_{\min}} = 3t_0$ (синтез по ДНФ) до $t_{СВМЕ_{\max}} = (n+1)t_0$ (синтез с использованием цепочки элементов запрета). Таким образом,

$$t_{СВП_1} \in \left[19t_0; \left(14 + 3N_{T_{\max}} + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0\right].$$

Одной из наиболее длинных цепочек логических элементов при обработке узлов является цепочка 18–19–20–21–В.СП(у): время задержки распространения сигнала по ней составляет $t_{CD} + t_{SUM} + t_{DC} + 2t_0 + 3t_0$, т.е.

$$t_{СВП_2} \in \left[11t_0; \left(4\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)} + 3\right) t_0\right].$$

Общее время работы схемы СВП определяется как

$$t_{СВП} \in \left[\begin{array}{l} (12+15\overline{n}_R) t_0; \\ \left(8+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)}\right) t_0 + \left(\frac{17}{2} + \frac{3}{2} N_{T_{\max}} + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{\left(2^m\right)} + \left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R \end{array}\right].$$

Одной из наиболее длинных цепочек логических элементов, определяющей время работы схемы СРС, является цепочка 2–В.СП(у)–В.СП(нл)–В.МВ–7–В.МВ, время распространения сигнала по которой составляет

$$t_{DC} + \left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + \left(2+\left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil\right) t_0 + \left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + t_0 + t_0 =$$

$$= t_{DC} + 6t_0 + 2t_0\left\lceil \log_2 N_{T_{\max}} \right\rceil + t_0\left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil,$$

а время работы схемы находится в следующих пределах:

$$t_{СРС} \in \left[\begin{array}{l} \left(8+2\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil\right) t_0; \\ \left(6+2\left\lceil \log_2 N_{T_{\max}} \right\rceil + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + \left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil\right) t_0 \end{array}\right].$$

Время работы схемы СУПП определяется как $t_{СУПП} = t_{СУПП_1} + t_{СУПП_2}$, где $t_{СУПП_1}$ – длительность обработки наборов листьев, $t_{СУПП_2}$ – длительность обработки узлов. Одной из наиболее длинных цепочек логических элементов при обработке наборов листьев является цепочка 17–2–СМНП 1–3–4–СВМЕ–СМНП 2–7–СВМН–В.СП(нл)–9–В.СП(нл): время задержки распространения сигнала по ней составляет

$$\begin{aligned} & t_0 + 2t_0 + t_{СМНП} + t_0 + t_0 + t_{СВМЕ} + t_{СМНП} + \\ & + t_0 + t_{СВМН} + \left(1+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 + 2t_0 + 3t_0 = \\ & = 12t_0 + 2t_{СМНП} + 2t_{СВМЕ} + \left\lceil \log_2 N_{T_{\max}} \right\rceil t_0, \end{aligned}$$

причем $t_{СВМЕ} = t_{СВМН}$. При этом

$$t_{СУПП_1} \in \left[\begin{array}{l} \left(24+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_L; \\ \left(18+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + \left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_L \end{array}\right].$$

Одной из наиболее длинных цепочек логических элементов при обработке узлов является цепочка 1–2–СМНП 1–3–4–СВМЕ–СМНП 2–7–СВМН–В.СП(у)–8–В.СП(у): время задержки распространения сигнала по ней совпадает с временем распространения сигнала при обработке наборов листьев, а

$$t_{СУПП_2} \in \left[\begin{array}{l} \left(24+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_T; \\ \left(18+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + \left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_T \end{array}\right].$$

Общее время работы схемы СУПП составляет

$$t_{СУПП} \in \left[\begin{array}{l} \left(24+\left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R; \\ \left(18+4\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + \left\lceil \log_2 N_{T_{\max}} \right\rceil\right) t_0 \overline{n}_R \end{array}\right].$$

Максимальное время обработки акселератором пары *R*-выражений (при условии наличия возможности проведения подстановки) определяется временем работы схем преобразований:

$$t_{обр} = \max\left(t_{СППВД}, t_{СРМД} + t_{СР}\right) + t_{СПОНВ} + t_{СУП} + t_{СВП} + t_{СРС} + t_{СУПП},$$

