Binary Adder Circuits of Asymptotically Minimum Depth, Linear Size, and Fan-Out Two

STEPHAN HELD and SOPHIE THERESA SPIRKL, Research Institute for Discrete Mathematics, University of Bonn

We consider the problem of constructing fast and small binary adder circuits. Among widely used adders, the Kogge-Stone adder is often considered the fastest, because it computes the carry bits for two n-bit numbers (where n is a power of two) with a depth of $2\log_2 n$ logic gates, size $4n\log_2 n$, and all fan-outs bounded by two. Fan-outs of more than two are disadvantageous in practice, because they lead to the insertion of repeaters for repowering the signal and additional depth in the physical implementation.

However, the depth bound of the Kogge-Stone adder is off by a factor of two from the lower bound of $\log_2 n$. Two separate constructions by Brent and Krapchenko achieve this lower bound asymptotically. Brent's construction gives neither a bound on the fan-out nor the size, while Krapchenko's adder has linear size, but can have up to linear fan-out. With a fan-out bound of two, neither construction achieves a depth of less than $2\log_2 n$.

In a further approach, Brent and Kung proposed an adder with linear size and fan-out two but twice the depth of the Kogge-Stone adder.

These results are 33–43 years old and no substantial theoretical improvement for has been made since then. In this article, we integrate the individual advantages of all previous adder circuits into a new family of full adders, the first to improve on the depth bound of $2\log_2 n$ while maintaining a fan-out bound of two. Our adders achieve an asymptotically optimum logic gate depth of $\log_2 n + o(\log_2 n)$ and linear size $O(n)$.

CCS Concepts: • **Theory of computation** → *Circuit complexity*; • **Hardware** → **Logic circuits**;

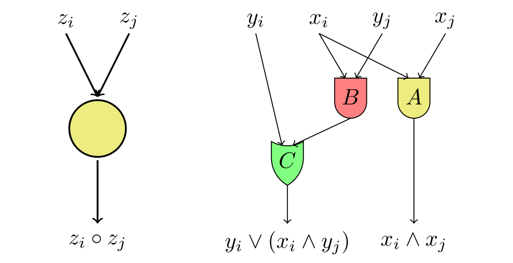
Additional Key Words and Phrases: Binary addition, circuit, combinational complexity, parallel, depth, size, fan-out

ACM Reference format:
Stephan Held and Sophie Theresa Spirkl. 2017. Binary Adder Circuits of Asymptotically Minimum Depth, Linear Size, and Fan-Out Two. *ACM Trans. Algorithms* 14, 1, Article 4 (December 2017), 18 pages.
https://doi.org/10.1145/3147215

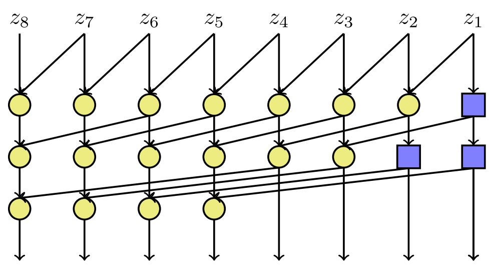
1 INTRODUCTION

The addition of binary numbers is one of the most fundamental computational tasks. Given two binary addends $A = (a_n \dots a_1)$ and $B = (b_n \dots b_1)$, where index n denotes the most significant bit, their sum $S = A + B$ has $n + 1$ bits. We are looking for a logic circuit, also called an *adder*, that computes S. Here, a *logic circuit* is a non-empty connected acyclic directed graph consisting of nodes that are either *gates* with incoming and outgoing edges, *inputs* with at least one outgoing edge

Authors' addresses: S. Held, Research Institute for Discrete Mathematics, University of Bonn, Lennéstraße 2, 53113 Bonn, Germany; email: held@or.uni-bonn.de; S. Spirkl (Current address), Program for Applied and Computational Mathematics, Princeton University, Fine Hall, Washington Road, Princeton, NJ 08544, USA; email: sspirkl@math.princeton.edu.



(a) A prefix gate (left) represents the underlying logic circuit (right) consisting of two AND gates and an OR gate.



(b) The eight-input Kogge-Stone prefix graph consisting of prefix gates and repeaters (blue squares).

Fig. 1. Examples of a prefix gate and a prefix graph.

and no incoming edges, or *outputs* with exactly one incoming edge and no outgoing edges. Gates represent one or two bit Boolean functions, specifically AND, OR, XOR, NOT, or their negations. A small example is shown on the right side of Figure 1(a). The *fan-in* of a gate is the maximum number of incoming edges at the corresponding vertex, and it is bounded by two for all gates.

The main characteristics in adder design are the *depth*, the *size*, and the *fan-out* of a circuit. The depth is defined as the maximum length of a directed path in the logic circuit and is used as a measure for its speed. The lower the depth, the faster is the adder. The size is the total number of gates in the circuit and is used as a measure for the space and power consumption of the adder, both of which we aim to minimize. The fan-out is the maximum number of outgoing edges at a vertex. High fan-outs increase the delay and require additional repeater gates (implementing the identity function) in physical design. Thus, when comparing the depth of adder circuits, their fan-out should be considered as well; we will focus on the usual fan-out bound of two. Circuits with higher fan-outs can be transformed into fan-out two circuits by replacing each interconnect with high fan-out by a balanced binary *repeater tree*, i.e., the underlying graph is a tree and all gates are repeater gates. However, this operation increases the size linearly and the depth with a logarithmic dependence on the fan-out. Hoover et al. (1984) gave a smarter way to bound the fan-out of a given circuit, but it would also triple the size and depth in our case of gates with two inputs.

Using logic circuit depth as a measure for speed is a common practice in logic synthesis that simplifies many aspects of physical hardware. Despite its simplicity, the depth-based model is at the core of programs such as BONNLOGIC (Werber et al. 2007) for refining carry bit circuits, which is an integral part of the current IBM microprocessor design flow. Recently, we reduced the running time for computing such carry bit circuits significantly from $O(n^3)$ to $O(n \log n)$ (Held and Spirkl 2017). In CMOS technology, NAND/NOR gates are faster than AND/OR gates and efficient implementations exist for integrated multi-input AND-OR-Inversion gates and OR-AND-Inversion gates. Therefore, we assume that a *technology mapping* step (Chatterjee et al. 2006; Keutzer 1987) translates the adder circuit after logic synthesis using logic gates that are best for the given technology. In Section 5, we give an example for a technology mapping using only NAND/NOR and NOT gates, and we show how to accomplish this without affecting the key properties of our construction significantly.

Like most existing adders, we use the notion of generate and propagate signals (Sklansky 1960; Brent 1970; Knowles 1999). For each position $1 \leq i \leq n$, we compute a *generate signal* y_i and a *propagate signal* x_i, which are defined as follows:

$$\begin{aligned} y_i &= a_i \wedge b_i, \\ x_i &= a_i \oplus b_i, \end{aligned} \tag{1}$$

where $\wedge$ and $\oplus$ denote the binary AND and XOR functions, respectively. The *carry bit* at position $i+1$ can be computed recursively as $c_{i+1} = y_i \vee (x_i \wedge c_i)$, since there is a carry bit at position $i+1$ if the ith bit of both inputs is 1 or, assuming this is not the case, if at least one (hence exactly one) of these bits is 1 and there was a carry bit at position i.

The first carry bit c_1 represents the *carry-in*, but we usually assume $c_1 = 0$. The last carry bit c_{n+1} is also called the *carry-out*. From the carry bits, we can compute the output S via

$$s_i = c_i \oplus x_i \text{ for } 1 \leq i \leq n \text{ and } s_{n+1} = c_{n+1}. \tag{2}$$

With this preparation step of constant depth, linear size, and fan-out two at the inputs a_i, b_i and fan-out one at the carry bits c_{i+1} $(i = 1, \dots, n)$, we reduce binary addition to the problem of computing all carry bits c_{i+1} from x_i, y_i $(i = 1, \dots, n)$.

Convention: *From now on, we will omit the preparatory steps (1) and (2) and consider a circuit an adder circuit if it computes all c_{i+1} from x_i, y_i $(i = 1, \dots, n)$. Moreover, for simplicity, we will always assume that n is a power of two.*

Expanding the recursive formula for c_{i+1} as in Equation (3) results in a logic circuit that is a path of alternating AND and OR gates. It corresponds to the long addition method and has linear depth $2(n-1)$,

$$c_{i+1} = \; y_i \vee (x_i \wedge (y_{i-1} \vee (x_{i-1} \wedge \cdots \wedge (y_2 \vee (x_2 \wedge y_1)) \dots))) . \tag{3}$$

1.1 Prefix Graph Adders

In this section, we review previous results about prefix graphs, a general framework that is used to construct adders. We show that adders derived from this framework do not have optimal depth due to the way that prefix graphs translate to logic circuits. In later sections, we will use generalizations of prefix graphs to achieve better depth bounds for our adders.

