Построение деревьев разводки сигнала

Т. Р. Сытдыков

В работе рассмотрена задача о возможности построения дерева разводки сигнала с заданными величинами задержек сигнала до листьев дерева. Исследован класс деревьев с функциями задержки сигнала, на которые наложены некоторые ограничения. Предложен алгоритм, решающий данную задачу. Для фиксированного набора функций задержки сигнала время работы алгоритма полиномиально относительно количества листьев дерева.

Ключевые слова: Синтез больших интегральных схем, разводка сигнала, дерево буферов.

Введение

В такой казалось бы сугубо инженерной области, как синтез больших интегральных схем, традиции использования математических моделей чрезвычайно глубоки и восходят к работам К.Шеннона [1], С.В.Яблонского [2, 3] и О.Б.Лупанова [4, 5], посвященных синтезу таких управляющих систем, как контактные схемы и схемы из функциональных элементов. Далее было очень много работ, посвященных этой тематике, среди которых можно выделить работы А.Е.Андреева [6, 7]. Хороший обзор по

сложности синтеза управляющих систем можно найти в работах В.Б.Кудрявцева [8, 9]. Во всех этих работах мерой сложности управляющих систем является количество функциональных элементов, и эта мера сложности совсем не учитывает длину проводов. В работе В.А.Ли [10] исследуется задача укладки, минимизирующая длину проводов, но ее модель довольно абстрактная и не рассматривает такой вопрос, как именно будут проходить провода. Управляюшая система, в которой имеются коммуникационные элементы, соответствующие проводам, появилась в работе С.С.Кравцова [11] и называется клеточными или плоскими схемами. Далее эта управляющая система развивалась в работах Н.А.Шкаликовой [12, 13]. Другая мера сложности, называемая мощностной, связанная с потребляемой схемой мощностью, исследуется в работах О.М.Касим-Заде [14], Ю.С.Шуткина [15, 16] и Г.В.Калачева [17, 18, 19], соответственно для схем из функциональных элементов, контактных схем и плоских схем. Из работ С.С.Кравцова, Н.А.Шкаликовой и Г.В.Калачева следует, что в плоских схемах, наиболее близких к интегральным схемам, доля коммуникационных элементов на порядок больше чем функциональных элементов, что делает проблему разводки проводов очень актуальной. Одна из таких проблем, связанная с разводкой проводов в интегральных схемах, рассматривается в данной работе. Так при синтезе больших интегральных схем часто возникают сети с большим ветвлением, т. е. ситуации, когда сигнал с выхода некоторого логического элемента нужно подать на входы большого числа других элементов, например, когда результат некоторого вычисления может быть использован сразу во многих местах. Если

при этом выход исходного управляющего элемента напрямую подать на входы управляемых элементов, то емкость на выходе управляющего элемента будет очень большой и задержка данного элемента станет неприемлемой. В таких случаях используют деревья разводки сигнала, которые состоят из буферов, т. е. логических элементов, реализующих тождественную функцию и использующихся для усиления сигнала. Типичным примером сети с большим ветвлением является сеть синхронизации регистров, где регистр — это устройство, позволяющее сохранить результат вычисления предыдущего такта для использования в течение последующего такта. В сети синхронизации регистров управляющий контакт — это выход устройства, реализующего периодический сигнал (часы), а управляемые контакты — это управляющие входы регистров, которые указывают моменты, когда надо засасывать в регистр новое значение. Задача построения сети синхронизации регистров — задача построения такого дерева буферов, что сигнал от управляющего контакта до любого из управляемых контактов доходит за одинаковое время, т. е. синхронно. Задача построения синхронизирующих деревьев описывается и исследуется, например, в [20, 21, 22, 23]. Теперь рассмотрим ситуацию, когда делается два последовательных вычисления, которые сохраняются на регистрах, причем первое вычисление сложное и задается формулой большой глубины, и результат этого вычисления подается на вход первого регистра, а второе вычисление простое, задается формулой малой глубины, одним из аргументов этой формулы является выход первого регистра, и результат второго вычисления сохраняется на втором регистре. Поскольку глубина формулы первого вы-

числения большая, а вычисление надо выполнить за один такт, то, чтобы выполнить ограничения по таймингу, придется использовать мощные дорогостоящие элементы для реализации этой формулы. Между тем, если послать отпирающий сигнал на управляющий вход первого регистра с некоторой задержкой относительно второго регистра, т. е. не синхронно, то время на вычисление первой формулы увеличится, а на время на вычисление второй формулы уменьшится, тем самым удовлетворить ограничения тайминга будет легче. Использование такого приема описано, например, в [24]. Тем самым асинхронная доставка отпирающего сигнала до регистров может существенно облегчить процесс синтеза чипов, делая их более дешевыми. В [20] можно найти и другие примеры, когда управляющий сигнал надо доставить до управляемых контактов за разное время, причем время задержки до каждого из управляемых контактов строго определенное. В [25] подробно рассмотрен простейший частный случай задачи построения деревьев разводки сигнала, где в качестве функции задержки сигнала в вершинах дерева использовалась тождественная функция.

В математической постановке эта задача может быть сформулирована следующим образом.

Пусть дано ориентированное корневое дерево. Будем считать, что от корня дерева к листьям вдоль ребер дерева распространяется сигнал. В каждой вершине дерева сигнал испытывает задержку, которая вычисляется в соответствии с некоторой функцией. В общем случае разным вершинам дерева могут соответствовать разные функции. На всем пути от корня к некоторой вершине α сигнал испытывает задержку, равную сумме

задержек во всех вершинах, входящих в путь, исключая саму α. Таким образом, для каждой вершины задержка сигнала будет целым неотрицательным числом, которое однозначно восстанавливается по структуре дерева. Для данного дерева можно составить мультимножество $\{a_1, a_2, \ldots, a_n\}$ — величины задержек сигнала для листьев дерева, где n — число листьев.

Исследуемая задача заключается в следующем. Пусть дано мультимножество $\{a_1, a_2, \ldots, a_n\}$ целых неотрицательных чисел. Пусть также дан набор функций, которые могут быть функциями задержки сигнала в вершинах дерева. Требуется определить, существует ли дерево, мультимножество задержек сигнала до листьев которого совпадает с заданным мультимножеством. Если такое дерево существует, мультимножество назовем *реализуемым*, а дерево — *реализующим* данное мультимножество.

Поставленная задача будет сведена к более простой задаче, для которой будет получен алгоритм, решающий ее за время $O(n^p)$, где n — мощность мультимножества, а величина p будет зависеть от заданного набора функций. Таким образом, для фиксированного набора функций время работы алгоритма полиномиально зависит от мощности мультимножества.

Автор выражает благодарность профессору Э. Э. Гасанову за постановку задачи и помощь в работе.

Постановка задачи и формулировка результатов

Постановка задачи

Будем использовать следующие обозначения:

- $\mathfrak{M}$ — множество всех мультимножеств целых неотрицательных чисел;

- $E = \{0\}$ — тривиальное мультимножество;
- $M(A)$ — максимальный элемент мультимножества A;
- $out(\alpha)$ — исходящая степень вершины α.

Далее будем использовать следующие формы записи мультимножества A:

- $\{a_1, a_2, \ldots, a_n\}$ — перечисление элементов мультимножества;
- $(k_0, k_1, k_2, \ldots, k_m)$ — вектор кратностей элементов, входящих в мультимножество, где $m \geq M(A)$, k_i — количество элементов, равных i, в данном мультимножестве, $i = 0, 1, \ldots, m$.

Пусть задан непустой конечный набор функций $F = (f_1, f_2, \ldots, f_l)$, причем произвольная функция $f_i, 1 \leq i \leq l$, удовлетворяет следующим условиям:

- Область определения функции f_i — множество натуральных чисел за вычетом единицы, область значения функции — некоторое подмножество множества натуральных чисел:

$$\mathbf{D}(f_i) = \mathbb{N} \setminus \{1\}, \ \mathbf{Im}(f_i) \subseteq \mathbb{N}, \tag{1}$$

- f_i является неубывающей на всей области определения:

$$\forall\ x \in \mathbb{N} \setminus \{1\}\ f_i(x) \leq f_i(x+1), \tag{2}$$

- При достаточно больших x функция $f_i(x)$ асимптотически доминирует над логарифмической функцией:

$$f_i(x) = \omega(\log_2 x) \tag{3}$$

В дальнейшем из этих условий будет выведено следующее соотношение:

$$\begin{aligned}
&\forall\ i = 1, 2, \ldots, l\ \exists\ r_i \in \mathbb{N} \setminus \{1\} : \\
&\forall\ x \in \mathbb{N},\ x > r_i\ \exists \text{ набор } y_1, y_2, \ldots, y_s,\ s \geq 2, \\
&\forall\ j = 1, 2, \ldots, s\ \ 2 \leq y_j \leq r_i, \\
&\prod_{j=1}^{s} y_j \geq x,\ \sum_{j=1}^{s} f_i(y_j) \leq f_i(x).
\end{aligned} \tag{4}$$

Числа r_i будут играть важную роль в получении итогового результата.