где $t_{СР}$ – время работы комбинационных схем сравнения ω-мощностей 19 и 20. При этом $\max\left(t_{СППВД}, t_{СРМД} + t_{СР}\right) = t_{СРМД} + t_{СР}$, т.к. $t_{СППВД} < t_{СРМД}$. С учетом того, что время работы схемы сравнения совпадает с временем работы сумматора, время обработки пары выражений системы Ξ определяется как

$$t_{обр} \in \left[\begin{array}{l} \left(71+5\left\lceil \log_2 N_{T_{\max}} \right\rceil + \frac{1}{2}\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R + \left(56+2\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil\right) t_0; \\ \left(\frac{91}{2}+7\left\lceil \log_2 N_{T_{\max}} \right\rceil + \frac{3}{2}N_{T_{\max}} + L_{\max} + 2\left\lceil \log_2 L_{\max} \right\rceil\right) t_0 \overline{n}_R + \\ + \left(38+(11\overline{n}_R+18)\left\lceil \log_2 N_{T_{\max}} \right\rceil^{(2^m)} + 2\left\lceil \log_2 N_{T_{\max}} \right\rceil + \left\lceil \log_2 \left\lceil \log_2 N_{T_{\max}} \right\rceil \right\rceil\right) t_0 \end{array}\right].$$

Верхняя оценка временем задержки логического элемента составляет 0,94 нс. С учетом формулы для расчета $t_{обр}$ и времени t_0 можно получить предельные значения времени обработки пары *R*-выражений в секундах.

Таблица 7. Предельные затраты времени на обработку пары выражений системы Ξ

$L_{\max}$	$N_{T_{\max}}$	$\overline{n}_R$	$t_{обр}$
100	10	5	$[557t_0; 1653t_0]$ $[0{,}5\ мкс; 1{,}6\ мкс]$
1 000	32	16	$[1\,847t_0; 20\,093t_0]$ $[1{,}7\ мкс; 18{,}9\ мкс]$
10 000	100	50	$[6\,073t_0; 518\,229t_0]$ $[5{,}7\ мкс; 0{,}5\ мс]$

Указанные в таблице 7 затраты времени учитывают только время обработки, но не учитывают время загрузки исходных данных и выгрузки результирующего *R*-выражения. *R*-выражение имеет следующий размер в битах, определяемый совокупностью размеров образующих его полей:

$$2\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + \left(4 + 3\log_2 N_{T_{\max}}\right)\overline{n}_T + \left(L_{\max} + 2\log_2 N_{T_{\max}} + 3\right)\overline{n}_L =$$
$$= 2\left\lfloor \log_2 N_{T_{\max}} \right\rfloor + \left(\frac{7}{2} + \frac{5}{2}\log_2 N_{T_{\max}} + \frac{1}{2}L_{\max}\right)\overline{n}_R.$$

Таблица 8. Средний размер *R*-выражений (в байтах) при обработке алгоритмов управления с различным числом вершин

$L_{\max}$	$N_{T_{\max}}$	$\overline{n}_R$	Размер
100	10	5	325 Б (0,04 КБ)
1 000	32	16	8 308 Б (1,01 КБ)
10 000	100	50	251 064 Б (30,7 КБ)

С учетом того, что передача данных производится в пакетном режиме, пропускная способность PCI Express x1 составляет 0,5 ГБ/с [27], а при обработке пары выражений системы Ξ производится загрузка четырех *R*-выражений и выгрузка двух, время обмена акселератора с памятью составляет $\frac{0{,}04\ КБ \times 6}{0{,}5\ ГБ/с} = 0{,}48\ мкс$ при $L_{\max} = 100$, $\frac{1{,}01\ КБ \times 6}{0{,}5\ ГБ/с} = 12\ мкс$ при $L_{\max} = 1\,000$ и $\frac{30{,}7\ КБ \times 6}{0{,}5\ ГБ/с} = 368\ мкс$ при $L_{\max} = 10\,000$. Общее время обработки пары выражений системы Ξ, выраженное суммой времен загрузки исходных данных в акселератор, их обработки и выгрузки в память, приведено в таблице 4.8.