For two pairs $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$, we define the associative *prefix operator* $\circ$ as

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} \circ \begin{pmatrix} x_j \\ y_j \end{pmatrix} = \begin{pmatrix} x_i \wedge x_j \\ y_i \vee (x_i \wedge y_j) \end{pmatrix}. \tag{4}$$

We call a circuit computing (4) a *prefix gate*, and it represents the logic circuit consisting of three gates and with depth two as shown in Figure 1(a). For $i = 1, \dots, n$, the results of the prefix computations $z_i \circ \cdots \circ z_1$ of the expression $z_n \circ \cdots \circ z_1$ contain the carry bit c_{i+1}:

$$\begin{pmatrix} x_i \wedge x_{i-1} \wedge \cdots \wedge x_1 \\ c_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \circ \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} \circ \cdots \circ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}. \tag{5}$$

A circuit of $\circ$-gates computing all prefixes $z_i \circ \cdots \circ z_1$ $(i = 1, \dots, n)$ for an associative operator $\circ$ is called a *prefix graph*. A prefix graph yields an adder by expanding each $\circ$-gate as in Figure 1(a) and extracting the carry bits as in Equation (5).

Based on the variable pairs in Equation (5), the notion of propagate and generate signals can be extended to any consecutive subsequence of $1, \dots, n$: For $1 \leq s \leq t \leq n$, let $X_{s,t}$ and $Y_{s,t}$ denote the propagate and generate signal for the sequence of indices between s and t, i.e.,

$$\begin{aligned} X_{s,t} &= \textstyle\bigwedge_{i=s}^{t} x_i \\ Y_{s,t} &= y_t \vee (x_t \wedge (y_{t-1} \vee (x_{t-1} \wedge \cdots \wedge (y_{s+1} \vee (x_{s+1} \wedge y_s)) \dots))) . \end{aligned} \tag{6}$$

Most previous constructions for adders are based on prefix graphs of small depth, size, and/or fan-out. Sklansky (1960) developed a prefix graph of minimum depth $\log_2 n$ and size $\frac{1}{2} n \log_2 n$ but high fan-out $\frac{1}{2}n + 1$. The first prefix graph with logarithmic depth $(2 \log n - 1)$ and linear size $(3n - \log n - 2)$ was developed by Ofman (1962), exhibiting a non-constant fan-out of $\frac{1}{2} \log n$. Kogge and Stone (1973) introduced the *recursive doubling algorithm* which leads to a prefix graph with depth $\log_2 n$ and fan-out two (see Figure 1(b)). Since we will use variants of it in our construction, we describe it in detail. For $1 \leq s \leq t \leq n$, let $Z_{s,t} := (X_{s,t}, Y_{s,t}) = z_t \circ \cdots \circ z_s$, and for $x \in \mathbb{R}$, let $(x)^+ := \max\{x, 0\}$. The graph has $\log_2 n$ levels, and on level i it computes for every input j ($1 \leq j \leq n$) the prefix $Z_{1+(j-2^i)^+,j}$ according to the recursive formula

$$Z_{1+(j-2^i)^+,j} = Z_{1+(j-2^{i-1})^+,j} \circ Z_{1+(j-2^i)^+,(j-2^{i-1})^+}, \tag{7}$$

from the prefixes of sequences of 2^{i-1} consecutive inputs computed in the previous level. The fan-out is bounded by two, since every intermediate result is used exactly twice: once as the "upper half" and once as the "lower half" of an expression of the form $z_j \circ \cdots \circ z_{1+(j-2^i)^+}$. Note that for level i ($1 \leq i \leq \log_2 n$), we use a repeater gate (which computes the identity function) instead of a $\circ$-gate if $j \leq 2^i$, i.e., in the case that the right input in Equation (7) is empty. The Kogge-Stone prefix graph minimizes both depth and fan-out. On the other hand, since there is a linear number of gates at each level, the total size in terms of prefix gates is $n\log_2 n - \frac{n}{2}$.

Note that the resulting adder circuit would have fan-out 3, because the underlying logic circuits uses a left propagate signal twice. However, we can bound the fan-out by two using a repeater gate at each propagate input and behind each gate A in Figure 1(a). This increases the size by small factor and the total depth by one.

Ladner and Fischer (1980) constructed a prefix graph of depth $\log_2 n$ but high fan-out. Brent and Kung found a linear-size prefix graph with fan-out two but twice the depth of the other constructions. Finally, Han and Carlson (1987) described a hybrid between a Kogge-Stone adder and a Brent-Kung adder that achieves a tradeoff between depth and size. Lower bounds for the tradeoff between the depth and size of a prefix graph can be found in Fich (1983) and Sergeev (2013).

The above prefix graphs can be used for prefix computations with respect to any associative operator $\circ$. In fact, we will later use a prefix graph in which the operator $\circ$ represents an And gate. When turning one of the above prefix graph adders into a logic circuit for addition, we replace each prefix gate with the circuit show in Figure 1(a). As a result, the depth of the logic circuit is twice the depth of the prefix graph and the number of logic gates is three times the number of prefix gates. The fan-out of the underlying logic circuit can increase by one compared to the prefix graph, because the left propagate signal x_i is used twice within a prefix gate. In Section 4.1, we will see that in the case of the Brent-Kung adder a fan-out of two can be achieved by using reduced prefix gates.

Any adder constructed from a prefix graph has a logic gate depth of at least $\log_\varphi n - 1 > 1.44 \log_2 n - 1$, where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio (Held and Spirkl 2017), see also Rautenbach et al. (2008). In Held and Spirkl (2017), we describe an adder of size $O(n \log_2 \log_2 n)$ that attains this depth bound asymptotically, however, with a high fan-out of $\sqrt{n} + 1$.

1.2 Non-Prefix Graph Adders

This section gives a brief overview of adder constructions that are not derived from a prefix graph.

Since none of the $2n$ inputs x_i, y_i ($1 \leq i \leq n$) except for x_1 are redundant for c_{n+1}, the depth of any adder circuit using 2-input gates is at least $\log_2 n + 1$, which would be attained by a balanced binary tree with inputs/leaves x_i, y_i ($1 \leq i \leq n$). With adders that are not based on prefix graphs, this bound is asymptotically tight. Krapchenko showed that any formula (a circuit with tree topology) for computing c_{n+1} has depth at least $\log_2 n + 0.15 \log_2 \log_2 \log_2 n + O(1)$ (Krapchenko 2007).

Brent (1970) gives an approximation scheme for a single carry bit circuit attaining an asymptotic depth of $(1 + \varepsilon) \log_2 n + o(\log_2 n)$ for any given $\varepsilon > 0$. The best-known depth for a single carry bit circuit is $\log_2 n + \log_2 \log_2 n + O(1)$, due to Grinchuk (2008). However, Grinchuk (2008) and Brent (1970) did not address how to overlay circuits for the different carry bits to bound the size and

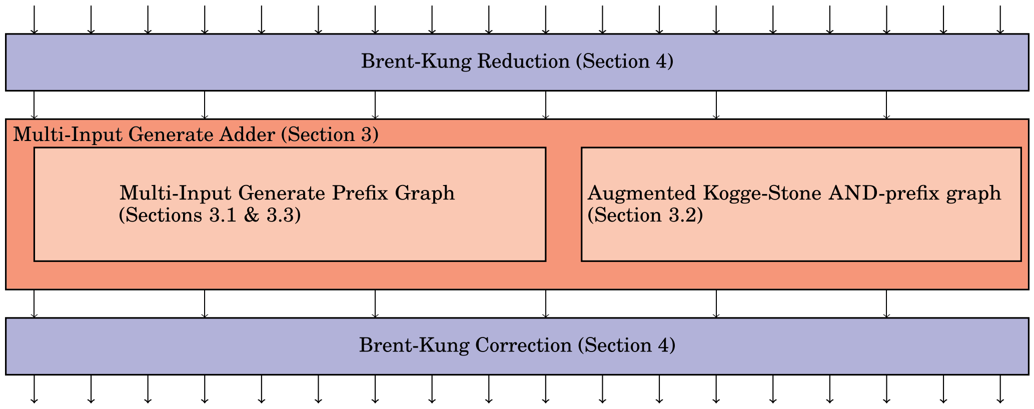

Brent-Kung Reduction (Section 4)
Multi-Input Generate Adder (Section 3)
Multi-Input Generate Prefix Graph (Sections 3.1 & 3.3)
Augmented Kogge-Stone AND-prefix graph (Section 3.2)
Brent-Kung Correction (Section 4)

Fig. 2. High-level adder description.

fan-out of an adder based on their circuits. One problem in sharing intermediate results is that this creates high fan-outs.