Далее, пусть дано ориентированное корневое дерево. Отнесем каждую вершину α дерева с исходящей степенью, не меньшей чем 2, к одному из l типов вершин, занумерованных числами от 1 до l, и обозначим соответствующий вершине тип как t_α: $t_\alpha = i, 1 \leq i \leq l$. Будем говорить, что функция $f_i \in F$ *характеризует* i - й тип вершин дерева, и для краткости будем также обозначать эту функцию как f_α.

Будем считать, что по дереву от корня к листьям распространяется сигнал. Для любой вершины α дерева, не являющейся листом, определим величину *добавочной задержки сигнала в вершине (добавочной задержки)* c_α следующим образом:

$$c_\alpha = \begin{cases} f_\alpha(x), & \text{если } x \text{ — исходящая степень } \alpha,\ x \geq 2; \\ c, & \text{если исходящая степень } \alpha \text{ есть единица,} \end{cases}$$

где c — некоторое целое неотрицательное число.

Для любой вершины α дерева определим величину *задержки сигнала (задержки)* z_α в данной вершине: она равна сумме добавочных задержек вершин дерева, лежащих на пути от корня дерева к вершине α, исключая саму α. Задержку сигнала

до корня по определению положим равной 0. Из определения и свойств функций из набора F следует, что для любой вершины задержка сигнала будет целым неотрицательным числом.

Данному дереву можно сопоставить мультимножество $A = \{a_1,\ a_2,\ \ldots,\ a_n\}$, состоящее из величин задержек сигнала до листьев дерева, где n — число листьев дерева.

Исследуемая задача заключается в следующем. По заданному непустому конечному набору функций $F = (f_1,\ f_2,\ \ldots,\ f_l)$, удовлетворяющих условиям (1) – (3), а также заданному мультимножеству целых неотрицательных чисел $A = \{a_1, a_2, \ldots, a_n\}$ требуется определить, существует ли дерево, мультимножество задержек до листьев которого совпадает с A. Если такое дерево существует, будем называть мультимножество A *реализуемым*, и будем говорить, что дерево *реализует мультимножество* A.

Будем использовать следующие обозначения:

- $\mathfrak{M}_F$ — множество всех реализуемых мультимножеств;
- $\mathfrak{D}$ — множество всех ориентированных корневых деревьев, некоторым вершинам которых приписаны функции из F в соответствии с правилами, описанными выше;
- $J(\mathbf{D})$ — мультимножество, реализуемое деревом $\mathbf{D}$.

Таким образом, если $\exists\ \mathbf{D} \in \mathfrak{D}\colon J(\mathbf{D}) = A$, то $A \in \mathfrak{M}_F$, в противном случае $A \notin \mathfrak{M}_F$.

Пусть $a \in \mathbb{N}\backslash\{1\}$. Среди функций из набора F выберем такую функцию, значение которой при аргументе a не превосходит значений других функций из F при аргументе a, и обозначим ее f^a (если таких функций несколько, выберем произвольную из них).

Пусть далее $r = max_{i=1}^{l} r_i$, где r_i — число, фигурирующее в соотношении (4). Определим функцию $g\colon \{1, 2, 3, \ldots, r\} \to \mathbb{N}$, следующим образом.

$$g(a) = \begin{cases} 1, & a = 1 \\ f^a(a), & 2 \le a \le r. \end{cases} \tag{5}$$

Функция g будет использована при получении асимптотики итогового алгоритма, решающего задачу о реализуемости.

Формулировка результатов

Для задачи о реализуемости получена следующая теорема.

Теорема 1. *Пусть $F = (f_1, f_2, \ldots, f_l)$ — непустой конечный набор функций, удовлетворяющих условиям (1) – (3), $A \in \mathfrak{M}$, $n = |A|$, операции извлечения и удаления максимального элемента мультимножества выполняются за $O(1)$. Тогда задача о реализуемости A решается за $O(r \cdot l + g(2) \cdot g(r) \cdot n^{g(r)+1} \cdot S(n, r))$, где r — число, фигурирующее в соотношении (4), $S(n, r)$ — количество способов разбиения числа n на неупорядоченные слагаемые, не превосходящие r, g — функция, определяемая из соотношений (5).*

Если рассматривать данную задачу для фиксированного F и различных мультимножеств, то величины $l, r, g(2), g(r)$ можно считать константами. $S(n, r)$ — величина, которую можно достаточно грубо оценить следующим образом:

$$S(n, r) \le (n + 1)^{r-1}.$$

Учитывая это, получаем следующее утверждение.

Следствие 1. *Пусть* $F = (f_1, f_2, \ldots, f_l)$ *— фиксированный непустой конечный набор функций, удовлетворяющих условиям (1) – (3),* $A \in \mathfrak{M}$, $n = |A|$, *операции извлечения и удаления максимального элемента мультимножества выполняются за* $O(1)$. *Тогда задача о реализуемости* A *решается за* $O(n^{r+g(r)})$.

Простейшие свойства реализуемых мультимножеств

Пусть A — произвольное мультимножество, содержащее некоторое число a, и $v \in \mathbb{Z}_+$. Введем следующий оператор:

$G(A, a, v) = A \cup \{a + v\} \setminus \{a\}$.

Если A — мультимножество, $a \in A$, $|A| \geq 2$, то введем оператор

$H(A, a) = A \setminus \{a\}$.

Будем называть G *оператором увеличения элемента*, а H *— оператором удаления элемента*. Далее для краткости будем опускать числовые аргументы операторов увеличения элемента и удаления элемента.

Утверждение 1. *Пусть* $A \in \mathfrak{M}_F$, $B = V_1 * V_2 * \cdots * V_k(A)$, *где* V_i *— либо оператор увеличения элемента, либо оператор удаления элемента,* $i = 1, 2, \ldots, k$. *Тогда* $B \in \mathfrak{M}_F$.

Доказательство. Достаточно показать, что операторы увеличения элемента и удаления элемента не выводят из класса реализуемых мультимножеств.

1) Пусть $A \in \mathfrak{M}_F$, a — произвольный элемент из A, $v \in \mathbb{Z}_+$, $C = G(A, a, v)$. Рассмотрим дерево **D**, реализующее A, и заме-

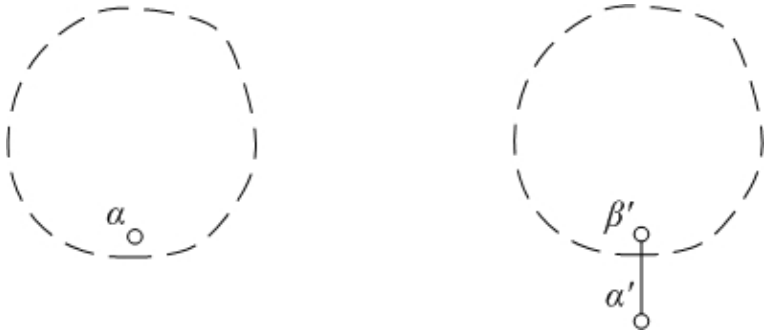

Рис. 1: Преобразование дерева для оператора увеличения элемента.

ним лист α, задержка которого равна a, на цепочку из двух вершин β' и α', где β' — родитель α' (рис. 1). Так как $out(\beta') = 1$, в качестве $c_{\beta'}$ можно положить произвольное целое неотрицательное число. Положим $c_{\beta'} = v$, тогда полученное дерево реализует мультимножество

$$A \cup \{a + v\} \setminus \{a\} = G(A, a, v) = C,$$

т. е. мультимножество C реализуемо.

2) Пусть $A \in \mathfrak{M}_F$, a — произвольный элемент из A, $C = H(A, a)$. Рассмотрим дерево $\mathbf{D}$, реализующее A, и возьмем лист α с задержкой a. Рассмотрим путь p от корня дерева к α. Так как оператор H определен только при $|A| \geq 2$, то существует лист $\beta \in \mathbf{D}$, $\beta \neq \alpha$. Следовательно, $\exists\ \gamma \in p\colon out(\gamma) \geq 2$. Без ограничения общности будем считать, что все вершины p, расположенные между γ и α, имеют исходящую степень, равную 1.