Таблица 9. Общее время обработки пары выражений системы Ξ

$L_{\max}$	$N_{T_{\max}}$	$\overline{n}_R$	t_{Ξ}
100	10	5	$[0{,}98\ мкс; 2{,}08\ мкс]$
1 000	32	16	$[13{,}7\ мкс; 30{,}9\ мкс]$

10 000	100	50	$[0{,}37\ мс; 0{,}87\ мс]$

С учетом того, что в худшем случае производится $\frac{\tilde{N}_R\left(\tilde{N}_R-1\right)}{2}+\tilde{N}_R$ операций с *R*-выражениями, где $\tilde{N}_R \approx \frac{L_{\max}}{10}$ – число выражений системы Ξ, общее время обработки *R*-выражений с использованием акселератора составляет $T_R=\left(\frac{\tilde{N}_R\left(\tilde{N}_R-1\right)}{2}+\tilde{N}_R\right)t_{\Xi}$. Численные значения времени обработки приведены в таблице 10.

Таблица 10. Общее время обработки *R*-выражений

$L_{\max}$	$N_{T_{\max}}$	$\overline{n}_R$	t_{Ξ}
100	10	5	$[53{,}9\ мкс; 114\ мкс]$
1 000	32	16	$[69{,}1\ мс; 156\ мс]$
10 000	100	50	$[3\ мин; 7{,}3\ мин]$

По сравнению с программной реализацией выигрыш во времени обработки при использовании разработанного акселератора (при условии использования наиболее быстрых вариантов построения комбинационных схем) составляет 520 раз при $L_{\max}=100$, 3 228 раз при $L_{\max}=1\,000$ и 23 233 раз при $L_{\max}=10\,000$.

16. Выводы

1) Предложена структурно-функциональная организация комбинаторно-логического акселератора для быстрой обработки конструктивных подмножеств вершин граф-схем параллельных алгоритмов (*R*-выражений), основанная на использовании многопортовой ассоциативном памяти, векторной обработке множеств вершин (наборов листьев) и комбинационных схемах, учитывающих специфику выполняемых операций.
2) Приведена оценка аппаратной сложности акселератора в эквивалентных вентилях, составляющая 60 млн. ЭВ при обработке граф-схем параллельных алгоритмов из 10 000 вершин. Реализация ускорителя возможна как с использованием ПЛИС, так и при заказном исполнении.
3) Приведена оценка быстродействия ускорителя, демонстрирующая выигрыш от нескольких сотен до нескольких десятков тысяч раз.

Список литературы

1. Зотов И.В., Колосков В.А., Титов В.С. Выбор оптимальных разбиений алгоритмов при проектировании микроконтроллерных сетей // Автоматика и вычислительная техника. 1997. № 5. С. 51–62.
2. Организация и синтез микропрограммных мультимикроконтроллеров / Зотов И.В. и др.; Курск: Изд-во «Курск», 1999. 368 с.
3. Поиск базового сечения в задаче разбиения параллельных алгоритмов / Ватутин Э.И., Зотов И.В.; КГТУ. Курск, 2003. 30 с. Деп. в ВИНИТИ 24.11.03 № 2036-В2003.