Krapchenko (1967) (see Wegener (1987, pp. 42–46)) presented an adder with asymptotically optimum depth $\log_2 n + o(\log_2 n)$ and linear size, and Gashkov et al. (2007) gave improvements for small n. However, the fan-out is almost linear.

1.3 Our Contribution

In this article, we present the first family of adders of asymptotically optimum depth, linear size, and fan-out bound two:

Theorem 1.1 (Main Theorem). *Given two n-bit numbers A and B, there is a logic circuit computing the sum $A + B$, using gates with fan-in and fan-out two and that has depth $\log_2 n + o(\log n)$ and size $O(n)$.*

At the core of our new adder we develop a new family of adders of asymptotically minimum depth, fan-out two, but super-linear size $O(n\lceil\sqrt{\log_2 n}\rceil^2 2^{\sqrt{\log_2 n}})$, which we present in Section 3. Compared to a traditional Kogge-Stone adder, it reduces the number of levels by using specially designed multi-input generate gates (Section 3.1). The propagate signals are computed by a separate Kogge-Stone AND-prefix graph (Section 3.2), which we augment with result duplication to bound the fan-out.

Then, in Section 4, similar to Krapchenko (1967), we transform our adder into an adder of linear size with the asymptotically same depth, thus proving Theorem 1.1. The transformation is based on Brent-Kung reductions and corrections that allow us to use a multi-input generate adder for a significantly reduced number of inputs.

While all of the above adders use only And/Or gates and repeaters, we show in Section 5 that Theorem 1.1 holds also if only Nand/Nor and Not gates are available.

2 HIGH-LEVEL OVERVIEW

A high-level schematic of our adder is shown in Figure 2. Our first step is to develop an adder with a prefix-like structure but with depth $\log_2 n + o(\log n)$. By Held and Spirkl (2017), an adder built from prefix gates cannot achieve such a depth bound, and in fact, every prefix gate contributes two layers of logic gates. Therefore, in Section 3.1, we develop a multi-input generate gate that computes the joint generate signal of 2^r inputs with depth $r + 1$ (with respect to those inputs). By using $O(\sqrt{\log n})$ layers of gates with $2^{O(\sqrt{\log n})}$ inputs each (instead of $\log_2 n$ layers with 2 inputs), we achieve a depth bound of $\log_2 n + O(\log n)$. This is proved in Section 3.3.

The next step of our construction is to bound the fan-out. Having multi-input gates also means that each output of a gate will be used many times. For generate signals, we remedy this by duplicating the output signal within the multi-input generate gate. We compute the propagate signals separately using an AND-prefix graph; and we describe an auxiliary structure providing sufficiently many copies of each propagate signal in Section 3.2. Together with the multi-input generate gate prefix graph, this forms the multi-input generate adder described in Section 3.3.

Finally, we want to bound the size of the construction. Our construction up to this point has size $O(n \log n 2^{\sqrt{\log n}})$ for n inputs. We use a reduction due to Krapchenko (1967) that allows us to choose $\tau = O(\log\log n + \sqrt{\log n})$ and construct an n-input adder by using an $(n/2^{\tau})$-input adder (which has size $O(n)$) and combining it with a circuit of depth $O(\tau)$, linear size, and fan-out two. This last step is detailed in Section 4.

3 ASYMPTOTICALLY OPTIMUM DEPTH AND FAN-OUT TWO

In this section, we present a new family of adders of asymptotically minimum depth, fan-out two, but super-linear size $O(n\lceil\sqrt{\log_2 n}\rceil^2 2^{\sqrt{\log_2 n}})$.

The adders based on prefix graphs as in Section 1.1 impose a common topological structure on the computation of intermediate results $X_{s,t}$ and $Y_{s,t}$. In the adder described by Brent (1970), on the other hand, intermediate results $X_{s,t}$ and $Y_{s,t}$ are computed separately within larger blocks.

A central idea of generating a faster adder is to use multi-fan-in (also called high-radix) subcircuits within a Kogge-Stone prefix graph. While all the prefix gates in Figure 1(b) have fan-in two, we want to use prefix gates with fan-in 2^r for some $r \geq 2$. With fan-in 2^r gates, the number k of levels reduces from $\lceil \log_2 n \rceil$ for fan-in two to $\lceil \log_{2^r} n \rceil = \lceil \frac{1}{r} \log_2 n \rceil$. In the following, we let $n = 2^{rk}$ for $r \in \mathbb{N}$ and $k \in \mathbb{N}$. We will construct adders with k rows of 2^r-input generate gates, compute their depth and size, and then select the best values for r and k for our construction.

Each prefix gate with fan-in 2^r represents a logic circuit with fan-in and fan-out bounded by two. Since the output of each prefix gate will be used in 2^r prefix gates at the next level, our approach also requires to duplicate the intermediate result at the output of a prefix gate 2^{r-1} times. To accomplish this, we consider the computation of generate and propagate sequences separately.

Our adder consists of two global Kogge-Stone type prefix graphs. The first such graph uses 2-input AND-gates and computes propagate signals (Section 3.2). The propagate signals are used in the generate prefix graph (Section 3.3), which uses 2^r-input subcircuits (Section 3.1) that are arranged in the same way as the Kogge-Stone graph.

Both graphs are modified to duplicate intermediate generate signals 2^{r-1} times and intermediate propagate signals 2^r times so that the overall constructions obeys the fan-out bound of two.

3.1 Multi-Input Generate Gates

We now introduce *multi-input generate gates*, which are the main building block for computing the generate signals, and we prove some of their basic properties. In Section 3.3, we will use them to construct adders using generalized prefix graphs. Given 2^r propagate and generate pairs $(\tilde{x}_{2^r}, \tilde{y}_{2^r}), \ldots, (\tilde{x}_1, \tilde{y}_1)$, a multi-input generate gate computes the generate signal

$$\tilde{Y}_{1,2^r} = \tilde{y}_{2^r} \vee (\tilde{x}_{2^r} \wedge (\tilde{y}_{2^r-1} \vee (\tilde{x}_{2^r-1} \wedge \cdots \wedge (\tilde{y}_2 \vee (\tilde{x}_2 \wedge \tilde{y}_1)) \ldots))) .$$

The input pairs $(\tilde{x}_i, \tilde{y}_i)$ $(i \in \{1, \ldots, 2^r\})$ are not necessarily the input pairs of the adder; they can be intermediate results.

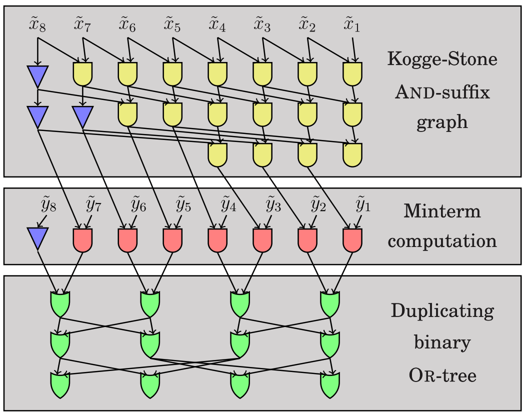

$\tilde{x}_8$ $\tilde{x}_7$ $\tilde{x}_6$ $\tilde{x}_5$ $\tilde{x}_4$ $\tilde{x}_3$ $\tilde{x}_2$ $\tilde{x}_1$
Kogge-Stone
AND-suffix
graph
$\tilde{y}_8$ $\tilde{y}_7$ $\tilde{y}_6$ $\tilde{y}_5$ $\tilde{y}_4$ $\tilde{y}_3$ $\tilde{y}_2$ $\tilde{y}_1$
Minterm
computation
Duplicating
binary
OR-tree

Fig. 3. A 2^r-input 2^{r-1}-output generate gate for $r = 3$. The top three rows are a 2^r-input Kogge-Stone AND-suffix graph.

Each multi-input generate gate has 2^{r-1} outputs, each of which provides the result $\tilde{Y}_{1,2^r}$, because later we want to reuse this signal 2^r times and bound the fan-out of each output by two. In contrast to two-input prefix gates computing (4), multi-input generate gates do not compute the propagate signals $\tilde{X}_{1,2^r} = \bigwedge_{i=1}^{2^r} \tilde{x}_i$ for the given input pairs. We compute all required propagate signals using the separate AND-prefix graph, described in Section 3.2.