Удалим из $\mathbf{D}$ часть пути от корня к α, расположенную после γ. Обозначим всех оставшихся сыновей γ как $\kappa_1, \kappa_2, \ldots, \kappa_m$, $m \geq 1$. Заменим вершину γ на цепочку из двух вершин γ' и δ', где δ' — родитель γ', и в качестве родителя κ_i,

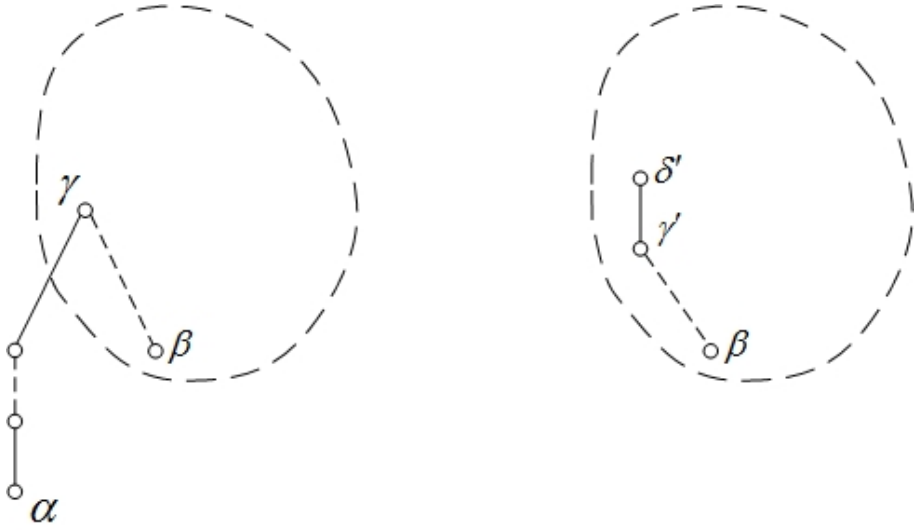

Рис. 2: Преобразование дерева для оператора удаления элемента.

$i = 1, \ldots, m$, выберем γ' (рис. 2). В зависимости от исходящей степени γ до преобразования возможны два варианта.

а) $out(\gamma) = 2$. Тогда $out(\gamma') = out(\delta') = 1$. Следовательно, в качестве добавочных задержек γ' и δ' можно положить 0 и c_γ соответственно.

Посмотрим, как изменились величины задержек до листьев дерева после преобразования. Задержки до листьев, не имеющих γ' в качестве предка, не изменились. Задержки до листьев, являющихся потомками γ', совпадают с соответствующими задержками листьев - потомков γ (кроме удаленного листа α), так как они изменились на $-c_\gamma + c_{\delta'} + c_{\gamma'} = 0$. Следовательно, полученное дерево реализует мультимножество $A \setminus \{a\} = H(A, a)$.

b) $out(\gamma) > 2$. Тогда $out(\gamma') = k > 1$, $out(\delta') = 1$. Пусть $t_\gamma = r$. Положим $t_{\gamma'} = r$, тогда $c_{\gamma'} = f_r(k) \leq f_r(k+1) = c_\gamma$ (неравенство следует из свойства (2) функции f_r). Положим $c_{\delta'} = c_\gamma - c_{\gamma'}$.

Посмотрим, как изменились величины задержек до листьев дерева после преобразования. Задержки до листьев, не имеющих γ' в качестве предка, не изменились. Задержки до листьев, являющихся потомками γ', совпадают с соответствующими задержками листьев - потомков γ (кроме удаленного листа α), так как они изменились на $-c_\gamma + c_{\delta'} + c_{\gamma'} = 0$. Следовательно, полученное дерево реализует мультимножество $A \setminus \{a\} = H(A, a)$.

В обоих случаях мультимножество $A \setminus \{a\}$ реализуемо.

■

Замечание. Если отождествить вершины α и α', рассмотренные при преобразовании для оператора увеличения элемента, можно утверждать следующее. В преобразованиях дерева, соответствующих оператору увеличения элемента, появилась одна новая вершина с исходящей степенью 1. Исходящие степени и типы остальных вершин, если они были определены, не изменились.

Если отождествить вершины γ и γ', рассмотренные при преобразовании для оператора удаления элемента, можно утверждать следующее. В преобразованиях дерева, соответствующих оператору удаления элемента, появилась одна новая вершина, исходящая степень которой равна 1. Кроме того, исходящая степень еще одной вершины была понижена на 1, при этом если ее первоначальная исходящая степень была больше 1, то ее тип не изменился. Исходящие степени и типы остальных вершин, если они были определены, не изменились.

Эти факты будут использованы в дальнейшем.

Пусть $a \in \mathbb{N} \setminus \{1\}$. Напомним, что функция из набора F, значение которой на аргументе a не больше значений других функций из F на аргументе a, была обозначена как f^a (если таких функций несколько, в качестве f^a может быть любая из них).

Утверждение 2. *Пусть* $A \in \mathfrak{M}_F$. *Тогда* $\exists\ \boldsymbol{D}_0 \in \mathfrak{D}\colon\ J(\boldsymbol{D}_0) = A$, $\forall\ \alpha \in \boldsymbol{D}_0\colon$

$$\textit{если } out(\alpha) \geq 2,\ \textit{то } f_\alpha = f^{out(\alpha)}. \tag{6}$$

Доказательство. Рассмотрим дерево $\mathbf{D}$, реализующее A. Пусть $\exists\ \alpha \in \mathbf{D}\colon\ out(\alpha) = x \geq 2,\ f_\alpha \neq f^x$. Заменим тип вершины α на тип, соответствующий f^x, и посмотрим, как изменились задержки листьев поддерева α. Так как $f^x(x) \leq f_\alpha(x)$, то эти задержки уменьшатся на величину $v = f_\alpha(x) - f^x(x)$. Но тогда, если преобразовать дерево $\mathbf{D}$ так, как было описано при доказательстве утверждения 1 для случая оператора увеличения элемента, то, согласно замечанию к утверждению 1, у всех вершин, исходящие степени которых не меньше 2, типы не изменятся, и новых таких вершин не появится.

Следовательно, полученное преобразованиями дерево $\mathbf{D}'$ будет реализовывать A, причем количество вершин дерева, не удовлетворяющих условию (6), уменьшится на 1. Но тогда, в силу конечности количества вершин дерева, многократно применяя описанную процедуру, можно получить дерево $\mathbf{D}_0$, реализующее A, все вершины которого удовлетворяют условию (6).

■

Далее необходимо вывести рассмотренное ранее соотношение (4).

Утверждение 3. *Пусть функция f удовлетворяет следующим условиям:*

$$\boldsymbol{D}(f) = \mathbb{N} \setminus \{1\},$$
$$f(x) = \omega(\log_2 x).$$

Тогда будет выполнено следующее соотношение:

$$\exists\, r \in \mathbb{N} \setminus \{1\} :$$
$$\forall\, x \in \mathbb{N},\ x > r\ \exists \textit{ набор } y_1, y_2, \ldots, y_s,\ s \geq 2,$$
$$\forall\, j = 1, 2, \ldots, s\ \ 2 \leq y_j \leq r,$$
$$\prod_{j=1}^{s} y_j \geq x,\ \sum_{j=1}^{s} f(y_j) \leq f(x).$$

Доказательство. Так как $f(x) = \omega(\log_2 x)$, то $\exists\, r \in \mathbb{N} \setminus \{1\}$: $\forall\, x \in \mathbb{N},\ x > r\ f(x) \geq f(2) \cdot (\log_2 x + 1)$. Рассмотрим произвольное $x > r$ и положим $s = \lceil \log_2 x \rceil$, $y_1 = y_2 = \cdots = y_s = 2$. Тогда получим $\prod_{j=1}^{s} y_j = 2^{\lceil \log_2 x \rceil} \geq 2^{\log_2 x} = x$. В то же время $\sum_{j=1}^{s} f(y_j) = \lceil \log_2 x \rceil \cdot f(2) < (\log_2 x + 1) \cdot f(2) \leq f(x)$.

■

Замечание. Покажем, что для функции $\log_2 x$ утверждение будет неверным. Предположим, что соответствующее число r существует. Возьмем произвольное простое число x, превосходящее r (такое число всегда существует) и покажем, что для него соотношение не выполнено. Для произвольного набора $y_1, y_2, \ldots, y_s,\ s \geq 2, 2 \leq y_j \leq r$, если выполнено соотношение $\prod_{j=1}^{s} y_j = p \geq x$, то в силу простоты числа x получим $p > x$. В то же время $\sum_{j=1}^{s} \log_2 y_j = \log_2 p > \log_2 x$, что противоречит утверждению.