4. Ватутин Э.И., Зотов И.В., Титов В.С. Построение множества сечений в задаче оптимального разбиения параллельных управляющих алгоритмов // Известия ТулГУ. Вычислительная техника. Информационные технологии. Системы управления. Тула: ТулГУ, 2003. Т. 1. Вып. 2. С. 70–77.
5. Ватутин Э.И., Зотов И.В. Метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Параллельные вычисления и задачи управления (PACO'04). М.: Институт проблем управления им. В.А. Трапезникова РАН, 2004. С. 884–917.
6. Ватутин Э.И., Зотов И.В. Параллельно-последовательный метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Свидетельство об официальной регистрации программы для ЭВМ № 2005613091 от 28.11.05.
7. Ватутин Э.И., Зотов И.В. Повышение качества разбиения алгоритмов при синтезе логических мультиконтроллеров с использованием метода параллельно-последовательной декомпозиции // Перспективы развития систем управления оружием: сборник докладов IV научно-практической конференции, Курск, 19-20 сентября 2007 г. М.: Изд-во «Бедретдинов и Ко», 2007. С. 84–92.
8. Ватутин Э.И. Интересные свойства *R*-выражений в задаче синтеза разбиений параллельных алгоритмов управления // Молодежь и XXI век. Ч. 1. Курск: изд-во КурскГТУ, 2008. С. 30–31.
9. Ватутин Э.И., Зотов И.В., Титов В.С. Выявление изоморфных вхождений *R*-выражений при построении множества сечений параллельных алгоритмов логического управления // Информационно-измерительные и управляющие системы. № 11, Т. 7. М.: «Радиотехника», 2009. С. 49–56.
10. Комбинаторно-логические задачи синтеза разбиений параллельных алгоритмов логического управления при проектировании логических мультиконтроллеров / Э.И. Ватутин, И.В. Зотов, М.Ю. Сохен, В.С. Титов. Курск: изд-во КурскГТУ, 2010. 200 с.
11. Ватутин Э.И., Зотов И.В. Программная система для построения разбиений параллельных управляющих алгоритмов // Идентификация систем и задачи управления (SICPRO'06). М.: Институт проблем управления им. В.А. Трапезникова РАН, 2006. С. 2239–2250.
12. Ватутин Э.И., Зотов И.В. Визуальная среда синтеза разбиений параллельных алгоритмов логического управления // Свидетельство об официальной регистрации программы для ЭВМ № 2007613222 от 30.07.07.
13. Ватутин Э.И., Волобуев С.В., Зотов И.В. Комплексная сравнительная оценка методов выбора разбиений при проектировании логических мультиконтроллеров // Идентификация систем и задачи управления (SICPRO'08). М.: Институт проблем управления им. В.А. Трапезникова РАН, 2008. С. 1917–1940.
14. Ватутин Э.И., Волобуев С.В., Зотов И.В. Комплексный сравнительный анализ качества разбиений при синтезе логических мультиконтроллеров в условиях присутствия технологических ограничений // Параллельные вычисления и задачи управления (PACO'08). М.: Институт проблем управления им. В.А. Трапезникова РАН, 2008. С. 643–685.
15. Ватутин Э.И. Однородная среда электронной модели дерева для аппаратно-ориентированной обработки *R*-выражений // Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации (Распознавание – 2008). Ч. 1. Курск: изд-во КурскГТУ, 2008. С. 90–92.
16. Ватутин Э.И., Зотов И.В., Титов В.С. Использование схемных формирователей и преобразователей двоичных последовательностей при построении комбинаторно-логических акселераторов // Известия КурскГТУ, 2008. № 4 (25). С. 32–39.
17. Комбинаторные аппаратные модели и алгоритмы в САПР / В.М. Курейчик, В.М. Глушань, Л.И. Щербаков. М.: Радио и связь, 1990. 216 с.
18. А. с. 1273941 СССР, МКИ[3] G06F 15/20. Устройство для разбиения графа на подграфы / В.М. Глушань, Л.И. Щербаков, И.П. Левин. Опубл. 1986, Бюл. № 44.
19. http://ru.wikipedia.org/wiki/Сортировка_пузырьком
20. http://algolist.manual.ru/sort/bubble_sort.php
21. AMD Athlon Processor x86 Code Optimization Guide. Order Number #22007 // http://developer.amd.com, 2000. 320 p.
22. Ватутин Э.И. Оптимизация обработки множеств // Медико-экологические информационные технологии. Курск: изд-во КурскГТУ, 2005. С. 145–147.
23. Угрюмов Е.П. Цифровая схемотехника. СПб: БХВ – Санкт-Петербург, 2000. 526 с.
24. Ватутин Э.И., Зотов И.В. Аппаратная модель для определения минимального числа блоков при декомпозиции параллельных алгоритмов логического управления // Известия вузов. Приборостроение. 2008. Т. 51, № 2. С. 39–43.
25. Ватутин Э.И., Зотов И.В., Титов В.С. Алгоритм и устройство выявления изоморфных вхождений *R*-выражений при построении множества сечений параллельных алгоритмов логического управления // Известия вузов. Приборостроение. 2009. Т. 52, № 2. С. 37–45.
26. http://www.ixbt.com/news/all/index.shtml?11/30/11
27. http://ru.wikipedia.org/wiki/PCI_Express