Figure 3 shows an example of a multi-input generate gate with 8 inputs. A 2^r-input prefix gate computes $\tilde{Y}_{1,2^r}$ as in the disjunctive normal form

$$\tilde{Y}_{1,2^r} = \bigvee_{j=1}^{2^r} \left(\tilde{y}_j \wedge \left(\bigwedge_{i=j+1}^{2^r} \tilde{x}_i \right) \right)$$

in three steps.

In the first step, we compute the terms $\bigwedge_{i=j+1}^{2^r} \tilde{x}_i$ for $i = 0, \dots, 2^r - 1$ using a Kogge-Stone AND-suffix graph, which arises from a Kogge-Stone prefix graph by reversing the ordering of the inputs. This uses each input $\tilde{x}_i$ exactly twice.

Next, we use a single row consisting of AND gates and one repeater to compute the minterms $m_j := \tilde{y}_j \wedge (\bigwedge_{i=j+1}^{2^r} \tilde{x}_i)$ $(j = 1, \dots, 2^r)$. Each input $\tilde{y}_i$ is used exactly once within this circuit. The repeater is dispensable but simplifies the size formula and will become useful in Section 5.

Finally, instead of computing the disjunction $\bigvee_{j=1}^{2^r} m_j$ by a balanced binary OR tree and duplicating the results 2^{r-1} times through a balanced repeater tree, we accomplish the duplication using r rows of 2^{r-1} OR-gates as shown in the duplicating binary OR-tree in Figure 3. Formally, let $M_{i,j} = \bigvee_{i'=i}^{j} m_{i'}$ be the conjunction of minterms $i, i+1, \dots, j$. Then, on level $l \in \{1, \dots, r\}$, we compute each signal of the form $M_{i2^l+1,(i+1)2^l}$, $i = 0, \dots, 2^{r-l} - 1$, from the previous level, and we compute 2^{l-1} copies of it. By using $M_{i2^l+1,(i+1)2^l} = M_{2i2^{l-1}+1,(2i+1)2^{l-1}} \vee M_{(2i+1)2^{l-1}+1,(2i+2)2^{l-1}}$, and since each preceding signal is available 2^{l-2} times $(l \geq 2)$, we can ensure that each of them has fan-out two. On the last level, we will have computed 2^{r-1} copies of $M_{1,2^r} = \tilde{Y}_{1,2^r}$. Each level uses 2^{r-1} OR-gates. Note that a similar construction for reducing fan-out has been used by Lupanov when extending his well-known bounded-size representation of general Boolean functions to circuits with bounded fan-out (Lupanov 1962).

Lemma 3.1. *Given $r \in \mathbb{N}$, 2^r generate inputs $\tilde{y}_i$ and 2^r propagate inputs $\tilde{x}_i$ for $i \in \{1, \ldots, 2^r\}$, there is a multi-input generate gate G that computes the signal*

$$\tilde{Y}_{1,2^r} = \bigvee_{j=1}^{2^r} \left(\tilde{y}_j \wedge \left(\bigwedge_{i=j+1}^{2^r} \tilde{x}_i \right) \right)$$

and provides it at 2^{r-1} outputs, and each propagate input has fan-out two and each generate input has fan-out one. Moreover, the gate G consists of $r2^r + (r+1)2^{r-1}$ logic gates, each of which has fan-out at most two. The outputs are at depth $2r+1$ with respect to the propagate inputs $\tilde{x}_i$, and at depth $r+1$ with respect to the generate inputs $\tilde{y}_i$ ($i \in \{1, \ldots, 2^r\}$).

Proof. All the terms $\bigwedge_{i=j+1}^{2^r} \tilde{x}_i$ are computed as a Kogge-Stone And-suffix graph as shown in Figure 3 of size

$$2^r \lceil \log_2 2^r \rceil - \frac{2^r}{2} = (r-1)2^r + 2^{r-1}.$$

Then, there is a level of 2^r (red) And gates and one repeater, concluding the computation of the minterms. Finally, there are $r2^{r-1}$ (green) Or-gates to compute the disjunction $\bigvee_{j=1}^{2^r} m_j$ 2^r times, for a total of

$$r2^r + (r+1)2^{r-1}$$

gates. By construction, no gate and propagate input has fan-out larger than two, and all generate inputs have fan-out one. The depth is r for the And-suffix graph, one for the red gates, and r for the disjunctions, yielding the desired depths of $2r+1$ for the propagate inputs and $r+1$ for the generate inputs. □

3.2 Augmented Kogge-Stone And-Prefix Graph

The second important component of our construction is the augmented Kogge-Stone And-prefix graph, which we describe in this section. While multi-input generate gates are used to compute generate signals throughout our construction, the propagate signals (which are used by multi-input generate gates) are supplied by a separate circuit.

The *augmented Kogge-Stone* And*-prefix graph* computes $X_{s,t} = \bigwedge_{i=s}^{t} x_i$ for all $1 \leq t \leq n$ and $s = 1 + (t - 2^{rl})^+$ with $0 \leq l < k$, providing each output 2^r times through 2^r individual gates. The construction is as follows. First, we take a Kogge-Stone (1973) prefix graph, where the prefix operator is an And-gate, i.e., $\circ = \wedge$. It consists of $\log_2 n$ levels, and on level i it computes for every input j ($1 \leq j \leq n$) the prefix $X_{1+(j-2^i)^+,j}$ from the prefixes of sequences of 2^{i-1} consecutive inputs computed in the previous level.

Each of the results $X_{s,t}$ from level rl will later be used in 2^r multi-input generate gates for all $0 \leq l < k$, $s = 1 + (t - 2^{rl})^+$ and $1 \leq t \leq n$. To achieve a fan-out bound of two, starting at the inputs, we insert one row of n repeaters after every r levels of And-gates. This allows us to use the repeaters as the inputs for the next level and to extract the signals $X_{s,t}$ once at the And-gates before the repeaters. The construction is shown in Figure 4. The last block of r rows of gates of the Kogge-Stone prefix graph can be omitted in our construction to reduce the size.

Each output signal $X_{s,t}$ is the input of a multi-input generate gate, and it has fan-out two within that gate. Thus, each output $X_{s,t}$ of the augmented Kogge-Stone And-prefix graph has to be provided through an individual gate. To this end, at each of the nk outputs, we add $2^{r+1} - 1$ repeater gates as the vertices of a balanced binary tree to create 2^r copies of the signal with a single repeater serving each leaf.

Lemma 3.2. *For $k, r \in \mathbb{N}$, the augmented Kogge-Stone* And*-prefix graph with $k \cdot r$ levels computes the signal $X_{s,t}$ for all $1 \leq t \leq n$ and $s = 1 + (t - 2^{rl})^+$ for all $0 \leq l < k$. It obeys the fan-out bound*

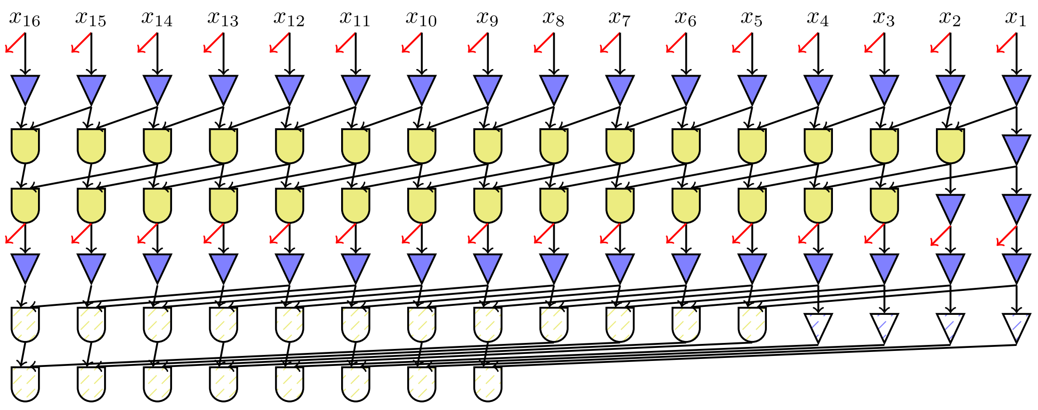


Fig. 4. The augmented Kogge-Stone AND-prefix graph for $r = k = 2$, with the extracted outputs $X_{s,t}$ shown as red arrows. The last block of r rows of gates is hatched, and, for simplicity, repeaters providing multiple copies of each output signal are hidden.

of two and its size is $nr(k-1) + nk2^{r+1}$. *The output signals* $X_{s,t}$ *are provided* 2^r *times at a depth of* $(l+1)(r+1)$.