Замечание. Число r, фигурирующее в утверждении, играет существенную роль в алгоритме, рассмотренном далее. В част-

ности, оно влияет на асимптотику алгоритма — чем меньше r, тем быстрее будет работать алгоритм. Поэтому важным является вопрос получения этого числа для заданной функции Из доказательства утверждения следует, что для нахождения r достаточно исследовать множество решений неравенства $f(x) \geq f(2) \cdot (\log_2 x + 1)$. В то же время полученное значение r не обязательно будет минимально возможным. В частности, для простейшего случая тождественной функции $f(x) = x$, исследуя неравенство, получим $r = 7$, в то же время минимально возможным значением r является 3 (для $x = 4$ можно положить $y_1 = y_2 = 2$, для $x = 5$ и $x = 6$ достаточно взять $y_1 = 2, y_2 = 3$, для $x = 7$ допустимо $y_1 = y_2 = 3$). Задача поиска для заданной функции f минимально возможного r выходит за рамки данной статьи. В дальнейшем будем считать, что если задан набор функций $F = (f_1,\ f_2,\ \ldots,\ f_l)$, то соответствующие числа $r_1, r_2, \ldots, r_l$ известны заранее.

Утверждение 4. *Пусть* $A \in \mathfrak{M}_F$, $r = max_{i=1}^{l} r_i$. *Тогда* $\exists\ \boldsymbol{D}_0 \in \mathfrak{D}$: $J(\boldsymbol{D}_0) = A$, $\forall\ \alpha \in \boldsymbol{D}_0\ out(\alpha) \leq r$.

Доказательство. Рассмотрим дерево $\mathbf{D}$, реализующее A. Пусть $\exists\ \alpha \in \mathbf{D}$: $out(\alpha) = x > r$. Согласно условию (4), $\exists$ набор $y_1, y_2, \ldots, y_s : \forall\ j = 1, 2, \ldots, s\ \ 2 \leq y_j \leq r,\ s \geq 2$, $\prod_{j=1}^{s} y_j \geq x$, $\sum_{j=1}^{s} f_\alpha(y_j) \leq f_\alpha(x)$. Пусть сыновья α есть $\gamma_1, \gamma_2, \ldots, \gamma_x$.

Преобразуем дерево следующим образом (рис. 3).

Удалим всех потомков α и вместо них добавим y_1 сыновей $\beta_1, \beta_2, \ldots,\ \beta_{y_1}$. К каждой из вершин β_i, $i = 1, 2, \ldots, y_1$, в свою очередь, добавим y_2 сыновей $\beta_{i1}, \beta_{i2}, \ldots, \beta_{iy_2}$. Продолжая аналогично, после s шагов получим $y_1 \cdot y_2 \cdot \cdots \cdot y_s$ вершин $\beta_{i_1 i_2 \ldots i_s}$, где $\forall\ j = 1, 2, \ldots, s\ 1 \leq i_j \leq y_j$. В силу того, что $\prod_{j=1}^{s} y_j \geq x$,

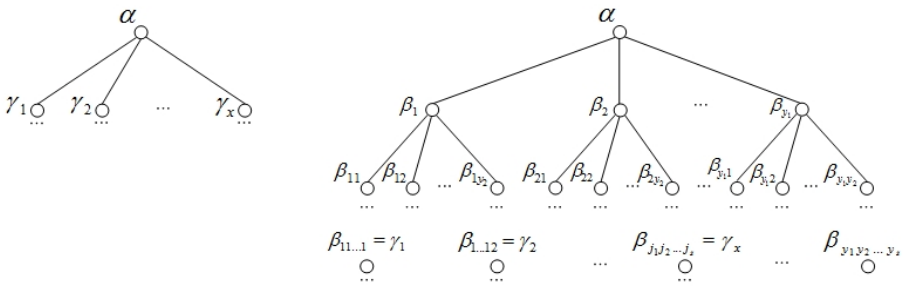

Рис. 3: Преобразование дерева для избавления от вершины с большой исходящей степенью.

можно сопоставить первые x из этих вершин с сыновьями α до преобразования $\gamma_1, \gamma_2, \ldots, \gamma_x$, и провести из каждой из них соответствующее поддерево. Всем остальным вершинам, полученным на некотором шаге в ходе преобразований, таким, что их исходящие степени превосходят 1, сопоставим функцию f_α, и заметим, что их исходящие полустепени не больше r. Назовем полученное дерево $\mathbf{D}'$.

Посмотрим, как изменились задержки листьев дерева после этих преобразований. Задержки до листьев, не являющихся потомками α, не изменились. Задержки до листьев - потомков γ_i, $i = 1, 2, \ldots, s$, изменились на величину $\sum_{j=1}^{s} f_\alpha(y_j) - f_\alpha(x) \leq 0$. Кроме того, появилось $\prod_{j=1}^{s} y_j - x$ новых листьев. Но тогда, используя замечание к утверждению 1, получим, что можно преобразованиями увеличения и удаления элементов получить из $\mathbf{D}'$ новое дерево $\mathbf{D}''$ такое, что $J(\mathbf{D}'') = J(\mathbf{D})$, при этом исходящая степень вершины α стала не больше r, и добавились новые вершины с исходящей степенью не больше r.

Таким образом, получена процедура преобразования дерева, не меняющая мультимножество задержек до его листьев и уменьшающая на 1 число вершин с полустепенью исхода больше r. Но тогда, применяя описанную процедуру многократно, можно получить такое дерево $\mathbf{D}_0$, что $J(\mathbf{D}_0) = J(\mathbf{D})$ и $\forall\ \phi \in \mathbf{D}_0\ out(\phi) \leq r$.

■

Таким образом, из утверждения 4 следует, что для решения задачи о реализуемости достаточно рассматривать деревья с ограниченными исходящими степенями вершин. В свою очередь, из утверждения 2 следует, что достаточно рассматривать деревья, у которых вершины с одинаковой полустепенью исхода имеют одинаковый тип, а именно, если для вершины α $out(\alpha) = a > 1$, то всегда можно рассматривать только случай $f_\alpha = f^a$.

Напомним, что функция $g\colon \{1, 2, 3, \ldots, r\} \to \mathbb{N}$ была определена следующим образом:

1) $g(1) = 1$;

2) $\forall\ a \in \{2, 3, \ldots, r\}\ g(a) = f^a(a)$.

Рассмотрим произвольное дерево $\mathbf{D}$, не содержащее вершин с полустепенью исхода выше r, такое, что любая вершина с полустепенью исхода $a > 1$ характеризуется функцией f^a. Заменим все вершины α с полустепенью исхода, равной 1, и задержкой сигнала в вершине, равной c_α, на цепочку вершин длины c_α. Каждая вершина цепочки будет иметь полустепень исхода, равную 1, и задержку сигнала в вершине, равную 1. При таком преобразовании $J(\mathbf{D})$ не изменится. После преобразования мож-

но считать, что в любой вершине α дерева, если $out(\alpha) = a \geq 1$, то $c_\alpha = g(a)$.

Таким образом, вместо заданного набора функций F достаточно рассматривать единственную функцию g, при этом формально считая, что эта функция вычисляет задержку также в вершинах дерева с полустепенью исхода, равной 1, и не рассматривать деревья, в которых имеются вершины с полустепенью исхода, превосходящей r. Далее будем рассматривать задачу в такой постановке.

Операторная формулировка задачи о реализуемости

Пусть $A = (k_0, k_1, \ldots, k_m) \in \mathfrak{M}$, $n = |A| = \sum_{i=0}^{m} k_i$. $\forall\ i = 1, \ldots, m$, $\forall\ j = 1, \ldots, r$ определим оператор g_i^j, действующий на мультимножествах. Пусть $k_i \geq j$, $i \geq g(j)$. Тогда

$g_i^j(A) = (k_0, k_1, \ldots, k_{i-g(j)-1}, k_{i-g(j)} + 1, k_{i-g(j)+1}, \ldots, k_{i-1}, k_i - j, k_{i+1}, \ldots, k_m)$.

В противном случае полагаем, что $g_i^j(A)$ не определен.

Множество всех допустимых для мультимножества A операторов будем обозначать как $\mathfrak{G}(A)$.

Множество всех операторов $g_i^j, 1 \leq i \leq m, 1 \leq j \leq r$ будем обозначать как $\mathfrak{G}_m^r$.

Если $g_i^j(A)$ определен, будем говорить, что g_i^j *является допустимым преобразованием* или просто *преобразованием* для A.

$\forall$ цепочки преобразований $g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}$ мультимножество $A' = g_{i_1}^{j_1}\ (g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots))$ назовем *достижимым из* A (при условии, что все преобразования цепочки являются допустимыми для своих аргументов).

Пусть $A = (k_0, k_1, \ldots, k_m) \in \mathfrak{M}_F$, $\mathbf{D} \in \mathfrak{D}$, $J(\mathbf{D}) = A$. Пусть $\lambda_1, \lambda_2, \ldots, \lambda_v$ — листья $\mathbf{D}$, имеющие общего родителя α, причем $out(\alpha) = v$. Тогда $z(\lambda_1) = z(\lambda_2) = \cdots = z(\lambda_v) = a$ — некоторое число, не меньшее чем $g(v)$.

Рассмотрим дерево $\mathbf{D}'$, полученное из $\mathbf{D}$ удалением листьев $\lambda_1, \lambda_2, \ldots, \lambda_v$. Тогда α станет листом $\mathbf{D}'$. Учитывая, что $z(\alpha) = a - g(v)$, получим $J(\mathbf{D}') = (k_0, k_1, \ldots, k_{a-g(v)-1}, k_{a-g(v)} + 1, k_{a-g(v)+1}, \ldots, k_{a-1}, k_a - v, k_{a+1}, \ldots, k_m) = g_a^v(A)$.

В дальнейшем подобное преобразование дерева $\mathbf{D}$ будем обозначать как $G_i^j(\mathbf{D})$, по аналогии с преобразованием реализуемого $\mathbf{D}$ мультимножества. Таким образом, $\mathbf{D}' = G_i^j(\mathbf{D})$.

Можно продолжать преобразовывать $\mathbf{D}'$ аналогичным способом до получения тривиального дерева $\mathbf{D}_0$, состоящего из одного корня. При этом $J(\mathbf{D}_0) = \{0\} = E$. Тогда, если l — число вершин $\mathbf{D}$, не являющихся листьями, получим для некоторых $i_1, i_2, \ldots, i_l$; $j_1, j_2, \ldots, j_l$ соотношение $E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots))$.

Следовательно, задачу о реализуемости мультимножества A можно свести к задаче получения E из A последовательным применением операторов из $\mathfrak{G}_m^r$, где m — максимальный элемент A.

Аналогично, задача получения E из заданного мультимножества A применением операторов из $\mathfrak{G}_m^r$ может быть сведена к задаче о реализуемости A. Поэтому можно считать эти задачи эквивалентными и полагать, что задача получения E из мультимножества A является операторной формулировкой задачи о реализуемости A.

Общий алгоритм решения задачи о реализуемости мультимножества

Утверждение 5. *Пусть* $A \in \mathfrak{M}_F$. *Тогда* $\exists\ g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}\colon\ E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots))$, *причем* $\{i_k\}$ — *неубывающая последовательность.*

Доказательство. Достаточно доказать следующее. Пусть $A \in \mathfrak{M}_F$, тогда $\exists\ g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}\colon E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots))$, причем $i_l = \max_k i_k$. Тогда, применяя это рассуждение к A, $g_{i_l}^{j_l}(A)$ и т. д., докажем утверждение.

Пусть $\mathbf{D} \in \mathfrak{D}_F$, $J(\mathbf{D}) = A$. Пусть $m = M(A)$. Если $m = 0$, то $A = E$, и утверждение верно. Пусть $m > 0$. Тогда по крайней мере один из операторов $G_m^1, G_m^2, \ldots, G_m^r$ будет применим к $\mathbf{D}$. Без ограничения общности, пусть это оператор G_m^1. Тогда дерево $\mathbf{D}' = G_m^1(\mathbf{D})$ реализует мультимножество $A' = g_m^1(A)$. В силу реализуемости $A'\ \exists\ g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_{l-1}}^{j_{l-1}}, E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_{l-1}}^{j_{l-1}}(A')\ldots))$. Значит, $E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_{l-1}}^{j_{l-1}}(g_m^1(A))\ldots))$. Так как $m = M(A)$, то $\forall\ k = 1, \ldots, l-1\ \ i_k \le m$. Положим $i_l = m, j_l = 1$, получим оператор $g_{i_l}^{j_l}$, причем $i_l = \max_k i_k$. Следовательно, утверждение доказано.

■

Любую цепочку операций $g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}$, соответствующую условиям утверждения 5, назовем *жадной реализацией мультимножества* A.

Далее нам понадобится следующее определение.

Пусть $A = (k_0, k_1, \ldots, k_m) \in \mathfrak{M}$, $m = M(A)$, $h \in \{0, 1, \ldots, m-1\}$. *Оптимальным преобразованием мультимножества* A *по-*

рядка h (или просто *оптимальным преобразованием*) назовем мультимножество $A_h(A) = (k_0, k_1, \ldots, k_{h-1}, k_h + 1)$.

Для другой формы записи мультимножеств оптимальное преобразование будет выглядеть следующим образом. Пусть $A = \{a_1, a_2, \ldots, a_n\}$, $a_1 \le a_2 \le \cdots \le a_n$. Тогда $\forall\ h = 0, 1, \ldots, a_1 - 1\ A_h(A) = \{h\}$. $\forall\ h = a_1, a_1 + 1, \ldots, a_n - 1\ \exists\ i \in \{1, 2, \ldots, n - 1\}$: $a_i \le h < a_{i+1}$, и получим $A_h(A) = \{a_1, a_2, \ldots, a_i, h\}$.

Утверждение 6. *Пусть* $A \in \mathfrak{M}$, $A' = A_h(A)$, A' *достижимо из* A. *Тогда задача о реализуемости* A *эквивалентна задаче о реализуемости* A'.

Доказательство. 1) Если A' реализуемо, то E достижимо из A'. Так как A' достижимо из A, то и E достижимо из A. Значит, A реализуемо.

2) Пусть A реализуемо. Будем считать, что $A \ne E$, так как по определению для E не существует оптимальных преобразований. Рассмотрим некоторую жадную реализацию A, т. е. непустую цепочку допустимых преобразований $g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}$ такую, что $E = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots)), i_1 \le i_2 \le \cdots \le i_l = m$. Тогда либо $\exists\ l_1 \in \{1, 2, \ldots, l - 1\} : i_{l_1} \le h, i_{l_1+1} > h$, либо $i_1 > h$, и можно формально положить $l_1 = 0$, $i_{l_1} = i_0 = 0$. Рассмотрим мультимножество $A_1 = g_{i_{l_1+1}}^{j_{l_1+1}}(g_{i_{l_1+2}}^{j_{l_1+2}}(\ldots g_{i_l}^{j_l}(A)\ldots))$. Так как $M(A_1) = i_{l_1} \le h$, то A_1 не будет содержать элементов, превосходящих h, т. е. A_1 можно записать как $(k'_0, k'_1, \ldots, k'_h)$. Кроме того, $\forall\ i = 0, 1, \ldots, h\ k'_i \ge k_i$, причем $\exists\ h_1 \in \{0, 1, \ldots, h\} : k'_{h_1} > k_{h_1}$ (например, можно взять $h_1 = i_{l_1+1} - g(j_{l_1+1})$, так как преобразование $g_{i_{l_1+1}}^{j_{l_1+1}}$ добавляет в мультимножество один новый элемент, равный $i_{l_1+1} - g(j_{l_1+1}) \le h$). При этом A_1 является реализуемым. Учитывая, что $A' = (k_0, k_1, \ldots, k_{h-1}, k_h + 1)$, получим, что A' мо-

жет быть получено из A_1 операциями увеличения элемента и удаления элемента. Следовательно, по утверждению 1 A' также является реализуемым.

■

Рассмотрим следующий алгоритм преобразования мультимножества (назовем его *алгоритмом спуска по 2*).

Пусть $A = \{a_1, a_2, \ldots, a_n\}$, $a_1 \le a_2 \le \cdots \le a_n$. На нулевом шаге положим $b = a_n$, или $A = \{a_1, a_2, \ldots, a_{n-1}, b\}$. На i - м шаге положим $S_2(A) = \{a_1, a_2, \ldots, a_{n-i-1},\ b = min(a_{n-i}, b) - g(2)\}$, и далее $A = S_2(A)$, $i = 1, 2, \ldots, n-1$.