PROOF. The functional correctness and and the fan-out bound are clear by construction. For the size, note that each binary repeater tree at one of the nk outputs consists of $2^{r+1} - 1$ repeaters, summing up to $nk(2^{r+1} - 1)$ repeaters in these repeater trees. The remaining circuit consists of $r(k-1)$ rows of AND-gates and k rows of repeaters. Each row consists of n gates, summing up to $n(r(k-1) + k)$ gates. Altogether, the circuit contains $nr(k-1) + nk2^{r+1}$ gates.

For the depth bounds, let $1 \le t \le n$ and $0 \le l < k$. Then, for $s = 1 + (t - 2^{rl})^+$, the signal $X_{s,t}$ is available at the bottom of the lth block at a depth of $l(r+1)$. Subsequently, we create 2^r copies of the signal in a repeater tree of depth $r+1$. Together, this gives the desired depth $(l+1)(r+1)$. □

3.3 Multi-Input Generate Adder

We now describe the multi-input generate adder for $n = 2^{rk}$, which is an adder of low depth and fan-out two; later, we will show how to reduce its size without compromising the first two properties. The adder consists of an augmented Kogge-Stone AND-prefix graph from the previous section and a circuit consisting of multi-input generate gates similar to a radix-2^r Kogge-Stone adder.

The construction uses k rows, each with n multi-input generate gates or repeater trees (see Figure 5). The tth multi-input generate gate in level $l \in \{1, \ldots, k\}$ computes $Y_{1+(t-2^{rl})^+,t}$ according to the formula $Y_{1+(t-2^{rl})^+,t} =$

$$\bigvee_{j=1}^{2^r} \left(Y_{1+(t-j2^{r(l-1)})^+,(t-(j-1)2^{r(l-1)})^+} \wedge \left(\bigwedge_{k=j+1}^{2^r} X_{1+(t-k2^{r(l-1)})^+,(t-(k-1)2^{r(l-1)})^+} \right) \right). \tag{8}$$

If $(t - 2^{rl})^+ < (t - 2^{r(l-1)})^+$, then this computation is carried out using a multi-input generate gate from Section 3.1. As its inputs, it uses generate signals from the previous level, $l-1$, and propagate signals obtained from the augmented Kogge-Stone AND-prefix graph.

Except for the last level, each intermediate generate signal will be used 2^r times as in Equation (8) in the next level. As the fan-out of each generate input inside a multi-input generate gate is one, we need to provide 2^{r-1} copies through individual gates to serve 2^r multi-input generate gates with fan-out two.

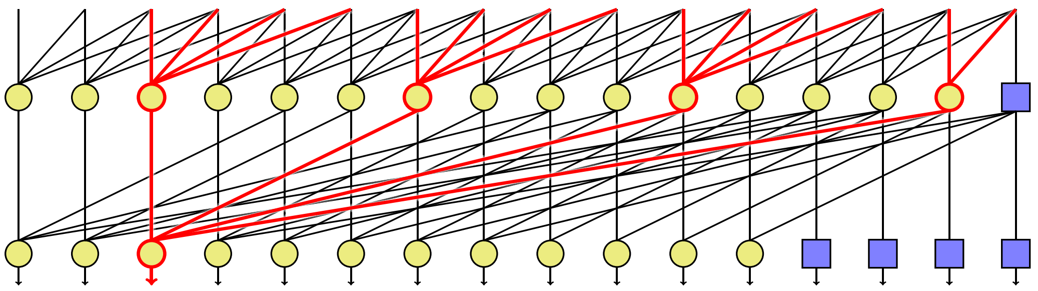


Fig. 5. A multi-input multi-output generate gate adder for $r = k = 2$, where yellow circles represent multi-input generate gates, and blue squares represent balanced binary repeater trees. The ancestors of one output are highlighted in red.

If $(t - 2^{rl})^+ = (t - 2^{r(l-1)})^+$, then the already-computed $Y_{1+(t-2^{rl})^+,t}$ at the previous level, and in this level it is sufficient to duplicate the signal 2^{r-1} times using a balanced binary repeater tree.

The augmented Kogge-Stone And-prefix graph provides each signal 2^r times with individual repeaters. Thus, it can be distributed to 2^r multi-input generate gates, where the fan-out of each propagate input is two.

For the first level of multi-input generate gates, we duplicate each generate signal y_i at an input $i \in \{1, \dots, n\}$ using a balanced binary repeater tree of depth $r - 1$ and size $2 + 2^2 + \cdots + 2^{r-1} = 2^r - 2$. Again, we can distribute each copy to two multi-input generate gates, maintaining fan-out two.

In the last level of multi-input generate gates, we do not need to duplicate the signals anymore. Instead of the r rows of 2^{r-1} Or-gates each, we can compute the single outputs using a balanced binary tree of $2^r - 1$ Or-gates and depth r.

Lemma 3.3. *The multi-input generate adder for $n = 2^{rk}$ bits ($r, k \in \mathbb{N}$) obeys a fan-out bound of two, contains less than*

$$3nk(r + 2)2^{r-1} + n2^r + nrk$$

gates and has depth

$$kr + 2r + k + 1.$$

Proof. Inside each multi-input generate gate, the fan-out of propagate inputs is two and the fan-out of generate inputs is one. Thus, it suffices to observe that in each non-output level there are 2^r copies of each propagate signal and 2^{r-1} copies of each generate signal and that the fan-out of two holds within the augmented Kogge-Stone graph and within each multi-input generate gate.

By Lemma 3.2, the size of the augmented Kogge-Stone And-prefix graph is $nr(k - 1) + nk2^{r+1}$. The size of the n balanced binary trees duplicating the input generate signals is $n(2^r - 2)$.

The remainder of the graph consists of k rows of n 2^r-input multi-input generate gates or repeater trees. The size of a repeater tree is at most $2^{r-1} - 1 \leq r2^r + (r + 1)2^{r-1}$ ($r \geq 1$), which is the size of a multi-input generate gate. Thus, the size of all these multi-input generate gates is at most $nk(r2^r + (r + 1)2^{r-1})$. Summing up, the total size is at most

$$\begin{aligned} & nr(k-1) + nk2^{r+1} + n(2^r - 2) + nk(r2^r + (r+1)2^{r-1}) \\ =\ & nk2^{r+1} + nkr2^r + n2^r + nk(r+1)2^{r-1} + nkr - n(r+2) \\ =\ & nk\,(4 + 2r + (r+1))\,2^{r-1} + n2^r + nkr - n(r+2) \\ <\ & 3nk\,(r+2)\,2^{r-1} + n2^r + nkr. \end{aligned}$$

For a simpler depth analysis, we assume that the input generate signals y_i arrive delayed at a depth of $r + 2$. The generate input signals traverse a binary tree of depth $r - 1$ and the propagate input signals traverse a binary tree of depth $r + 1$ before reaching the first multi-input generate gate, i.e., generate signals y_i become available at depth $2r + 1$ and propagate signals at depth $r + 1$. Thus, the first row of multi-input generate gates has depth

$$3r + 2 = \max\{2r + 1 + 1 + r, r + 1 + r + 1 + r\},$$

where the first term in the maximum is caused by the delayed generate signals y_i and the second term by the propagate signals x_i $(1 \leq i \leq 1)$.

For the next level, the propagate signals are available at time $2r + 2$, and the generate signals at time $3r + 2$, and the propagate signals again arrive r time units before the corresponding generate signals, so at the next level, both signals arrive $r + 1$ time units later than they did before. Inductively, we know that for each level $2 \leq l \leq k$, the generate and propagate signals arrive at a depth of $(l - 1)(r + 1)$ more than they did for at the first level. Consequently, the total depth of the adder is $(k - 1)(r + 1) + 3r + 2 = kr + 2r + k + 1$. □

If $\sqrt{\log n} \in \mathbb{N}$, then we can choose $r = k = \sqrt{\log n}$ and receive the following result.

Corollary 3.4. *If $\sqrt{\log n} \in \mathbb{N}$, then there is a multi-input generate adder for n bits with fan-out two, size at most*

$$3n(\log n + 2\sqrt{\log n})2^{\sqrt{\log n}-1} + n2^{\sqrt{\log n}} + n \log n,$$

and depth

$$\log n + 3\sqrt{\log n} + 1.$$

In general, $\sqrt{\log n} \notin \mathbb{N}$, and we get the following result.

Theorem 3.5. *Let $n \in \mathbb{N}$ be a power of two. For input pairs (x_i, y_i) $(i \in \{1, \ldots, n\})$, there is a circuit, computing all carry bits with maximum fan-out 2, depth at most*

$$\log_2 n + 5\left\lceil\sqrt{\log_2 n}\right\rceil + 2.$$

The size is at most

$$4n\left\lceil\sqrt{\log_2 n}\right\rceil^2 2^{\left\lceil\sqrt{\log_2 n}\right\rceil} \tag{9}$$

if $n \geq 16$ and at most

$$8n\left\lceil\sqrt{\log_2 n}\right\rceil^2 2^{\left\lceil\sqrt{\log_2 n}\right\rceil} \tag{10}$$

if $n \leq 15$.

Proof. We choose $r = k = \lceil\sqrt{\log_2 n}\rceil$. By Lemma 3.3 and $r = k$, we obtain

$$3nk(r + 2)2^{r-1} + n2^r + nrk = n(3(r^2 + 2r)2^{r-1} + 2^r + r^2). \tag{11}$$

Now, if $n \geq 16$, then we have $r = k \geq 2$. Thus, we can use $2r \leq r^2$ and $2^r + r^2 \leq r^2 2^r$ to bound the right-hand side by

$$n(3(r^2 + r^2)2^{r-1} + r^2 2^r) = 4nr^2 2^r,$$

implying Equation (9).

Otherwise, $n \leq 16$, $r = k \leq 2$, $r^2 \leq 2r$, $r^2 \leq 2^r$, and the right-hand side of Equation (11) is bounded by

$$n\,(3\,(2r + 2r))\,2^{r-1} + 2^r + 2^r = 8nr2^r \leq 8nr^2 2^r.$$

By Lemma 3.3, the resulting depth is bounded by

$$\begin{aligned} kr + 2r + k + 1 &= \left\lceil \sqrt{\log_2 n} \right\rceil^2 + 3\left\lceil \sqrt{\log_2 n} \right\rceil + 1 \\ &\le \left(\left\lfloor \sqrt{\log_2 n} \right\rfloor + 1 \right)^2 + 3\left\lceil \sqrt{\log_2 n} \right\rceil + 1 \\ &\le \log_2 n + 5\left\lceil \sqrt{\log_2 n} \right\rceil + 2. \end{aligned}$$

□

If $\sqrt{\log_2 n} \notin \mathbb{N}$, then the adder in Theorem 3.5 is larger than necessary, since it has $n' = 2^{\lceil \sqrt{\log_2 n} \rceil^2} > n$ inputs. If, for example $n = 32$, then we choose $r = k = 3$ and $n' = 512$. However, if $\lceil \sqrt{\log_2 n} \rceil^2 \ge n + \lceil \sqrt{\log_2 n} \rceil$, then choosing $r = \lceil \sqrt{\log_2 n} \rceil - 1$ instead still yields an adder with at least n inputs and outputs and reduces the size and depth significantly. For $n = 32$, we would still obtain a 64-input adder using this method. The following lemma shows how to decrease the size and depth further.

Lemma 3.6. *Let r, k in $\mathbb{N}$ such that $n \le 2^{rk}$, where n is a power of two. Then multi-input generate adder for $n' = 2^{rk}$ bits can be modified to obey a fan-out bound of two to contain less than*

$$3nk(r+2)2^{r-1} + n2^r + nrk$$

gates and to have depth

$$\log_2 n + 2r + k + 1.$$

Proof. The columns n' down to $n + 1$ in the augmented Kogge-Stone And-prefix graph and the multi-input gate graph can be omitted, since they are not used for the computations of the first n output bits. This reduces its size from $n'r(k-1) + n'k2^{r+1}$ to $nr(k-1) + nk2^{r+1}$. Moreover, the size of the repeater trees duplicating input signals is at most $n(2^r - 2)$. The multi-input generate gates in columns n' down to $n + 1$ can be removed as well, and so the total size of all multi-input generate gates is at most $nk(r2^r + (r+1)2^{r-1})$. Thus, the size bound follows as in Lemma 3.3; clearly, this does not increase the fan-out.

For the depth bound, note that if $n' > n$, then we can omit the left half of the construction and notice that the right half of the lowest row of multi-input generate gates has only 2^{r-1} inputs, so we can actually use 2^{r-1}-input generate gates and reduce the depth by 1. This process can be iterated until $n' = n$; the depth decreases by at least $rk - \log_2 n$. This implies the result of the lemma. □

In this section, we have achieved a depth bound of $\log_2 n + O(\sqrt{\log n}) = \log_2 n + o(\log_2 n)$, which is asymptotically optimal, since the lower bound is $\log_2 n$.

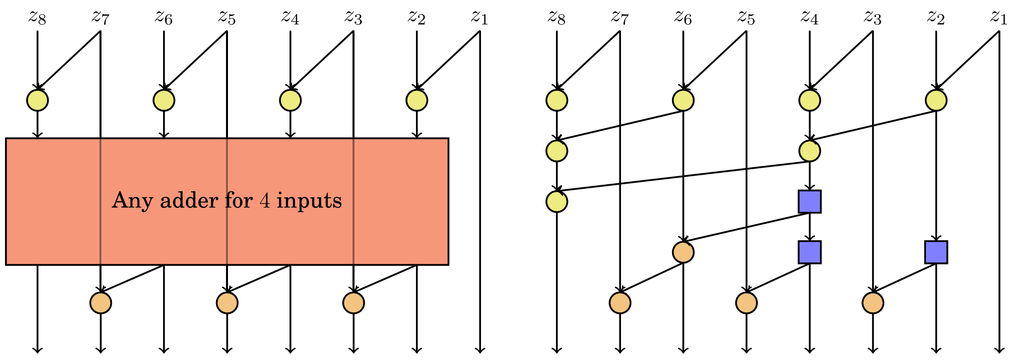
4 LINEARIZING THE SIZE OF THE ADDER

To achieve a linear size while preserving an asymptotically optimum depth, we adopt a technique similar to the construction by Brent and Kung (1982), which was first used as a size-reduction tool by Krapchenko (1967) (see Wegener (1987, pp. 42–46)).

4.1 Brent-Kung Step

We first describe a single reduction step, which can be used to transform an n-input adder into a $2n$-input adder.

Brent and Kung (1982) construct a prefix graph recursively as shown in Figure 6(a). If n is a at least two, then it computes the $n/2$ intermediate results $z_n \circ z_{n-1}; \dots; z_2 \circ z_1$ (see Section 1.1 for the definition of z_i). We use a prefix graph for these $n/2$ inputs to compute the prefixes $Z_{1,2i}$ for all even indices $i \in \{1, \dots, n/2\}$. For odd indices, the prefix needs to be corrected by one more prefix gate as $Z_{1,2i+1} = z_{2i+1} \circ Z_{1,2i}$ ($i \in \{1, \dots, n/2 - 1\}$). We call this method of input halving and output



(a) A Brent-Kung (reduction) step, which reduces the number of inputs outputs by a factor of two.

(b) The Brent-Kung prefix graph, obtained by repeatedly applying Brent-Kung reduction steps.

Fig. 6. A Brent-Kung step and the resulting prefix graph.

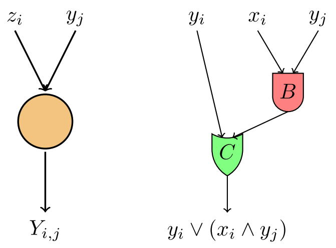


Fig. 7. The reduced output correction prefix gate of a refined Brent-Kung step with propagate signal computation omitted.

correction a *Brent-Kung step*. Note that the propagate signals are not needed after the correction step. Thus, we can use reduced prefix gates (Figure 7) in the output correction step. In these prefix gates, the left propagate signal x_i is used only once. Thus, the underlying logic circuit inherits the fan-out of two from the prefix graph.

The Brent-Kung step reduces the instance size by a factor of two, but it increases the depth of the construction by four and the size by $(5/2)n$ in terms of logic gates.

Applying these Brent-Kung steps recursively, Brent and Kung obtain a prefix graph that has prefix gate depth $2\log_2 n - 1$ and logic gate depth $4\log_2 n - 2$. The prefix gate depth is no longer optimal, but the adder has a comparatively small size of $\frac{1}{2}(5n - \log_2 n - 8)$ gates, and its fan-out is bounded by two at all inputs and gates. It is shown in Figure 6(b).

Brent-Kung steps were actually known before the article by Brent and Kung (1982), e.g., they were already used in Krapchenko (1967). But the Brent-Kung adder is based solely on these steps.

4.2 Krapchenko's Adder

In this section, we state a lemma that can be used to achieve a tradeoff of depth and size of an adder. This lemma is part of the construction of Krapchenko's adder, a non-prefix adder computing all carry bits with asymptotically optimal depth and linear size. Its fan-out, on the other hand, is almost linear as well, which makes it less useful in practice. Krapchenko's techniques can be used to derive the following reduction, based on Brent-Kung steps.