Заметим, что $|S_2(A)| = |A| - 1$, и будем использовать этот факт в дальнейшем.

i - й шаг алгоритма меняет два элемента мультимножества. Покажем, что если $min(a_{n-i}, b) - g(2) \ge 0$, то этот шаг может быть выражен через допустимые преобразования g_i^j. Действительно, пусть для определенности $a_{n-i} \ge b$ (случай $a_{n-i} < b$ рассматривается аналогично). Тогда шаг алгоритма эквивалентен следующей цепочке преобразований:

$S_2(A) = g_b^2(g_{b+1}^1(g_{b+2}^1(\ldots(g_{a_{n-i}}^1(A))\ldots)))$.

Утверждение 7. *Пусть* $A = \{a_1, a_2, \ldots, a_n\} \in \mathfrak{M}$, $a_1 \le a_2 \le \cdots \le a_n$, $M(A) \ge g(2) \cdot (|A| - 1)$. *Тогда для полученного на некотором (возможно нулевом) шаге алгоритма спуска по 2 мультимножества* $A' = \{a_1, a_2, \ldots, a_i, b\}$ *получим, что* A' *достижимо из* A, *и будет выполнен хотя бы один из двух вариантов.*

1) $A'' = \{a_1, a_2, \ldots, a_i, a_i\}$ *достижимо из* A' *и является оптимальным преобразованием для* A.

2) $|A'| = 1$.

Доказательство. Если изначально $|A| = 1$, то выполнен вариант 1, т. е. утверждение верно для нулевого шага алгоритма спуска по 2.

Пусть $|A| > 1$. Если $a_n > a_{n-1}$, то $A'' = \{a_1, a_2, \ldots, a_{n-1}, a_{n-1}\}$ достижимо из A и является жадным преобразованием для A, т. е. утверждение верно для нулевого шага алгоритма спуска по 2.

Будем рассматривать теперь случай $a_n = a_{n-1}$. Так как $n \geq 2$, то $a_n = M(A) \geq g(2) \cdot (n-1) \geq g(2) \cdot (2-1) = g(2)$. Значит, $a_n \geq g(2)$, $a_{n-1} \geq g(2)$, т. е. $S_2(A) = \{a_1, a_2, \ldots, a_{n-2}, b\}$, где $b = a_n - g(2)$, будет достижимо из A, и хотя бы один шаг алгоритма спуска по 2 после нулевого шага будет допустимым преобразованием для A.

Далее будем считать, что первый шаг алгоритма был выполнен, и доказывать утверждение для этого случая.

Если на каком-то ненулевом шаге алгоритма спуска по 2 получим

$A' = \{a_1, a_2, \ldots, a_i, b\}$, $|A'| \geq 2$, $b \geq a_i$, $(*)$

то $A'' = \{a_1, a_2, \ldots, a_i, a_i\}$ будет достижимо из A' и является оптимальным преобразованием для A порядка a_i (поскольку $\forall\ j = i+1, i+2, \ldots, n\ a_j > b \geq a_i$). Поэтому для доказательства того, что выполнен вариант 1, достаточно доказать, что на некотором ненулевом шаге i алгоритма спуска по 2 получим $(*)$.

Будем использовать индукцию по номеру последнего шага алгоритма спуска по 2. Покажем, что $\forall\ i = 1, 2, \ldots, n-1$ будет выполнено следующее: либо (i) получено мультимножество из одного элемента (вариант 2), либо (ii) получено мультимножество, удовлетворяющее $(*)$, т. е. варианту 1, либо (iii) оче-

редной $(i+1$ - й) шаг алгоритма спуска по 2 будет допустим, причем для $S_2((S_2)^i(a)) = \{a_1, a_2, \ldots, a_{n-i-2}, b\}$ будет выполнено $b \geq g(2) \cdot (n-1) - g(2) \cdot (i+1)$.

1) База индукции. $i = 1$. Имеем $S_2(A) = \{a_1, a_2, \ldots, a_{n-2}, b\}, b \geq g(2) \cdot (n-1) - g(2) \cdot 1$. Если $n = 2$, то $|S_2(A)| = 1$, и выполнен вариант (i) . Пусть $n > 2$, или $n \geq 3$. Тогда $b \geq g(2) \cdot (3-1) - g(2) \cdot 1 = g(2)$. Если $a_{n-2} \leq b$, то выполнен вариант (ii). В противном случае имеем $a_{n-2} > b \geq g(2)$, т. е. следующий (второй) шаг алгоритма спуска по 2 получит мультимножество, достижимое из A, причем $S_2(S_2(A)) = \{a_1, a_2, \ldots, a_{n-3}, b\}$, $b \geq g(2) \cdot (n-1) - g(2) \cdot 2$, т. е. выполнен вариант (iii).

2) Индуктивный переход. Пусть утверждение индукции верно для $j = 1, 2, \ldots, i-1$, покажем его истинность для i. Если $i = n-1$, то $|((S_2)^{n-1})(A)| = 1$, и выполнен вариант (i). Пусть $i < n-1$, тогда возможны два варианта:

a) $a_{n-i-1} \leq b$, тогда выполнен вариант (ii).

b) $a_{n-i-1} > b$, тогда $a_{n-i-1} > b \geq g(2) \cdot (n-1) - g(2) \cdot i \geq g(2)$, т. е. следующий $(i+1$ - ый) шаг алгоритма спуска по 2 получит мультимножество, достижимое из A, причем $S_2((S_2)^i(A)) = \{a_1, a_2, \ldots, a_{n-i-2}, b\}$, $b \geq g(2) \cdot (n-1) - g(2) \cdot (i+1)$, т. е. выполнен вариант (iii).

Утверждение индукции доказано. Но тогда, в силу ограниченности количества шагов алгоритма спуска по 2, на некотором шаге индукции будет выполнен один их вариантов (i) или (ii), т. е. один из вариантов 1 или 2.

■

Обозначим $\mathfrak{M}^o = \{A \in \mathfrak{M} \colon M(A) < g(2) \cdot (|A| - 1)\}$.

Утверждение 8. *Пусть $A \in \mathfrak{M}$, $|A| = n$, и извлечение / удаление максимального элемента мультимножества выполняется за $O(1)$. Тогда задача реализуемости A сводится либо к задаче о реализуемости некоторого мультимножества $A' \in \mathfrak{M}^o$, либо к задаче о реализуемости некоторого мультимножества $A' = \{a\}$ за время $O(n)$.*

Доказательство. Если $A \in \mathfrak{M}^o$ или $|A| = 1$, то утверждение верно. В противном случае будем применять алгоритм спуска по 2 с проверкой на соответствие очередного полученного мультимножества условиям утверждения. По утверждению 7, на некотором шаге сможем получить либо мультимножество с единственным элементом, либо мультимножество, из которого можно получить оптимальное преобразование A, принадлежащее $\mathfrak{M}^o$.

Осталось показать, что суммарное время выполнения всех действий алгоритма не превосходит $O(n)$. Будем обозначать мультимножество, полученное на очередном шаге алгоритма спуска по 2, как A'. Для получения линейности времени работы алгоритма будем хранить $A' = \{a_1, a_2, \ldots, a_i, b\}$ в виде пары (B, b), где $B = \{a_1, a_2, \ldots, a_i\}$. Тогда значение $M(B)$ может быть получено за $O(1)$, и операция $B = B \backslash \{M(B)\}$ также может быть выполнена за $O(1)$.

Количество k допустимых шагов алгоритма спуска по 2 удовлетворяет соотношению $k \leq n$.

Проверка выполнения условий утверждения занимает константное время (сравнение $|A| = 1$ занимает константное время, сравнение максимального элемента $M(A') = max(M(B), b)$ и величины $g(2) \cdot (n-1)$ также занимает константное время,

сравнение $M(B)$ и b для проверки допустимости оптимального преобразования также занимает константное время).

Один шаг алгоритма спуска по 2 занимает константное время (фактически имеем $S_2(A') = S_2((B,b)) = (B \setminus M(B), min(M(B),b)-2)$, все операции выполняются за $O(1)$).

Следовательно, для общего времени выполнения алгоритма будет верно соотношение $T = O(n)$.

■

Пусть $A = (k_0, k_1, \ldots, k_m) \in \mathfrak{M}$, $m = M(A)$. Обозначим $\mathfrak{L}_p(A) = \{A' \in \mathfrak{M} \mid M(A') \leq p, \exists$ преобразование $G = g_{i_1}^{j_1} \cdot g_{i_2}^{j_2} \cdot \cdots \cdot g_{i_l}^{j_l}$, где $p < i_1 \leq i_2 \leq \cdots \leq i_l$, такое, что $A' = g_{i_1}^{j_1}(g_{i_2}^{j_2}(\ldots g_{i_l}^{j_l}(A)\ldots))\}$.