Lemma 4.1 (Krapchenko 1967, see Wegener (1987, pp. 42–46)). *Let $n \in \mathbb{N}$ be a power of two and $\tau \leq \log_2 n - 1$. Then given a family of adders computing k carry bits with depth $d(k)$, maximum fan-out $f(k)$ and size $s(k)$, there is a family of adders computing n carry bits with depth $d(n/2^\tau) + 4\tau$ and size $s(n/2^\tau) + 5n$.*

With size $s(n/2^\tau) + (11/2)n$, we can achieve the same depth and a maximum fan-out of at most $\max\{2, f(n/2^\tau)\}$.

Proof. We apply τ Brent-Kung steps and construct the remaining adder for $n/2^\tau$ from the given adder family. Figure 6(a) shows the situation for $\tau = 1$. The simple application of τ Brent-Kung steps would achieve the claimed depth and fan-out result, except with at most $2n$ additional 2-input prefix gates (because we will never add more prefix gates than are present in the Brent-Kung prefix graph) and thus with $6n$ additional logic gates.

To see that $5n$ logic gates are enough, we show that we can omit the propagate signal computation for the parity-correcting part of the Brent-Kung step. Such a reduced output prefix gate is shown in Figure 7. With this construction, note that for i even, we have computed $(x, y) = z_i \circ \cdots \circ z_1$. For $z_{i+1} = (y_{i+1}, x_{i+1})$, the carry bit arising from position $i + 1$ is $c_{i+2} = x_{i+1} \vee (y_{i+1} \wedge y)$, which uses two gates. It follows that a Brent-Kung step uses only the propagate signals at the inputs. For the next Brent-Kung step, the inputs are the $n/2$ pairs $z_n \circ z_{n-1}; \ldots; z_2 \circ z_1$, and therefore we need three logic gates per prefix gate for the reduction step.

Note that in Figure 6(b), the propagate signal at a gate is used if and only if there is a vertical line from this gate to another prefix gate (and not to an output or repeater). These lines exist only in the "upper half" of the adder, i.e., the parts with depth at most $\log_2 n$. Since parity correction occurs exclusively in the lower half with depth greater than $\log_2 n$, the propagate signals from parity correction steps are never used.

As in the Brent-Kung prefix graph, $\frac{n}{2}$ repeaters can be used to distribute the fan-out and reduce the maximum fan-out of the parity-correcting gates to two (see also Figure 6(b)). □

The fact that the refined Brent-Kung step does not require the inner adder to provide the propagate signals, which a prefix graph adder would provide, allows us to use the multi-input generate adder with the size and depth bounds stated in Theorem 3.5, where the last r rows of And gates in Figure 4 in the augmented Kogge-Stone And-prefix graph are omitted.

Lemma 4.1 can be used to achieve different tradeoffs. In particular, constructions for all carry bits of size up to $n^{1+o(1)}$ can be turned into linear-size circuits with the same asymptotic depth or depth guarantee, since we could choose $\tau = o(\log_2 n)$. This works for prefix graphs and logic circuits; for example, with $\tau = \log_2 \log_2 n$, the Kogge-Stone prefix graph will have size $3n$, depth $\log_2 n + 2\log_2 \log_2 n$ and fan-out bounded by two in terms of prefix gates (Han and Carlson 1987).

While the technique in Lemma 4.1 is essentially a 2-input prefix gate construction, the main result of Krapchenko (1967) cannot be constructed using only prefix gates.

4.3 Adders with Asymptotically Minimum Depth, Linear Size, and Fan-Out Two

In this section, we combine Theorem 3.5 and Lemma 4.1 to obtain our main result: an adder of asymptotically minimum depth, linear size, and with fan-out at most two.

Theorem 4.2. *Let $n \in \mathbb{N}$ be a power of two. There is an adder for n inputs of size bounded by $(27/2)n$ with depth*

$$\log_2 n + 8\left\lceil \sqrt{\log_2 n} \right\rceil + 6\left\lceil \log_2 \left\lceil \sqrt{\log_2 n} \right\rceil \right\rceil + 2$$

and maximum fan-out two. If $n \geq 4{,}096$, then the size can be bounded by $(19/2)n$.

Proof. We apply Lemma 4.1 with $\tau = \lceil\sqrt{\log_2 n}\rceil + 2\lceil\log_2\lceil\sqrt{\log_2 n}\rceil\rceil$ and use an adder for $n/2^\tau = 2^{-\tau}n$ inputs according to Theorem 3.5 as an inner adder. From the proof of Lemma 4.1, we have seen that the output correction of the Brent-Kung step does not require propagate signals from the inner adder. So the fan-out is indeed two. Using Equation (10), this results in an adder of size

$$\begin{aligned}
&\left\lceil 8n \cdot 2^{-\tau} \cdot 2^{\lceil\sqrt{\log_2(2^{-\tau}n)}\rceil} \cdot \left\lceil\sqrt{\log_2(2^{-\tau}n)}\right\rceil^2\right\rceil + (11/2)n \\
&\leq \left\lceil 8n \cdot 2^{-\tau} \cdot 2^{\lceil\sqrt{\log_2 n}\rceil} \cdot \left\lceil\sqrt{\log_2 n}\right\rceil^2\right\rceil + (11/2)n \\
&= \left\lceil 8n \cdot 2^{-\lceil\sqrt{\log_2 n}\rceil} \cdot 2^{-2\lceil\log_2\lceil\sqrt{\log_2 n}\rceil\rceil} \cdot 2^{\lceil\sqrt{\log_2 n}\rceil} \cdot \left\lceil\sqrt{\log_2 n}\right\rceil^2\right\rceil + (11/2)n \\
&\leq 8n + (11/2)n = (27/2)n.
\end{aligned}$$

If $n \geq 4{,}096$, then we have $n/2^\tau \geq 16$ that allows us to apply the alternative bound (9) to achieve a size bound of $(19/2)n$.

The depth is

$$\begin{aligned}
\log_2(2^{-\tau}n) + 5\left\lceil\sqrt{\log_2(2^{-\tau}n)}\right\rceil + 2 + 4\tau &= \log_2 n + 5\left\lceil\sqrt{\log_2(2^{-\tau}n)}\right\rceil + 2 + 3\tau \\
&= \log_2 n + 8\left\lceil\sqrt{\log_2 n}\right\rceil + 6\left\lceil\log_2\left\lceil\sqrt{\log_2 n}\right\rceil\right\rceil + 2,
\end{aligned}$$

using the definition of τ. □

From Theorem 4.2, we can easily conclude our main result stated in Section 1.3:

Theorem 1.1 (Main Theorem). *Given two n-bit numbers A,B, there is a logic circuit computing the sum $A + B$, using gates with fan-in and fan-out two and that has depth* $\log_2 n + o(\log n)$ *and size* $O(n)$.

5 TECHNOLOGY MAPPING

In this section, we show that our construction from Theorem 4.2 can be transformed into an adder using only Nand/Nor, and Not gates, which are faster than And/Or gates and repeaters in current CMOS technologies. This increases the depth by one and the size by a small constant factor.

Theorem 5.1. *Let $n \in \mathbb{N}$ be a power of two. There is an adder for n inputs using only* Nand, Nor, *and* Not *gates. Its size is bounded by $(18 + \frac{1}{3})n$, the depth is at most*

$$\log_2 n + 8\left\lceil\sqrt{\log_2 n}\right\rceil + 6\left\lceil\log_2\left\lceil\sqrt{\log_2 n}\right\rceil\right\rceil + 3,$$

and the maximum fan-out is two. If $n \geq 4{,}096$, then the size is bounded by $(15 + \frac{5}{6})n$,

In the next two lemmas, we show how to transform the two main components of our construction, the Brent-Kung steps and the multi-input multi-output generate gate adder, into circuits using only the desired gates.

Lemma 5.2. *Brent-Kung reduction and correction steps can be implemented using only* Nand *and* Not *gates achieving the same depth an less than $\frac{5}{3}$ the number of gates as with* And *and* Or *gates.*

Proof. Brent-Kung reduction steps can be implemented using Nand/Not prefix gates as shown in Figure 8. Similarly, the reduced output correction gate in Figure 7 can be realized by two Nand gates and one Not gate, i.e., by eliminating the two rightmost gates in Figure 8. The modified

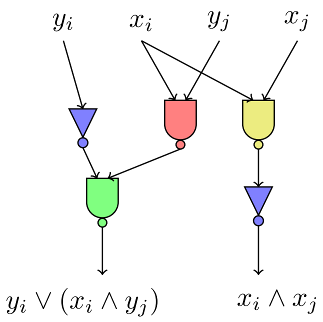


Fig. 8. A NAND/NOT prefix gate used for a modified Brent-Kung reduction step.

prefix gates do not increase the depth of the Brent-Kung step and increase the size by a constant factor less than $\frac{5}{3}$. □

Similarly, we can implement the adder from Theorem 3.5 using only NAND, NOR, and NOT gates, increasing the depth by one and size by a factor $\frac{5}{3}$.

LEMMA 5.3. *Let $n \in \mathbb{N}$. For input pairs (x_i, y_i) $(i \in \{1, \dots, n\})$, there is a circuit using only* NAND, NOR, *and* NOT *gates, computing all carry bits with maximum fan-out 2, depth at most*

$$\log_2 n + 5\left\lceil\sqrt{\log_2 n}\right\rceil + 3.$$

The size is at most

$$\frac{5}{3}4n\left\lceil\sqrt{\log_2 n}\right\rceil^2 2^{\left\lceil\sqrt{\log_2 n}\right\rceil}$$

if $n \geq 16$, and at most

$$\frac{5}{3}8n\left\lceil\sqrt{\log_2 n}\right\rceil^2 2^{\left\lceil\sqrt{\log_2 n}\right\rceil}$$

if $n \leq 15$.

PROOF. We transform multi-input multi-output generate gate adders from Theorem 3.5 applying DeMorgan's laws. For easier understanding, we first insert repeaters so that the gates can be arranged in rows, such all input signals for gates in odd rows are computed in even rows and vice versa. Here row zero refers to the input signals x_i, y_i $(i = 1, \dots, n)$. This bipartite structure is already given in the augmented Kogge-Stone AND-prefix graph (see Figure 4).

Let us now consider a multi-input generate gate as shown in Figure 3. By inserting $2^r/2$ repeaters gates in the last row of the AND-suffix graph, we achieve a uniform depth for this first stage. The red row of AND gates and the final 2^{r-1}-output OR already have a uniform depth. The additional repeaters increase the size by less than a factor of $\frac{5}{3}$. Except for the first row of generate gates, the depth of the generate signals equals the depth of the propagate signals when they are merged in the red row of AND gates. In the first row of generate gates, the propagate signals arrive there at depth $2r + 1$, while the generate signals arrive at depth $r - 1$ (see the proof of Lemma 3.3). Thus, if r is odd, we add one additional repeater at every generate input signal so that it arrives at an odd depth at the red level of AND gates. Note that we can do this without increasing the overall depth, as we already assumed that the generate signals are delayed by $r + 1$ in the proof of Lemma 3.3. At most n repeaters are inserted this way.

Some gates of the multi-input generate gate adder are repeater trees. They have depth $r - 1$, which is odd if and only if the depth $r + 1$ of the corresponding paths of generate signals through multi-input generate gates is odd. Thus, they preserve the bipartite structure.

Now we can use the bipartite structure to transform the multi-input multi-output generate adder into a circuit consisting of Nand, Nor, and Not gates. In our construction we will maintain the following characteristics. Inputs to an odd row, i.e., outputs of an even row, will be the original function values, while inputs to an even row, i.e., outputs of an odd row, will be the negated original function values. We achieve this by transforming gates as follows: Repeaters are always transformed into Not gates. In odd rows, we translate And gates into Nand gates and Or gates into Nor gates. In even rows, we translate And gates into Nor gates and Or gates into Nand gates. If the number of rows is odd, then we add one row of Not gates to correct the otherwise negated outputs of the adder.

Together with the n repeaters that we insert after each generate input signal if r is odd, this constitutes $2n$ gates that can by accounted for by the size of the augmented Kogge-Stone And-prefix graph (see Figure 4), which is at least $3n$ if $r \geq 1$. Thus, the overall size of the generate adder increases by a factor of at most $\frac{5}{3}$. □

In combination with the mapping of the Brent-Kung step in Lemma 5.2, this proves Theorem 5.1.

Conclusion

We introduced the first full adder with an asymptotically optimum depth, linear size, and a maximum fan-out of two. Asymptotically, this is twice as fast and significantly smaller than the Kogge-Stone adder, which is often considered the fastest adder circuit, as well as most other prefix graph adders.

For small n, Theorem 4.2 will not immediately improve on existing adders. When focusing on speed for small n, one would rather omit the size reduction from Section 4. Without the size reduction, our results in Lemma 3.3 match the depth of the Kogge-Stone adder for 512 inputs and improve on it for 2,048 inputs, where $r = 3, k = 4$ yields an adder with depth 21 for our construction, but the adder of Kogge-Stone will have depth 22.

Today's microprocessors usually contain adders for at most a few hundred bits. However, adders for 2,048 bit numbers are used for cryptographic chips. Thus we expect that adders based on our ideas will find their way into hardware.

REFERENCES

R. P. Brent. 1970. On the addition of binary numbers. *IEEE Trans. Comput.* 19, 8 (1970), 758–759.

R. P. Brent and H.-T. Kung. 1982. A regular layout for parallel adders. *IEEE Trans. Comput.* 100.3 (1982): 260–264.

S. Chatterjee, A. Mishchenko, R. Brayton, X. Wang, and T. Kam. 2006. Reducing structural bias in technology mapping. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 25, 12 (2006), 2894–2903.

F. E. Fich. 1983. New bounds for parallel prefix circuits. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC'83).* ACM.

S. B. Gashkov, M. I. Grinchuk, and I. S. Sergeev. 2008. On the construction of schemes for adders of small depth. *Diskr. Anal. Issledov. Operat. Ser.* 1, 14, 1 (2007), 27–44 (in Russian). [English translation in *J. Appl. Industr. Math.* 2, 2 (2008, 167–178).

M. I. Grinchuk. 2008. Sharpening an upper bound on the adder and comparator depths. *Diskr. Anal. Issledov. Operat. Ser.* 1, 15.2 (2008): 12-22 (in Russian). [English translation in *J. Appl. Industr. Math.* 3, 1 (2009), 61–67.]

T. Han and D. A. Carlson. 1987. Fast area efficient VLSI adders. In *Proceedings of the8th IEEE Symposium on Computer Arithmetic*, 49–56.

S. Held and S. Spirkl. 2017. Fast prefix adders for non-uniform input arrival times. *Algorithmica* 77, 1 (2017), 287–308.

H. J. Hoover, M. M. Klawe, and N. J. Pippenger. 1984. Bounding Fan-out in Logical Networks. *J. ACM* 31, 1 (1984), 13–18.

K. Keutzer. 1987. DAGON: technology binding and local optimization by DAG matching. In *24th ACM/IEEE Design Automation Conference.* 341–347.

S. Knowles. 1999. A family of adders. In *Proceedings of 14th IEEE Symposium on Computer Arithmetic.* 277–281.

P. M. Kogge and H. S. Stone. 1973. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Comput.* C-22, 8 (1973), 786–793.

V. M. Krapchenko. 1970. Asymptotic estimation of addition time of a parallel adder. *Probl. Kibern.* 19 (1967), 107–122 (in Russian). [English translation in *Syst. Theor. Res.* 19 (1970), 105–122.]

V. M. Krapchenko. 2008. On possibility of refining bounds for the delay of a parallel adder. *Diskr. Anal. Issledov. Operat. Ser.* 1, 14, 1 (2007), 87–93. [English translation in *J. Appl. Industr. Math.* 2.2 (2008): 211–214.]

R. E. Ladner and M. J. Fischer. 1980. Parallel prefix computation. *J. ACM* 27, 4 (1980), 831–838.

O. B. Lupanov. 1963. A class of schemes of functional elements. *Probl. Kibernet.* 7 (1962), 61–114. [English translation in Probl. Cybernet. 7 (1963), 68–136.]

Y. P. Ofman. 1963. The algorithmic complexity of discrete functions. *Dokl. Akad. Nauk SSSR* 145, 1 (1962), 48–51. [English translation in *Sov. Phys. Dokl.* 7 (1963): 589–591.]

D. Rautenbach, C. Szegedy and J. Werber. 2008. On the cost of optimal alphabetic code trees with unequal letter costs. *Eur. J. Combinator.* 29, 2 (2008), 386-394.

I. Sergeev. 2013. On the complexity of parallel prefix circuits. In *Electronic Colloquium on Computational Complexity*, Vol. 20.

J. Sklansky. 1960. Conditional-sum addition logic. *IRE Trans. Electr. Comput.* 2 (1960), 226–231.

I. Wegener. 1987. *The Complexity of Boolean Functions.* Wiley-Teubner (1987).

J. Werber, D. Rautenbach, and C. Szegedy. 2007. Timing optimization by restructuring long combinatorial paths. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design.* 536–543.

Received January 2017; revised September 2017; accepted September 2017