Рассмотрим произвольное $A' = (k'_0, k'_1, \ldots, k'_p) \in \mathfrak{L}_p(A)$. В силу того, что в цепочке преобразований $g_{i_1}^{j_1}, g_{i_2}^{j_2}, \ldots, g_{i_l}^{j_l}$, преобразующей A в A', $\forall\ s \in \{1,2,\ldots,l\}\ i_s > p,\ j_s \in \{1,2,\ldots,r\}$, получим, что $k'_q = k_q$ при $q = 0,1,\ldots,p-g(r)$, $k'_q \geq k_q$ при $q = p-g(r)+1, p-g(r)+2, \ldots, p$. Таким образом, для кодирования мультимножеств из $\mathfrak{L}_p(A)$ достаточно знать $g(r)$ чисел $k'_p - k_p, k'_{p-1} - k_{p-1}, \ldots, k'_{p-g(r)+1} - k_{p-g(r)+1}$. Для фиксированного A' будем обозначать эти $g(r)$ чисел соответственно как $a_0, a_1, \ldots, a_{g(r)-1}$. При $p < g(r)-1$ не все числа $a_0, a_1, \ldots, a_{g(r)-1}$ имеют смысл с точки зрения допустимости преобразований (например, неясно, как понимать число $k'_{-1} - k_{-1}$), поэтому будем считать, что соответствующие числа имеют формальный смысл. Считаем при этом $k_{-i} = 0\ \forall\ i \in \mathbb{N}$.

По определению, $\mathfrak{L}_m(A) = \{(0,0,\ldots,0)\}$.

Если $(a,0,0,\ldots,0) \in \mathfrak{L}_0(A)$, где $a + k_0 = 1$, то $A \in \mathfrak{M}_F$ (так как получим, что E достижимо из A). При этом верно обратное

(из определения $\mathfrak{L}_0(A)$). Таким образом, задачу о реализуемости мультимножества A можно решать путем нахождения множеств $\mathfrak{L}_p(A)$, $0 \le p \le m = M(A)$, и проверки множества $\mathfrak{L}_0(A)$ на наличие вектора $(a, 0, 0, \ldots, 0)$.

Рассмотрим $A' \in \mathfrak{L}_p(A)$, $p = 0, 1, \ldots, m-1$, и пусть $A' = g^{j_1}_{i_1}(g^{j_2}_{i_2}(\ldots\ g^{j_l}_{i_l}(A)\ldots))$. Тогда возможны два варианта: либо $i_1 > p+1$, либо $\exists\ s : i_s = p+1$, $i_{s+1} > p+1$. В первом случае $A' \in \mathfrak{L}_{p+1}(A)$, во втором $g^{j_{s+1}}_{i_{s+1}}(\ldots g^{j_l}_{i_l}(A)\ldots) \in \mathfrak{L}_{p+1}(A)$. Таким образом, элементы $\mathfrak{L}_p(A)$ получаются из элементов $\mathfrak{L}_{p+1}(A)$ цепочкой преобразований (возможно, пустой цепочкой, т. е. тождественным преобразованием). Следовательно, для получения $\mathfrak{L}_p(A)$ достаточно, имея $\mathfrak{L}_{p+1}(A)$, перебрать преобразования g^j_{p+1} всех элементов, равных $p+1$, принадлежащих мультимножествам из $\mathfrak{L}_{p+1}(A)$.

Утверждение 9. *Пусть $A \in \mathfrak{M}$. Если для $p > 0$ имеется $\mathfrak{L}_p(A)$, существует способ выполнить построение $\mathfrak{L}_{p-1}(A)$ за $O(P \cdot g(r) \cdot S(n,r))$, где $P = |\mathfrak{L}_p(A)|$, $S(n,r)$ — количество способов разбиения числа n на неупорядоченные слагаемые, не превосходящие r.*

Доказательство. Рассмотрим все $A' \in \mathfrak{L}_p(A)$ (т. е. соответствующие вектора $(a_0, a_1, \ldots, a_{g(r)-1})$) — получится P векторов. Покажем, как каждый вектор обработать за $O(g(r) \cdot S(n,r))$ так, чтобы получить из соответствующего ему мультимножества всевозможные достижимые мультимножества $A'' \in \mathfrak{L}_{p-1}(A)$. Тогда, объединяя эти мультимножества и кодируя их соответствующими векторами, получим закодированное представление $\mathfrak{L}_{p-1}(A)$.

Рассмотрим преобразования элементов A', равных p (таких элементов будет $k'_p = k_p + a_0$). Заметим, что $k'_p \le n$. Далее, нуж-

но рассмотреть все возможные преобразования $G = (g_p^{j_1} \cdot g_p^{j_2} \cdot \cdots \cdot g_p^{j_l})$ мультимножества A' такие, что $j_1 + j_2 + \cdots + j_l = k'_p$. Тогда G однозначно определяется вектором $(v_1, v_2, \ldots, v_r)$, где $\forall\ i\ v_i$ есть количество чисел, равных i, среди чисел $j_1, j_2, \ldots, j_l$. Но таких векторов ровно столько, сколько существует способов разбиения числа k'_p на неупорядоченные слагаемые от 1 до r включительно, т. е. $S(k'_p, r)$. В силу того, что $k'_p \leq n$, имеем $S(k_p, r) \leq S(n, r)$. В свою очередь, каждое преобразование G соответствует получению очередного элемента из $\mathfrak{L}_{p-1}(A)$, которое получается по следующей формуле. Введем вектор $V = (v_1, 0, \ldots, 0, v_2, 0, \ldots, 0, v_3, 0, \ldots, 0, v_{r-1}, 0, \ldots, 0, v_r)$, где $\forall\ i = 1, 2, \ldots, r\ v_i$ находится на позиции $g(i)$. Размерность этого вектора будет равна $g(r)$. Обозначим искомый вектор - код элемента $\mathfrak{L}_{p-1}(A)$ за L, тогда $L = V + (a_1, a_2, \ldots, a_{g(r)-1}, 0)$. Это следует из следующих рассуждений: V соответствует элементам, появившимся при преобразовании G, а вектор $(a_1, a_2, \ldots, a_{g(r)-1}, 0)$ — элементам, полученным при предыдущих преобразованиях. Следовательно, вектор L может быть получен за $O(g(r))$ действий.

Таким образом, на генерацию всех элементов из $\mathfrak{L}_{p-1}(A)$ будет потрачено $O(P \cdot g(r) \cdot S(n, r))$ действий.

■

Утверждение 10. *Пусть* $A \in \mathfrak{M}$, $n = |A|$, $m = M(A)$. *Тогда* $\forall\ p \in \{0, 1, \ldots, m\}\ |\mathfrak{L}_p(A)| \leq (n+1)^{g(r)}$.

Доказательство. Каждый элемент $\mathfrak{L}_p(A)$ есть вектор из $g(r)$ элементов. Каждый элемент этого вектора, в свою очередь, есть некоторое число от 0 до n включительно. Количество таких векторов есть в точности $(n+1)^{g(r)}$. Следовательно, $|\mathfrak{L}_p(A)| \leq (n+1)^{g(r)}$.

■

Утверждение 11. *Пусть* $A = (k_0, k_1, \ldots, k_m) \in \mathfrak{M}$, $n = |A|$, $m = M(A)$, *тогда задача о реализуемости* A *решается за* $O(m \cdot g(r) \cdot n^{g(r)} \cdot S(n,r))$, *где* $S(n,r)$ *— количество способов разбиения числа* n *на неупорядоченные слагаемые, не превосходящие* r.

Доказательство. Будем использовать следующий алгоритм. Построим $\mathfrak{L}_m(A), \mathfrak{L}_{m-1}(A), \ldots, \mathfrak{L}_0(A)$ и проверим, имеется ли вектор $(a, 0, \ldots, 0) \in \mathfrak{L}_0(A)$ такой, что $a + k_0 = 1$, ответ на этот вопрос и будет ответом на задачу о реализуемости A.

Построение $\mathfrak{L}_m(A)$ занимает $O(1)$ действий.

$\forall\ p = m, m-1, \ldots, 1$ построение $\mathfrak{L}_{p-1}(A)$ из $\mathfrak{L}_p(A)$, согласно утверждению 9, выполнится за время $O(|\mathfrak{L}_p(A)| \cdot g(r) \cdot S(n,r))$. Согласно утверждению 10, имеем $|\mathfrak{L}_p(A)| \leq (n+1)^{g(r)} = O(n^{g(r)})$. Таким образом, на построение всех $\mathfrak{L}_p(A)$, $p = m, m-1, \ldots, 0$ будет потрачено $O(m \cdot g(r) \cdot n^{g(r)} \cdot S(n,r))$ действий.

Финальная проверка требует не более $g(r) \cdot |\mathfrak{L}_0(A)| \leq g(r) \cdot (n+1)^{g(r)}$ действий.

Таким образом, общее время работы алгоритма будет $O(m \cdot g(r) \cdot n^{g(r)} \cdot S(n,r))$.

■

Теорема 1. *Пусть* $F = (f_1, f_2, \ldots, f_l)$ *— непустой конечный набор функций, удовлетворяющих условиям (1) – (3),* $A \in \mathfrak{M}$, $n = |A|$, *операции извлечения и удаления максимального элемента мультимножества занимают* $O(1)$ *времени. Тогда задача о реализуемости* A *решается за* $O(r \cdot l + g(2) \cdot g(r) \cdot n^{g(r)+1} \cdot S(n,r))$, *где* r *— число, определяемое из соотношения (4),* $g(x)$ *— функция, определяемая соотношением (5),* $S(n,r)$ *—*

количество способов разбиения числа n на неупорядоченные слагаемые, не превосходящие r.

Доказательство. Будем использовать следующий алгоритм.

Шаг 1. Согласно утверждениям 2 и 4, сведем задачу о реализуемости A к задаче с одной функцией g. Это потребует $O(r \cdot l)$ действий на построение функции g по функциям из F (для каждого аргумента i от 2 до r найти функцию f^i из F, и положить $g(i) = f^i(i)$.

Шаг 2. Если $A \notin \mathfrak{M}^o$, применим к A алгоритм спуска по 2. Согласно утверждению 8, в силу того, что операции извлечения и удаления максимального элемента мультимножества занимают $O(1)$ времени, в результате за время $O(n)$ будет получено мультимножество A', причем или $|A'| = 1$ (в этом случае сразу получен ответ — A реализуемо), или $A' \in \mathfrak{M}^o$. Задача о реализуемости A сведена к задаче о реализуемости A'. Для удобства положим теперь $A = A'$.

Шаг 3. Применяя результаты утверждения 11, используя множества $\mathfrak{L}_p(A)$, решим задачу о реализуемости A. Так как на данном шаге $A \in \mathfrak{M}^o$, то $M(A) < g(2) \cdot (n-2) = O(g(2) \cdot n)$. Следовательно, согласно утверждению 11, этот шаг будет выполнен за $O(g(2) \cdot n \cdot g(r) \cdot n^{g(r)} \cdot S(n,r)) = O(g(2) \cdot g(r) \cdot n^{g(r)+1} \cdot S(n,r))$ действий.

Таким образом, суммарное время работы алгоритма есть $O(r \cdot l + g(2) \cdot g(r) \cdot n^{g(r)+1} \cdot S(n,r))$.

■

Если рассматривать задачу о реализуемости для фиксированного набора функций F и различных мультимножеств, то

величины $l, r, g(2), g(r)$ можно считать константами. Учитывая, что $S(n, r) \leq (n+1)^{r-1}$, получаем следующее утверждение.

Следствие 1. *Пусть* $F = (f_1, f_2, \ldots, f_l)$ *— фиксированный непустой конечный набор функций, удовлетворяющих условиям (1) – (3),* $A \in \mathfrak{M}$, $n = |A|$, *операции извлечения и удаления максимального элемента мультимножества занимают* $O(1)$ *времени. Тогда задача о реализуемости* A *решается за* $O(n^{r+g(r)})$.

Список литературы

[1] Shannon C.E. The synthesis of two-terminal switching circuits // Bell Syst. Techn. J. 1949. V 28. N 1. P. 59–98.

[2] Яблонский С.В. О классах функций алгебры логики, допускающих простую схемную реализацию // Успехи математических наук. 1957. Т. 12. Вып. 6. С. 189–196.

[3] Яблонский С.В. Об алгоритмических трудностях синтеза минимальных контактных схем // Проблемы кибернетики. 1959. Вып. 2. С. 75–121.

[4] Лупанов О.Б. О синтезе некоторых классов управляющих систем // Проблемы кибернетики. 1963. Вып. 10. С. 63–97.

[5] Лупанов О.Б. Об одном подходе к синтезу управляющих систем — принципе локального кодирования // Проблемы кибернетики. 1965. Вып. 14. С. 31–110.

[6] Андреев А.Е. Об одном методе получения нижних оценок сложности индивидуальных монотонных функций // ДАН СССР. 1985. Т. 281. № 2. С. 1033–1037.

[7] Андреев А.Е. Универсальный принцип самокорректирования // Математический сборник. 1985. Т. 127 (169). № 6. С. 147–172.

[8] Кудрявцев В.Б., Андреев А.Е. О сложности алгоритмов // Интеллектуальные системы. 2006. Т. 10 (169). Вып. 1–4. С. 695–760.

[9] Кудрявцев В.Б. Кафедра математической теории интеллектуальных систем // Интеллектуальные системы. 2014. Т. 18. Вып. 2. С. 5–30.

[10] Ли В.А. Порядок сложности укладки деревьев на плоскость // Интеллектуальные системы. 2010. Т. 14. Вып. 1–4. С. 417–429.

[11] Кравцов С.С. О реализации функций алгебры логики в одном классе схем из функциональных и коммутационных элементов // Проблемы кибернетики. 1967. Вып.19. С. 285–293.

[12] Шкаликова Н.А. О реализации булевых функций схемами из клеточных элементов // Математические вопросы кибернетики. 1989. Вып.2. С. 177–197.

[13] Шкаликова Н.А. О соотношении сложностей реализации некоторых функций схемами двух видов // Интеллектуальные системы. 2015. Вып.4.

[14] Касим-Заде О.М. Об одной мере активности схем из функциональных элементов. // Математические вопросы кибернетики. Вып. 4. М.: Наука, 1992. С. 218–228.

[15] Шуткин Ю.С. Асимптотически оптимальная реализация булевых функций информационными графами. // Дискретная математика, 23:4 (2011), С. 80–102

[16] Шуткин Ю.С. Моделирование схемных управляющих систем // Интеллектуальные системы. 2014. Т. 18. Вып. 3. С. 253–261.

[17] Kalachev G.V. Order of power of planar circuits implementing Boolean functions // Discrete Mathematics and Applications. 2014. Volume 24, Issue 4, Pages 185–205.

[18] Калачев Г.В. Порядок мощности плоских схем, реализующих булевы функции // Дискретная математика, 2014. Т. 26. Вып. 1. С. 49–74.

[19] Калачев Г.В. Нижние оценки мощности плоских схем, реализующих частичные булевы операторы // Интеллектуальные системы. 2014. Т. 18. Вып. 2. С. 279–322.

[20] Sherwani N. A. Algorithms for VLSI Physical Design Automation. — Kluwer Academic Publishers, 2002.

[21] Гасанов Э. Э., Проворова А. Л. О синтезе синхронизирующих деревьев // Материалы IX Международной конференции «Интеллектуальные системы и компьютерные науки» (23-27 октября 2006 г.), том 1, часть 1. — М.: Изд-во

механико-математического факультета МГУ, 2006. — С. 89–92.

[22] Гасанов Э. Э., Дин А. А. Построение синхронизирующих деревьев // Интеллектуальные системы. 2013. Т. 17. Вып. 1-4. С. 293–297.

[23] Pavisic I., Lu A., Zolotykh A. A., Gasanov E. E. Method in integrating clock tree synthesis and timing optimization for an integrated circuit design // United States Patent: 6,550,044. — April 15, 2003.

[24] Lu A., Pavisic I. Zolotykh A. A., Gasanov E. E. Changing clock delays in an integrated circuit for skew optimization // United States Patent: 6,550,045. — April 15, 2003.

[25] Сытдыков Т. Р. Линейный алгоритм построения деревьев разводки сигнала // Интеллектуальные системы. 2014. Т. 18. Вып. 3. С. 175–202.

Construction of signal routing trees

T. R. Sitdikov

The purpose of this article was the problem of signal routing tree construction with given multiset of signal delay to leaves of the tree. The class of trees with some restrictions to signal delay functions was investigated. It was proposed an algorithm solving the given problem. The found algorithm is polynomial time when the set of signal delay functions is fixed.

Keywords: Synthesis of large-scale integrated circuits, signal routing, tree of buffers.